Artificial Intelligence Methods in Process Plant Layout

by Andrew M^{c}Brien, BSc

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy, April, 1994

Table of Contents

Abstract

The thesis describes "Plant Layout System" or PLS, an Expert System which automates all aspects of conceptual layout of chemical process plant, from sizing equipment using process data to deriving the equipment items' elevation and plan positions. PLS has been applied to a test process of typical size and complexity and which encompasses a wide range of layout issues and problems. The thesis presents the results of the tests to show that PLS generates layouts that are entirely satisfactory and conventional from an engineering viewpoint.

The major advance made during this work is the approach to layout by Expert System of any kind of process plant. The thesis describes the approach in full, together with the engineering principles which it acknowledges.

Plant layout problems are computationally complex. PLS decomposes layout into a sequence of formalised steps and uses a powerful and sophisticated technique to reduce plant complexity. PLS uses constraint propagation for spatial synthesis and includes propagation algorithms developed specifically for this domain. PLS includes a novel qualitative technique to select constraints to be relaxed. A conventional frame based representation was found to be appropriate, but with procedural knowledge recorded in complex forward chaining rules with novel features. Numerous examples of the layout engineer's knowledge are included to elucidate the epistemology of the domain.

Acknowledgements

This thesis marks the culmination of my formal education throughout which my parents, Donald and Margaret, have supported and encouraged me. They instilled the desire to learn in me which ultimately led to this research. This is an excellent time to thank them for all they have done for me throughout the years.

I commenced this work while an Associate of a Teaching Company between the University of Nottingham and Zyqad Limited, and completed it while an employee of Zyqad. I would like to thank the Chairman of Zyqad, Mr Jim Madden, who made it possible for me to explore this fascinating subject and who granted permission for me to report the work.

I would also like to thank Dr Carl Pulford, my supervisor for the majority of this work. His advice on the direction and structure of the work has been invaluable, his proof-reading has been meticulous and swift and his comments have always been thoughtful and thought-provoking.

I would particularly like to thank my wife, Angela, who has accepted her status as a "thesis widow" with such good grace. Without her constant great patience and understanding, I could not have completed this thesis.

The untimely death of my original supervisor, Dr John Mecklenburgh, left the discipline of Process Plant Layout without one of its key authorities. My frequent references to his classic book reflect the contribution he made to the field.

Chapter 1 : Introduction

Traditionally, a chemical process plant is designed by two almost independent functions, often referred to as the "process design" and "plant design" functions. The process design function designs and enumerates a specification of the process to be operated in the plant. This specification includes the type of equipment needed to perform each stage; the service conditions within the items, such as temperatures, pressures and compositions; the characteristics of the items which define how they are to achieve their function, such as the number of trays in a column, or the heat transfer area in a heat exchanger; the streams of process media flowing between the items and the means by which the process will be controlled.

The plant design function derives a physical design for the plant. This conforms to the requirements imposed by the functional specification of the process and its components. Plant design includes the selection of equipment types to perform the required process duty, such as the type of pump to perform a stated pumping duty; the detailed mechanical design of non-standard items of equipment; the selection of piping components and the routing of pipes to carry the streams; civil design and structural design.

The task of determining the spatial arrangement of the items within the plant is often referred to as "conceptual" or "preliminary" layout. It is the first step in converting the functional specification of the process into the three dimensional, physical plant design. Taken together, process design and conceptual layout constitute the so-called "front-end" design phase.

Front-end design is characterised by being highly creative and offers the designer significant degrees of freedom. Thereafter, design becomes increasingly deterministic. Many of the requirements that the plant designers must meet arise directly from the front-end solutions. Similarly, the options available to the plant designers are heavily constrained by features of the front-end design. Accordingly, although only a small percentage of the total project budget has been spent once front-end design is complete, the majority of the budget is committed. One study showed that typically, 20% is spent and 80% committed [Craft 1985]. Clearly, the quality of the front-end design is instrumental in the commercial success of a project. Layout is an important element in this and Mecklenburgh [1985] states that a good layout provides

> "a plant that is safe and efficient to construct, operate and maintain, whilst making effective use of the land available"

and adds

> "a well thought-out layout also contributes to the successful planning of both engineering design and construction."

However, he warns that

> "a bad layout will probably lead to an unsuccessful and unsafe venture."

The process industry operates in an increasingly stringent commercial and legislative environment. More sophisticated products are manufactured to gain market differentiation. These are made by processes which are likely to be based on complex chemistry conducted using aggressive conditions and involving hazardous compounds. This has made process and plant engineering more difficult. At the same time, the general public and regulatory bodies such as the British Health and Safety Executive are demanding higher standards of safety and reduced environmental impact in plant design and operation. Additionally, pressure has increased to reduce capital and operating costs.

Nonetheless, the layout engineer still relies on individual experience and flair. Until very recently, his only tools were simple physical analogues used to visualise the layout as it emerged. This contrasts with other design disciplines which have adopted sophisticated computerised systems. These systems support increased optimisation of the process by simulation and modelling and implement elaborate data recording and change control systems. They have improved design consistency and quality and made design data more accessible to review and validation.

These positive experiences of computerisation provide compelling arguments that conceptual layout should be furnished with similarly sophisticated computer support. Certainly, the almost archaic current approach employed by the conceptual layout engineer is hardly commensurate either with the computer systems employed by the other disciplines or with the importance of high quality layout designs.

The problem is exacerbated further. It is a paradox that the process designers require three-dimensional physical data to develop their functional model of the process from which the layout will ultimately be derived. Detailed data are required to support specific calculations. For example, elevation differences are required to calculate pump hydraulics and thermosyphon reboiler boil-up rates. It is also likely that a number of alternative processes will be developed and compared before one is chosen for design to completion. Rose [1978] argues that approximate layouts should be developed to support these comparisons. The technical feasibility and suitability of each process option for the intended site may be assessed more fully and the precision of cost estimation and hazard assessments increased.

Even though the potential benefits may be great, it is difficult to justify the significant investment in time required to develop provisional layouts at the early stages of a project. The process engineers compensate by including significant factors of safety in the process design. In many cases, this leads to over-design which reduces efficiency and economy throughout the plant's life cycle. The lack of precision in the technical, economic and safety assessments introduces uncertainty and risk into perhaps major investment decisions.

1.1 The Original Work

Many computer systems have been constructed which attempted to automate layout of both process plant and other facilities. Many of these embodied computing techniques at the forefront of development at the time they were constructed. Even so, none was practicable for conceptual process plant layout. Many were designed for other applications with few features in common with conceptual layout. These were inappropriate for process plant layout although many succeeded in their intended uses. Others were designed specifically for conceptual layout. These were probably the least successful because the computing technology available at the time could not match the needs of the domain.

Since then, Artificial Intelligence techniques have matured and can now be used to solve problems of the large scale and complexity of the conceptual layout task. In particular, Expert Systems are now in common use solving real problems including various design tasks. Brachman *et al* [1983] present the features that distinguish Expert Systems both from Artificial Intelligence in general and from conventional procedural programming. There are compelling arguments that we are much more likely to succeed in building a computer system to automate conceptual layout if we use Expert System technology than previous computing techniques.

The overall brief for the original work was to progress as far as possible towards an Expert System that might be used commercially to generate automatically three dimensional conceptual layouts from process data. A system such as this would require two major components. It would require a generic structure which implemented the system's problem solving approach. No precedent existed for this structure so original research would be required to develop it from first principles. The system would also require a knowledge base which embodied a wide range of the knowledge and practices of layout engineers would also have to be elicited and encoded. A knowledge base that would be sufficiently broad to be used in practice would require many man-years of elicitation and coding to catalogue the knowledge of practising layout engineers. A large team funded commercially would be required to build a knowledge base such as this. It was decided at the outset of this work that it was most appropriate for the author to concentrate on the intellectually challenging research required to develop the generic problem solving structure. Thus, the following three specific objectives, which were actually intimately linked in practice, were set for the original work reported in this thesis:

Objective 1
To construct a prototype for an Expert System which substantially automates process plant layout. The techniques that engineers currently use and the information that they manipulate and derive were to be considered. Representations for the data and knowledge, and techniques for reasoning were to be selected or developed, combining a number of each if necessary. The priority was to develop the underlying problem-solving techniques of the system.

Objective 2
To use the prototype to lay out a test case plant to demonstrate the efficacy of the approach adopted. The test plant was to be similar in size and complexity to a typical conceptual layout task. The plant was to encompass a wide range of layout issues and problems to ensure the generality of the prototype.

Objective 3
To establish a general approach to automating process plant layout which can be embodied in any Expert System, reflecting the design of the prototype, and to identify the features and constraints of the domain which any Expert System must recognise.

The work was directed toward developing a generic Expert System to support conceptual layout in the broadest sense. An alternative strategy might have been to construct a "point application" to support a very specific layout task, such as the repeat design of packaged systems such as air separation units. However, a system dedicated to one type of plant would only benefit engineers directly engaged in laying out examples of that type. Furthermore, other software developers could learn little from work of that type.

1.2 The Structure of the Work

When the approach that an experienced engineer uses to lay out a process plant is studied, it is apparent that it would be difficult to emulate it in a computer program, whether procedural or knowledge based. A fundamental approach to laying out process plant by Expert System was therefore developed from first principles during this work. The development of this approach ultimately comprised a major part of the original work. The working prototype was constructed in parallel with the development of the underlying approach. This prototype was called Plant Layout System or PLS. As the approach took shape, its raw concepts were converted into algorithms which were built into PLS. PLS catalysed and guided further development of the approach and served as a test bed for the approach as it developed. Thereby, the development of PLS and the underlying approach became intimately linked.

A knowledge base was constructed during the experimental work specifically to support the testing of PLS. This knowledge base includes knowledge that a layout engineer might use to derive preliminary designs and estimate sizes of items of process equipment. This knowledge was elicited from standard chemical engineering texts such as Perry [1984] and the very useful "Equipment Design Handbook" [Evans 1979]. The knowledge base also includes knowledge specific to plant layout, such as knowledge to identify the constraints on an item's position. In the main, this was elicited from practising layout engineers and from the standard texts on plant layout by Mecklenburgh [1985] and Kern [1977]. Numerous examples of PLS's knowledge will be provided to convey an understanding of the types of knowledge that are required in this domain. The knowledge base is not presented in detail within the thesis because it mainly records well documented and understood engineering principles. It is important to note, nonetheless, that engineers design a layout by recalling solution fragments which they have learned over many years. They no longer consider the underlying principles directly and rarely reason about the fundamental factors that govern the position of an item of equipment. The layout texts mainly record these solution fragments. PLS is designed to reason from the fundamental factors to a solution. This brings many benefits to the user and allows an appropriate and efficient implementation. These factors had to be teased out of the solution fragments during knowledge elicitation - the published information could not be encoded directly. Thus, building the knowledge base was a substantial task of itself.

When the work commenced, no shell or programming environment existed that offered the functionality that was required by PLS. Reynolds, in consultation with the author, constructed a knowledge representation language called "IRIS", designed specifically as the implementation vehicle for this work. IRIS itself is not reported in this thesis because it is neither the work of the author nor an essential element of PLS. The ways in which IRIS was used to represent layout knowledge and data in PLS are described in Chapter 5. The demands that the domain places on a knowledge representation language are highlighted.

PLS was applied to a test process of the size and complexity of a typical conceptual layout task and generated an entirely satisfactory layout. This test constitutes the formal results of the experimental work and is reported in full in Chapter 3. However, the major intellectual content, and benefits, in this work lie in the principles of PLS's operation. These can be employed in any Expert System for the same domain. The majority of the thesis concentrates on these principles to show the design requirements that the domain places on an Expert System and to describe the fundamental approach which meets these requirements very effectively. The prototype of PLS is discussed throughout the thesis to elucidate the requirements; to expand on the description of the approach; to substantiate the approach as implementable and efficacious and to indicate particular difficulties that arise.

IRIS has been developed into a proprietary system since the completion of this work.

Chapter 2 : Literature Review

One technique is observed almost universally for layout development. It is entirely manual. The engineer employs simple tools to visualise the layout as it emerges. Traditionally, these have been either two-dimensional cardboard cut-outs or crude three-dimensional physical block models. Very recently, the use of three-dimensional computer models has been reported. The first three sections of this review chapter discuss the current technique for conceptual layout, relate it to general models of design as a procedure and describe why such apparently crude tools are still highly favoured.

Systematic methods have been developed for factory and other layout work. The methods are historically important in their own right. They also provide the concepts embodied in a number of computer programs developed to assist the layout engineer by maintaining a record of the evolution of the design, providing some level of feedback and presenting the layout. The methods are described in the fourth section of this chapter, the programs which embody them in the fifth. The latter have never been successfully employed but collectively demonstrate a need if not a means of meeting it.

An enormous range of computer systems have been developed to support or even automate specific tasks which may be considered to be examples of design or configuration. These are well reviewed elsewhere [Coyne 1991]. It is unnecessary to replicate this review in this thesis. Specific techniques and results from these other systems will be referred to as appropriate within the description of the original work in the following chapters. The central element of PLS is its automated space planning capability. It is appropriate therefore to include a discussion of the wide range of programs developed for this. Some of these systems were developed specifically for application to process plant layout, others to general space planning tasks. The programming techniques range from numerical optimisation used in the seminal programs of the early 1960s through to advanced symbolic processing used today. Automation of process plant layout then of general automated space planning is reviewed in the last two sections of this chapter.

2.1 The Observed Technique

No practical systematic procedure is known that may be used to derive directly an optimal layout of a process plant. Certain essential tasks must be conducted to complete a layout from the essentially fixed start point. These tasks are conducted in substantially the same order by all designers, and all designers appear to approach each in a like manner. Effectively, an *ad hoc* technique has evolved. Within each task however, the actions are performed with little procedural structure. In some, even the means by which the actions are performed are poorly understood.

The tasks are presented here as a linear procedure. In practice, the steps are not as clearly delimited as this somewhat pedagogical description implies. Separate sections of the plant may be considered concurrently. The design may be progressed to a different stage in each section at any time. The designer may blur the distinctions between the steps, attempting more than one simultaneously.

The description of the technique presented here encompasses all steps likely to be observed. In certain circumstances, some steps may be unnecessary and be omitted. In extreme cases, the designer may do little more than identify a closely similar plant or plant section previously laid out and reiterate the previous design with a few, appropriate modifications. This approach is widely observed for example in the design of standardised modular or packaged systems [Bredbury 1986].

This section is presented to contextualise the original work of this research. It is not intended as a detailed exposition of the engineering issues facing the layout designer or the accepted solutions that have developed over time. Readers interested in these details are referred to the excellent books by Mecklenburgh [1985] and Kern [1977] which have achieved the status of classics in this field. The recent book by Bausbacher and Hunt [1993] which reflects more current practice is also highly recommended. These books comprise effectively all of the reasonably current publicly published material on process plant layout and the majority of this section is elicited from them. The author's personal experiences as a practitioner in the field and (perforce general) remarks on three major contractors' design standards for layout viewed by the author [Standards 1989a, 1989b, 1989c] are also incorporated.

2.1.1 Information Assimilation

The flowsheet and its attendant data represent the primary data input to conceptual layout. The flowsheet presents a description of the process, defining the equipment types and their interconnections. The equipment is described in terms of its functional specifications, such as the number of trays in a column, the heat transfer area of a heat exchanger, or the residence time of a vessel. At this stage, it is unlikely that physical designs of equipment will be available. The attendant data also includes process data, such as stream flow rates, compositions and service conditions. Generic data are also accumulated. These include the physical properties of the process media at the process service conditions and flammability and toxicity data.

Data describing the site (or sites) on which the plant is to be (or may be) built form a secondary input, if known. These data may include the space available, ground contour, soil loading and drainage, the positions at which service and process pipework may be connected to existing pipe tracks, and the positions of installations and off-site features susceptible to damage in the event of an on-plant incident.

The process and site data are assimilated from their diverse sources by the layout engineer as the first step in devising the layout.

2.1.2 Preliminary Equipment Design

The process data define functional specifications of the equipment to comprise the plant. They rarely prescribe either how the function of an equipment item is to be achieved or a physical design for that item. The PFD may show, for example, one pump. This represents a statement that pumping is required, rather than that one physical pump is to be incorporated into the plant. In the physical realisation, additional standby pumps may be added. In particular, the process data does not define the physical space requirements of the items, or the types and positions of fittings and scantlings. These must be inferred.

The procedure is best elucidated by an example. Consider a storage vessel with a bayonet heater. The operating volume may be specified in the process data, but not the dimensions. The bayonet heater will be shown on the flowsheet, but not nozzles and manholes. The preliminary design of the vessel proceeds through a number of steps:

1. Allowances for ullage and volume occupied by the heater are added to the operating volume to calculate the total volume of the vessel.
2. The dimensions of the vessel are determined from the total volume by rule of thumb, such as the typical length being three times diameter, or from tables of standard preferred vessel sizes. Note that these dimensions are provisional. They may be changed during layout so that the vessel may better fit available space or during detailed mechanical design.
3. The fittings of the vessel, such as its nozzles and manhole, are identified and sized. For example, the flowsheet will show streams connected to the vessel. Each connection must be implemented by one or more nozzles, and the layout engineer infers their existence. The bore of each nozzle can then be calculated from the stream flow rate and density and a knowledge of standard nozzle sizes. Similarly, the need for a manhole may be inferred

by assessing the potential for fouling of the heater tube bundle and thereby, the need for access to the tubes for cleaning.

4. The ancillaries and fittings are positioned on the vessel. Again, these positions are provisional.
5. The enclosing volume of the vessel is determined by adding the dimensions by which the ancillaries and fittings stand proud from its surface.
6. The free space required around the vessel to allow access to it and the ancillaries is determined. Note that this free space may be shared with other items in the final layout. Note also that the vessel has different free space needs for different operations conducted on it. The layout must allow for these space needs at the time they may be required. For example, the space allowed for withdrawal of the heater tube bundle for maintenance may be filled by the service pipework which must be removed to allow the withdrawal.

The space required by the vessel is derived specifically for the purpose of conceptual layout. The preliminary design of the vessel and the position of its fittings and ancillaries constitutes both a design assumption for the layout engineer and design requirement for detailed engineering.

2.1.3 Analyzing Process Flow

In a process plant, the cost of the pipework connecting the equipment contributes a significant proportion of the total capital cost. The pipework also occupies a significant proportion of the total volume of the plant. The flowsheet represents connectivity as streams. The stream implies the need for a physical connection in the plant. The length of pipe runs is reduced if the equipment positions in the layout emulate the topology of the major flows on the flowsheet. The engineer identifies these major flows to generate an overall form for the layout prior to attempting to position any items within this form.

The engineer must consider the properties of the stream to determine the relative cost of implementing the associated pipe run and hence the relative influence of the stream on the layout. Streams with high volumetric flow rate imply intrinsically more expensive large bore pipework. These streams are readily apparent from the flowsheet.

Other factors suggesting the importance of the flow are identified only after consideration of more fundamental data. The nature of the stream or even of the connected equipment may render a relatively low flow rate stream important in influencing the layout. Particulates entrained in a liquid stream may be susceptible to deposition in the pipework and corrosive streams imply expensive corrosion resistant pipework for example. The flow between a vessel and its discharge pump may also require proximity to minimise pressure drop to ensure sufficient head at the pump's inlet.

The major flows need not be those that follow the progress of the process media from raw material to finished state. For example, the recycle stream around a low yield reactor will exceed the product flow from the reactor loop and may dominate the layout.

It is likely that the designer will mark the principal flows on a copy of the flowsheet. This serves purely as a personal *aide memoire* to be discarded once the layout is complete.

2.1.4 Deriving Elevation

It is almost universal practice to derive equipment positions in two separate phases - the almost entirely deterministic calculation of equipment elevation, and the highly creative and unstructured derivation of plan position.

Elevation differences between equipment items may be imposed by the process design. A necessary pressure differential defines the height of a barometric leg, for example. The differences may also be imposed by hydraulics. A vessel may be elevated to meet the NPSH requirement of the pump it feeds. Almost universally, the minimum elevation differences can be calculated, and from these, the minimum absolute elevation of each item.

Where two items have a known elevation difference, the minimum absolute elevation assigned to the upper is taken to be the sum of the elevation of the lower and the difference. An item may have elevation difference requirements above more than one other. The sum of the elevation and difference is calculated for each lower object. The maximum of the sums is assigned to the upper. This procedure is started by assigning grade elevation to those which need not be elevated.

Support for an elevated item may be provided either by a major item (such as a column) or by floors. In the latter case, the elevation of the item is increased so that it is above the preferred elevation of the next higher floor.

In general, equipment items elevated above grade require support structures, elevated access platforms and walkways and more substantial foundations. This increases plant construction costs. Recognising this, it is common practice to accept the minimum absolute elevation of an item as being its actual elevation. Infrequently, the engineer may decide that an item should be raised to an elevation above its minimum. It may then be supported on a major item or on a structure required by other items, perhaps. Other economies (particularly in piping runs) may be achieved. Similarly, floor elevations may be manipulated to maximise the number of items that they may support.

2.1.5 Identifying Other Factors

The relative positions of items may also be constrained by factors which do not arise from their connectivity. For example, heat exchangers are often placed together rather than within the units they serve. This permits their tube withdrawal facilities to be shared. The positions of equipment items relative to site or off-site features may also be constrained. For example, items subject to severe wind loading may be attracted to an area of good load-bearing soil to minimise piling requirements.

Other issues require the segregation rather than proximity of the related equipment items or equipment items and site features. For example, a fired heater constitutes a potential cause of ignition. A potential source of a release of flammable vapour is constrained to be at least a calculable separation distance away from such a heater.

These constraints impose preferred positions and orientations on the plant as a whole as well as influencing its internal layout. Both Kern and the design standards viewed suggest a second inspection of the flowsheet to identify such constraints.

The features of the equipment and site features giving rise to the constraints may not be explicit or even readily apparent in these process data. The layout engineer must infer the causes of constraints from the available data then identify the constraints that arise from these causes.

It is worthy of note that the effects of many of these constraints are implicitly recognised in design standards. The configurations that have been found in the past to satisfy them are prescribed as approved or even requisite elements of a layout. Such configurations have almost acquired the status of idiom.

The author [1989a] recommends a unified and homogeneous treatment of flow and other issues in the form of constraints. As a lecturer on a professional development course on plant layout, he has observed approximately 100 engineers attempt layout exercises and has collected significant informal evidence of the efficacy of this. Those participants who specialise in conceptual layout are rarely willing to apply the strategy during exercises. Those participants who are required to perform conceptual layout as only a small part of their work often adopt it fully. The specialists usually achieve a solution to the exercises quicker than the other participants. The latter, however, are found to be less likely to overlook perhaps subtle issues, achieve solutions in which the impact of design considerations more accurately reflects their true relative importance and are more able to discuss and justify their solution once they reach it.

2.1.6 Plan Positioning

Even a simple plant will comprise forty to sixty items of process equipment. The simultaneous positioning of this many items far exceeds the capacity of the engineer. He adopts a number of abstract views of the plant. These abstractions represent differing levels of detail and different physical areas of the plant, often considered concurrently.

Each process unit may be considered initially as a single problem element. The major flows between the units suggest an approximate overall topology of the plant's pipe racks. The positions of the units are readily derived from this topology. To facilitate both construction and escape in the event of an incident, it is desirable to lay out the pipe racks with bends and corners minimised. Positioning the units is almost simplified to ordering them along relatively straight racks. The general linearity has removed many of the degrees of freedom. The engineer assumes sizes for the unit operations and groups and positions them as single elements.

This layout may be distorted to incorporate the effects of the attraction or repulsion of certain units to or from specific site features. For example, tall columns requiring extensive foundations may be positioned together on areas of good load-bearing ground, or fired heaters may be positioned far from administrative buildings to minimise casualties in the event of an explosion.

The engineer also mentally decomposes the equipment inventory into natural groups. Again, these may be manipulated initially as a macroscopic entity. The group may represent an aggregation of items positioned within a tangible physical feature of the plant, such as a compressor house or pump bay. Alternatively, engineering standards or good practice (perhaps even idiom) may suggest its members. Grouping exchangers and positioning them with their heads aligned makes possible the use of a common gantry crane to remove their tube bundles. This practice is frequently observed. A module or packaged unit is arguably a physical manifestation of a group - its members are manipulated collectively on site, as well as during design.

The positions of the units and natural groups in the outline of the layout provide approximate reference points around which the members of each may be laid out. Each section is considered separately, although probably none laid out to completion prior to others being considered. While working on one unit, the engineer readily shifts or enlarges his focus to incorporate equipment from others where these influence its internal layout. This integrates the layout of each group into that of the plant as a whole. Conversely, the engineer may minimize the number of items under consideration at any time if their relative positions are either particularly difficult to derive or their correct relative positioning is crucial to the success of the layout as a whole. The procedure is both *ad hoc* and opportunistic - as the engineer gains an increased understanding of the requirements of the solution and gathers information on the possible solutions, more precise and committing decisions become possible.

The engineer will constantly review the initial assumption of the outline of the layout while positioning individual items. In particular, the assumed size of groups and units will be checked as their internal layout emerges. Where the assumption proves invalid, the outline layout is modified. Of course, modifications may also arise spontaneously. In particular, the engineer may develop a layout fragment to satisfy one constraint then recognise a configuration that may be extended to incorporate the satisfaction of another. This has been identified as an important behaviour in design, and given the amusing name of "Aha!" design [Kant 1982].

In some cases, the sizes assumed for the groups will be found to be widely inaccurate. The modifications to the outline layout will then be extensive, effectively invalidating the layout at the level of the individual items of equipment. This can be obviated substantially by forming groups in a separate step prior to positioning. The groups may be formed either to represent a unit operation, an aggregation of equipment required by design standards or a physical feature. Each group is laid out internally in isolation to derive its approximate size. This technique significantly increases the likelihood of the assumed size remaining valid [McBrien 1989a].

The positions of the equipment within these preliminary groups are likely to be sub-optimal, of course. No account is taken of the "boundary conditions" imposed by relationships with members of other groups. Items of equipment within each often need to be re-positioned to integrate the isolated layouts of each group into a global optimum once the groups have been relatively positioned.

The discussion so far has emphasised the fairly deterministic approach of the designer of probably external plant whose layout is dominated by relatively simple patterns of high volume flow. This is typical of major petrochemical plants, for example. More creativity is required to lay out plant where other constraints approach flow in importance, or where the flow patterns are convoluted.

The procedure followed by the designer in these cases is much less clear, although the techniques already described are applied to some extent. Protocol analyses in the related domain of architectural design and building layout suggest that the designer addresses highly localised problems and generates almost independent solutions. Two models of the derivation of these partial solutions have been proposed.

The designer may apply rules similar to the production rules of an expert system which specify the action taken to satisfy a particular constraint [Freeman 1971]. An example of such a rule in plant layout might be

"If flow between two vessels is driven by gravity then position the vessels such that the outlet of the feeding vessel is above the inlet of the fed vessel."

The rule specifies a solution procedure to be applied locally to satisfy one or very few constraints.

The alternative model [Foz 1973] suggests that the designer recognises specific configurations of constraints and recalls partial solutions previously developed for that configuration. The partial solutions state directly the positions for the elements involved. The author believes that this is widespread in process plant layout. It would account in part for the ability of the experienced engineer to achieve acceptable designs quickly. Furthermore, both Kern and Mecklenburgh present many solution fragments and elucidate the circumstance in which each is applicable, collated from the experience of the authors. The ready acceptance of these works implies a naturalness of this presentation to practising engineers.

Evidence now suggests that both approaches are employed and that these models are complementary but only partial descriptions [Akin 1978].

The positions of major structural elements are also derived at this time. Note however, that these positions represent the design requirements passed on to the civil discipline rather than a formal structural design.

2.1.7 Review and amendment

Many thousands of constraints will exist between the items of equipment comprising a typical process flowsheet. The layout designer lacks the mental capacity rigorously and exhaustively to consider this number. He must concentrate on those constraints that he considers most important. Perhaps inevitably, a designer will be more mindful of the constraints arising from the issues most pertinent to his discipline.

When the layout is complete, many constraints may remain unsatisfied simply because they have been overlooked. It is quite feasible that the designer will have omitted to consider constraints of importance but alien to his own speciality. Other constraints may have been satisfied, but serendipitously. We can assume that the layout proposal will be sub-optimal.

To mitigate this, a multi-disciplinary panel assesses the proposal. Each member considers those constraints specific to his discipline. The panel identifies deficiencies in the layout which is then refined. The iteration proceeds until the solution is acceptable to all parties.

Typically, a number of candidate layouts will be designed initially. Each may conform to a specific design philosophy or consider specific aspects of the layout, such as operability, economy or safety. The most promising candidates are selected, refined further, subjected to further selection, and so on. In one particularly difficult case, a total of thirty two proposals were generated [Thompson 1989].

2.1.8 Disseminating The Layout

Once accepted, the layout design is disseminated to those plant design disciplines dependent upon it. Traditionally, the layout is presented on a general arrangement drawing often referred to as the "Plot Plan". This shows all major equipment items, major structures and buildings, although in outline only. The battery limits of the plot area are indicated together with roadways, access ways, extent of paving, pipe entry and exit points, maintenance areas, stairways and ladders. Elevation drawings and sections, in both plan and elevation, are also drawn. Other forms of presentation may also be used, such as physical or computer three-dimensional models. All convey much the same information as the Plot Plan and elevation drawings which they replace or augment.

2.2 Relationship To Design Studies

Numerous models of design as a process have been proposed. Those due to Asimow [1962], Luckman [1967] and Markus *et al* [1972] are both important and representative. There is widespread agreement that design occurs in three phases - analysis, synthesis and evaluation. In the analysis phase, the designer seeks to understand the problem and produce an explicit statement of goals. Plausible solutions are synthesized during the second phase. The solutions are compared against the identified goals in the third stage and preferred alternatives selected. The models all imply a cycle in which the solution is revised and improved by re-examining the analysis.

It is interesting to note that these phases are readily apparent in the technique for conceptual layout. A more detailed protocol analysis of architectural design [Akin 1978] distributes the three phases amongst six steps, outlined below.

1. <u>Information Acquisition</u> - the designer reads the problem specification and clarifies specification requirements.
2. <u>Problem Interpretation</u> - the designer transforms the problem requirements into a consistent format suitable to drive generation of the solution, conceptually identical to the designer identifying "constraints" [Eastman 1970], "criteria" [Archer 1968], "goals" [Simon 1973] or "requirements" [Alexander 1968].
3. <u>Problem Representation</u> - the designer adds new parameters to the design to represent those data which must be considered to allow the manipulation of the problem requirements or to allow new, more detailed design requirements to be postulated.
4. <u>Solution Generation</u> - the designer generates partial solutions which represent either localised complete solutions or solutions to the problem as a whole but considering only a subset of the design requirements.
5. <u>Solution Integration</u> - the designer integrates the partial solutions to form larger or more complete solution fragments.

6. Solution Assessment - the designer assesses the solutions against their satisfaction of the design requirements and the feasibility of integrating their component partial solutions.

Akin observed that steps 4 to 6 are actually conducted iteratively. The partial solutions are integrated with others and assessed as they are developed. The partial solutions are modified if necessary prior to the designer moving on to address other aspects of the design.

The technique for conceptual layout presented above was identified by informal observation and introspection. Nonetheless, it conforms very closely to the results of Akin's rigorous study in a closely related field.

2.3 Visualisation Tools

The layout designer reasons about three-dimensional configurations of the plant equipment in developing a layout proposal. He augments his powerful abstract spatial reasoning faculties with physical or computer representations of the design. The representations allow him to "play" with the "equipment" in space and apply the visual spatial reasoning skills learnt from childhood. Furthermore, as the engineer moves his focus of attention through different sections of his design, it is imperative that he may recall the solution developed for any one section when he returns to it. Similarly, he must remain aware of the design as a whole at all times so that he may achieve a unified and integrated solution. The capacity of the designer's memory is unlikely to be sufficient to allow this without aid. The visual representation of the evolving layout serves as a record and aide memoir.

If the layout is being developed by a team, a visual presentation of the design is obviously essential to allow communication between the members. It also serves as an aid to discussion of the finished proposal prior to formal drawings being prepared.

The representations range from simple two-dimensional shapes representing the "footprint" of the equipment to applications of powerful commercial CAD packages. The alternatives are reviewed in this section.

2.3.1 Physical Analogues

The traditional visualisation tool comprises cardboard shapes cut out to scale to represent the "footprint" of the equipment items. These are positioned on a piece of squared paper which may have site features drawn onto it. While crude in appearance, this tool is actually of enormous benefit to the user. The designer may manipulate the shapes as he develops the design in his mind, effectively "trying out" partial solutions for the layout. The cost is negligible, the scale shapes may be made very quickly once the equipment sizes are known and the user obviously requires no specific training.

Cardboard shapes are very widely used, particularly to lay out plant predominantly constructed on a single level. Indeed, this technique is also employed by architects to lay out floors of a building. In this case, each floor is effectively an independent, two-dimensional problem. Multi-floor plant must often be treated as a single three-dimensional problem. Many equipment items, such as columns, pierce the floors. The layout of each floor influences significantly that of others through these tall items. Multi-floor plant and plant where elevation constraints significantly influence the layout require a tool that adequately represents the third dimension. Very simple three-dimensional models may be constructed using simple blocks, often now of expanded polystyrene. The volume occupied by each item of equipment is represented by either a cylinder or cuboid, whichever is more appropriate. These blocks may be elevated on wires forced into them or floors may be added to the model on which the blocks may be stood. These block models are again very cheap and easy to use. The detail is very limited, particularly in comparison with the piping models which may be constructed late in the project. Nonetheless, it is commensurate with the precision at which the conceptual layout designer works.

Both tools are highly effective but both suffer from the same deficiency. As physical analogues manipulated manually, neither may be integrated into the increasingly computerised design environment employed by the process industry. Design consistency and quality has improved markedly as a result of this increased computerisation, principally because of the data integrity management and change control it offers. Conceptual layout is critical to the success of the project as a whole. At this critical step, data are extracted from the computerised design environment, manipulated manually then the results re-entered into the system. There is significant risk of transcription error and no data integrity or change management may be implemented.

2.3.2 Commercial CAD Packages

The uptake of computer packages in conceptual layout has lagged far behind their enthusiastic application in almost all other aspects of both process and plant design. Recent publications describe two contractors performing project engineering almost exclusively with CAD [Briggs 1987, Taffe 1990] while explicitly stating that it is not employed for conceptual layout.

Two-dimensional CAD packages require a great deal more setting up than the cardboard cutouts and are difficult to use to "sketch" a design. They are rarely used in conceptual layout other than to prepare the Plot Plan and other drawings once the layout is complete.

The principal requirement of the layout designer is ease of manipulation and visualisation of the model. The physical analogues described above are ergonomically very close to the ideal. Until recently, the technology of commercial three-dimensional CAD systems and the computer hardware on which they run was insufficiently advanced to provide similar functionality. The user positioned elements in the model by entering coordinates and translations via the keyboard. A view of the emerging model could only be generated by switching the system into a graphical output mode. Three-dimensional CAD systems have long been considered inappropriate for conceptual layout by its practitioners [Taffe 1990].

Latterly, computer graphics power has reached the level where it is feasible to use CAD as the tool with which the layout is developed rather than just represented as previously. Two alternative approaches have been reported in the literature.

ICI have used three-dimensional CAD for all layout development during a project from the initial rough flowsheet to the final issued design [Atkinson 1987]. They used their own (but commercially available) system "Provue-3D" which was developed principally for piping design and detailed layout engineering. Models of each equipment item are constructed in Provue-3D from a number of primitives such as cylinders and cuboids. Initially, the full power of the system was not employed. The process engineers estimated the size of the major vessels and represented them using a single primitive of the estimated size. These were positioned on screen graphically, almost "sketching" the layout. Modifications were made simply by pulling the primitives around the model. The lack of detail in the model initially allowed very rapid setup and system response. As the project progressed and more information became available, the models of the equipment were refined, nozzles were added from the piping component catalogues, major pipe runs sketched in, *etc.* The system was used very informally by intent and the final layout design was achieved solely using the computer model.

In an alternative approach, Foster Wheeler have used their three-dimensional modelling system (Intergraph's "PDS") to bring great detail into the conceptual layout quickly [Russo 1992]. Their PDS installation has been enhanced by numerous libraries of parametric three-dimensional equipment models. The user starts by drawing the major features of the plot as a two-dimensional "base board" for the model and marks the plan view with the positions of the equipment items. The positions are actually derived prior to this using the conventional technique. The user then commences to construct a detailed model of the plant.

The parametric routines prompt the user for those process data required and construct fully detailed models of the equipment items, including scantlings. The dialogues may take between two minutes and one hour for each item of equipment, depending on its complexity. The equipment models are added at the positions previously marked for them.

The finished model may be passed directly to the piping and structural engineers. In this approach, conceptual layout has been closely integrated with the project phase and one potentially error prone transcription of data has been eliminated. The parametric equipment models employed in this approach replace manual equipment sizing. This task is both highly deterministic and time consuming, suggesting it to be ideal for computerisation. The designer is freed to concentrate on the more creative aspects of his work. Indeed, this implementation may be viewed as an embryonic knowledge based system for conceptual layout.

2.4 Systematic Techniques

Systematic techniques for design in general have been a long-standing goal. For example, Mostow [1985] argues that design pervades our society and probably costs billions of dollars annually. Moreover, the cost of design errors in lives and property is untold. He concludes that scientific study and formalization of design is easily justified by its potential for improving cost or reliability of design.

A systematic method for conceptual layout offers the potential for obviating many of the deficiencies of the current technique. In particular, it may mitigate the current difficulty of verifying and validating a proposal. Many systematic general design techniques have been devised. Three are designed specifically for layout problems and are discussed here. It will be shown unfortunately that none are practically applicable to conceptual layout.

2.4.1 The Relationship Chart

Muther [1961] recognised the concept of relationships between objects governing their relative positions. He proposed the "Relationship Chart" to record their existence and relative importance as they are discovered manually. A triangular matrix is constructed. One row and one column is ascribed to each item of equipment. For every relationship discovered between a pair of items, an entry is made at the intersection of one's row with the other's column. This entry may be simply a tick, or better, a rough quantification of the importance of the relationship. A completed chart should contain all constraints on the positions of any item of equipment.

Muther proposed a technique whereby a layout may be derived directly from the chart. Even the simplified procedure he proposed later [Muther 1962] is too time consuming for practical application. For example, three man-days were required in one case to manually convert a 45 department chart with approximately 1000 pairs of inter-departmental relationships to a block plan layout [Lee 1967]. The smallest process plant one would expect to lay out will typically contain approximately this number of equipment items and relationships.

Nonetheless, the Relationship Chart has been used effectively to marshal data for entry into automated layout computer packages, specifically, CORELAP [Lee 1967] and the system constructed by Shuquair [1978]. Its potential value is greatest when used as an aide memoir to reduce the likelihood of a relationship being accidentally neglected.

2.4.2 The Correlation Chart

The well-known Correlation Chart is a diagrammatic method of recording the effects of constraints and the layouts that they allow. A worked example is presented elsewhere [Mecklenburgh 1985, p 478]. A grid is drawn with the rows representing possible positions of one plant item - such as floors in a building or numbered positions in an area - and the columns possible positions of another. The flowsheet is inspected and constraints on the relative positions of pairs of items are identified and assigned a reference label. If any constraint prohibits a particular position for an item, the row corresponding to that position is "struck out" with the reference label for that constraint. Vacant squares thus show permitted combinations.

The set of lines of the grid can be extended and crossed by rows and columns representing other items, and further prohibitions or preferences applied. Ultimately, the only feasible combinations are those that can be traced through the rectangular network of squares.

The possible positions of each item may also be represented symbolically, for example, A1 representing item A in position 1. The relative positions are represented by effectively algebraic expressions in which the addition operator is taken to mean alternative positions and the multiplication operator taken to mean co-existence. Thus A1 (B1 + B2) means item A in position 1 with item B in either position 1 or 2. Any impermissible relative positioning is represented by such an expression set to 0. The expressions are multiplied out and the remaining terms represent permutations which are acceptable.

The effects of constraints is immediately visible when presented on the chart, although comprehension is rapidly lost as the number of items increases. Larger problems may be represented more effectively using the algebraic representation, especially where a number of permutations is permissible for a layout. The layout is difficult to visualize from the algebraic representation, however, even for simple designs.

This technique may only be applied if the possible positions may be quantised. It may only be used to assign equipment to floors of a building or to derive plan positions for items which are all roughly a similar size and in similar restricted cases. Where applicable, it can give powerful guidance on the relative positions of albeit relatively few items of equipment.

2.4.3 The Travel Chart

Travel Charts [Wild 1972] were originally used for finding the optimal ordering of machines in jobbing shops, but it is claimed that a process unit layout can be derived using this technique. A chart is set up in squares. Both horizontally and vertically, the number of squares used is equal to the number of items to be positioned. The items are listed across the top and down one side in the same, initially arbitrary, order. A diagonal line is drawn across the chart.

The total magnitude per unit distance of the relationships between any pair of objects is entered in the square that is the meeting point of the row representing one object and the column representing the other. Wild suggested the use of the total cost of material transport between the objects as defining the relationships' magnitude. Mecklenburgh [1985] suggested that this could be extended to include subjective weightings of other constraints.

The position of the square in the chart holding the relationship reflects the proximity of the related items if a layout were constructed from it. In the case of neighbouring items, the relationship lies next to the leading diagonal, for items which are separated, the relationship lies far from it. The "cost" of a separation of two items in the layout is found by multiplying the value in the square by its distance from the diagonal. Thus, a lower total "cost" for the layout is achieved by permuting the columns to concentrate those relationships with high magnitude per unit distance close to the diagonal. Inspection of the table shows which permutations may be beneficial. The process is repeated until no further reduction in cost can be achieved. The order of the columns then defines the optimal linear solution for the layout, this technique being incapable of deriving non-linear solutions.

The method can be modified, in part, to compensate for different object sizes. In addition, groups of items which must be close can be treated as a single item.

Mecklenburgh [1985] states that the technique manifests the relative importance of having different pairs of items close to each other, a useful first step in the two- or three-dimensional problem. Nonetheless, he generally dismisses the technique, arguing that a one-dimensional arrangement of equipment items is rarely desired in process plant layout.

2.5 Layout Visualisation Packages

The unsuitability of early CAD systems to conceptual layout motivated the development of dedicated computer systems. The interactive graphical interface absent from the contemporary commercial CAD packages were central features of these systems.

The first system constructed to meet the specific needs of the layout engineer [Fine 1965] exhibited another feature typical of these dedicated systems - provision of constant feedback to the user on the "quality" of his design. Each item of equipment of the plant is approximated by a line, whose end-points are user defined. Nozzles are assumed to lie along this line, a stated distance from one end (the "reference end"). Tees, reducers, *etc* are assumed to be points, although with nozzles on them. Objects, including pipe fittings, are positioned manually on a VDU representation of the plot, allowing the absolute nozzle positions to be calculated. The length of a pipe is approximated by the sum of the moduli of the differences of the X, Y and Z coordinates of the nozzles at each end unless the pipe runs outside a box enclosing both. In this case, the pipe is assumed to be run via a pipe rack and its length is increased by twice the length from the rack to the box.

The system was extended to allow comparison of a number of layouts, and to incorporate interference detection. It allows the production of piping cost estimates much sooner than by manual means.

String may be used to represent pipework in conjunction with physical layout analogues. This has been observed to improve the resulting design. This was emulated in an interactive computer package [Bush 1972]. A central pipe rack is displayed on the screen, about which the plant or section thereof is laid out. The user selects a symbol representing the plan view of an item of equipment, and positions it. He then selects another item, positions it, and so on. As soon as both objects at the ends of a pipe are positioned, the pipe is automatically drawn in, and its cost added to the current total. It was intended that this cost data be used in selecting the initial positions of equipment, and to guide redesign and tuning of the layout. To allow this, the user is free to re-position equipment and re-route pipes at any time. Each item is surrounded by an access area. The program will not permit the infringement of this. The program ultimately outputs the absolute coordinates of all items.

A similar system [Leesley 1972] builds a model of the plant within a hierarchical database, although without the piping cost estimator. Pipe runs can be sketched in, and clash checked against both equipment and other pipes. Many of the concepts developed during this seminal work were later employed in "PDMS", an industry standard commercial piping design package. The sketching facility was not required for the intended role of PDMS and this major attraction of its precursor to the layout designer was omitted.

Shuquair constructed an advanced system intended originally to be a component of PDMS to redress this omission [Shuquair 1978]. The system was also intended to extend PDMS until it met many of the needs of conceptual layout designers. His objectives were

> "to construct a software package which would provide an easy way of discovering and analyzing the relationships existing between plant items to determine their relative closeness, to provide a rational way of conducting the plant layout task, and provide the user with quantitative feedback on the quality of the layout produced".

In actuality, he was forced to write a separate system, interfaced to PDMS, due to the inability of the PDMS data description language to represent those attributes of relevance to the plant layout task.

Relationships between plant items are identified manually, and the code number of the class to which the most important relationship between any pair is entered into the system on an activity chart, after Muther [1961]. These code numbers range from 1, for a relationship which must be satisfied if the plant is to operate, to 5, for an environmental constraint. Items which would pose a hazard if grouped are ascribed a relationship of code -1.

Activity groups are formed by the system from all objects which are related, irrespective of the magnitude of the relationship. All activity groups are checked to ensure that they contain no pairs of objects which have a repelling safety relationship between them. If any are found, an error message is output and the program stops. The activity groups may then be manually modified if they are either too large or trivially small, or if the relationships between their members are such that it is not meaningful for them to occupy the same group. For example, if a unit operation contains a heat exchanger, all other heat exchangers in the plant will also be added, as they have a relationship to the heat exchanger which should be present. The plant blocks formed by this manual intervention are then input to the plan layout programs. Each block is laid out in turn. The temporal sequence in which the members of the block should be positioned is determined by the system. A reference item is chosen for the block. Typically, this is the item most strongly attracted to the central pipe rack, defined by the user. The item most strongly related to it is next in the sequence, and so on. The actual plan positions are decided manually. A utility is provided that allows the user to define the position of an object in terms of a vector from the previously located item to specify the clearance required between the two items. This frees the user from the tedium of calculating absolute coordinates. The resulting layout is then automatically checked for clashes. The completed plant blocks are then ordered along the pipe rack by a permutation algorithm.

The system is a very powerful aid to the layout engineer. It functions as a sophisticated data recording and manipulation tool. It imposes rigorous structure on the design process, increasing the reliability of the results and frees the user from many mundane tasks, improving productivity. While the program does have limitations, some severe (*eg* it cannot manipulate elevation data), it is capable of producing results which are limited only by the skill of the user.

These systems represented state of the art computing at the time of their construction (1965 to 1978) and they successfully met their designers' objectives. At the time of their development, the principal benefit of CAD was perceived to be its scope for increasing productivity. This tended to focus its application into the project engineering disciplines. These consume the majority of project man-hours and by implication, present the most scope for significant savings. The conceptual layout systems were intended for the front-end engineers, not then considered valid users of CAD technology. I believe that this explains why none was commercialised in its original form.

As we have seen above, commercial general purpose CAD packages have now developed to a level where they can be employed in conceptual layout. It is unlikely in my opinion that new dedicated systems for visualisation alone will be developed.

2.6 Automating Process Plant layout

The computer systems described above perform useful functions for the conceptual layout designer. They assist the designer by presenting a visual representation of the layout as it evolves. Parametric equipment models may perform the mundane tasks, freeing the designer to concentrate on the creative aspects of his work. The dedicated conceptual layout packages generally provide feedback to assist the designer to assess a layout proposal.

Systems capable of automatically generating a layout are the natural progression. Such a system could provide a three-dimensional realisation of a process proposal. The process engineer could then gauge its feasibility or derive more accurate estimates without increased work load. The system could also be employed to generate rapidly candidate final layouts. These could be used by the layout designer as starting points in the iteration or to explore rapidly alternative solutions. During the period in which the visualisation tools were constructed, parallel work addressed this goal.

Gavett and Plyter [1966] suggested "branch and bound" optimisation may be suitable for automating space planning. This technique proceeds in a cycle. Two or more partial solutions are generated from the existing solution state, "branching" the solution. A "bounding function" is evaluated on each partial solution thus generated. This is a function that yields either an upper or lower bound to the objective function for any solution following from that partial solution. One knows that, in pursuing a given branch, the optimal solution for that branch will score at least a minimum or at most a maximum (depending on whether the bounding function evaluates minima or maxima). The branch which offers the lowest minimum or highest maximum as appropriate is selected and the procedure is repeated.

The branch and bound technique has been applied to the layout of chemical process plant [Mustacchi 1974]. This system orders the equipment comprising a process unit in a single line along a pipe rack. The branching at any point is generated by constructing two new partial solutions, one in which the object under consideration is to the left of another specified object, and one in which it is to the right. The bounding function determines the minimum cost of the pipework that may possibly be achieved in that branch. If the relative positioning of objects A and B is currently being considered, and placing object A to the left of object B yields a lower value of possible pipework cost than placing object A to the right of object B, the branch in which A is to the left of B is chosen for further analysis.

This system is of little practical application because it considers only one variable in the objective function and is limited to developing single line solutions, rarely desired.

A later system [Al-Asadi and Gunn 1980] was constructed to lay out plant within buildings, using a more realistic objective function. The impression is gained that this system is intended to produce better layouts than can be manually achieved, rather than to speed the design process.

The plant is manually decomposed into modules, each comprising one or more items of equipment which form a natural group. Each module is laid out manually, and then fully enclosed within a cuboid, together with any required access and maintenance space. Where a single cuboid represents a module wastefully, cuboidal sub-modules are assembled with their positions fixed relative to one another. The dimensions of each module or sub-module are entered as initial data. The positions of the starting and finishing nozzles of those pipes running between modules, and the cost per unit length of such pipes, are also defined in the input data.

The initial positions of each module is chosen manually and input. Gunn [1970] previously suggested applying Rosenbrock's search method to layout problems. This is employed in this system to manoeuvre the modules within the basic arrangement to achieve an optimal layout. The objective function is plant cost, comprising the sum of inter-module piping costs and the building cost (volume multiplied by cost per unit volume, or area multiplied by cost per unit area). A number of basic arrangements are input and optimised to yield the overall optimum layout.

Auxiliary modules representing control rooms, electrical switch rooms, *etc* may be included for their nuisance effect in occupying volume. Piping modules may be used to represent pipe racks to find their optimal positions.

The system was demonstrated laying out two polymer plants and produced ostensibly good results in both cases. Indeed, the system appears to be an effective tool to support the experienced layout engineer seeking to highly optimise a layout proposal. The extensive manual pre-processing and the unnatural work style required to use this system probably discourages its wide acceptance, however.

2.7 Automating Space Planning

Two systems intended specifically for process plant layout have been described. A wide range of systems have been constructed for application to other layout tasks. These are reviewed in this section.

2.7.1 The Quadratic Assignment Problem

The Quadratic Assignment Problem or QAP [Hillier 1966] is a numerical formulation widely employed in automated space planning systems. The QAP may be defined as follows.

Each entity to be positioned is referred to as a "facility". If the facilities occupy different areas, each is divided into a number of "sub-facilities" all with equal area. The planar site is then divided into a rectangular grid of elemental areas called "locations". The QAP assigns the sub-facilities to locations guaranteeing that all sub-facilities of a facility are adjacent in a convenient configuration and that materials handling costs are minimised.

A large number of systems implementing the QAP have been constructed. These differ only in detail from the seminal systems. The review papers of Moore [1974] and Foulds [1983] discuss over thirty examples between them. QAP programs use a simple grid of squares in which the detection of overlap is easy but resolution is limited by the grid size. Facilities of different sizes are represented by an integral number of sub-facilities. These can be forced to be adjacent but no shape can be imposed for the facility. The programs tend to generate solutions in which each facility has an awkward shape and which need extensive manual improvement. The shape of the solution as a whole is also difficult to control and the solution tends to sprawl. This behaviour is at odds to that required of a system for automating conceptual process plant layout. Here, the individual elements are (effectively) fixed in size and shape and the overall layout is highly structured around pipe racks, on-plant roads, *etc.*

It is instructive, nonetheless, to consider the algorithms adopted to solve the QAP as they embody some features pertinent to any space planning system. The programs can be divided into two broad categories - "additive" and "permutational" systems. In an additive system, a sub-facility is assigned a position based on attractions to others already positioned. Thereafter, its position remains fixed. In a permutational system, sub-facilities are initially randomly positioned. Their positions are then permuted until no further improvement can be made with reasonable computational effort.

Important examples of additive systems include CORELAP [Lee 1967], STUNI [Willoughby 1975] and the system of Whitehead and Eldars [1964]. The former is intended to generate factory layouts and optimizes on minimum transport of goods. The latter generate building layouts and attempt to minimise the travelling time of people and land usage. CRAFT [Armour 1964] is one of the most successful of the permutational systems. It has been enhanced [Lew 1968] and also modified to generate three-dimensional layouts in CRAFT-3D [Cinar 1968].

It has been demonstrated that permutational systems can approach the lower theoretical bounds of minimum cost [Francis 1974]. No additive system has yet been demonstrated performing as well as a human designer. In an additive system, the choice of position for a sub-facility can only be influenced by its relationships to the others already added to the plan. A sub-facility with relatively minor constraints on its position will be placed late in the procedure. These minor constraints should be allowed to distort the overall plan, but are neglected until after it is possible to satisfy them opportunistically.

In STUNI, elements with relationships to fixed site features are added first. The others are then added in descending order of attraction to those already positioned. If no position can be found for an element satisfying its relationships, the program will fail. The program is then re-run with that element at the head of the addition sequence. STUNI generates better layouts than the single-pass additive systems.

This shows that space planners must either hold decision-making in abeyance until sufficient information becomes available to guarantee correctness or embody a back-tracking mechanism to allow them to revise early, perhaps premature, decisions. These are two important problem solving techniques in Artificial Intelligence. The former is often referred to as "least commitment reasoning" and the latter is typical of systems that achieve a solution by search.

The space planners based on the QAP also demonstrate the efficacy of heuristics in algorithms to solve complex problems. It has been shown that the number of elementary computational steps required by any QAP algorithm will be an exponent of problem size - the QAP is so-called "NP-complete" [Sahni 1976]. All practical permutational systems constructed to date embody heuristics. This is of no relevance to additive systems which do not search for a solution. Alternatively, it could be said that additive systems do not consider significant sections of the search space because of their inability to back-track.

ALDEP [Seehof 1967] is a permutational system constructed to compare three heuristics. Layouts were randomly generated in a single pass then scored, with the best being used as the solution. In the best interchange strategy, every possible pairwise interchange of entities is generated and the interchange which most improves the layout is retained as the starting point for the next iteration. This strategy is also employed in CRAFT. In natural selection, the best interchange is selected, but only made if this improves the overall score. Natural selection is found to be both slightly faster and capable of generating slightly better solutions than best interchange. Unsurprisingly, random generation is found to be almost two orders of magnitude faster, but even its best layouts are of very low quality.

Liggett employed an alternative approach to construct a system with the speed of an additive system but the quality of results of the permutational system [Liggett 1972]. Her additive algorithm incorporates a probabilistic look-ahead mechanism which attempts to predict where allowance should be made for elements to be added later. The resulting layout is then polished up by permutation. This system generates solutions that are marginally superior to those produced manually by a skilled practitioner. (The machine generated layout satisfied 23 out of 48 desired adjacencies, as compared with 21 satisfied in the manual solution).

2.7.2 Simulated Annealing

Recent work uses an analogue of the annealing process as a space planning technique [Sharpe 1985]. As with the QAP formulations, control of the overall shape and structure of the solution is difficult to impose and each facility tends to be awkwardly shaped. Nonetheless, simulated annealing has become fashionable amongst AI workers for solving other problems so its application to layout is worthy of discussion.

In the annealing process, a bulk of material is heated to high temperature then slowly cooled to remove flaws in its lattice structure. The atoms vibrate and move to positions in the lattice where they have lower energy states. The amplitude of the vibrations fall with falling temperature, reducing the distance the atoms may move. Ordered crystals condense from the initially disordered atomic structure.

This is simulated by an algorithm devised by Metropolis *et al* [1953]. In each step, a pair of atoms is chosen based on their separation exceeding a set minimum. The positions of the atoms are interchanged and the change in total system energy is calculated. If the energy remains equal or falls, the interchange is accepted. The next interchange is applied to the new configuration. If the total energy increases, the energy change arising from the interchange of a "typical" pair of atoms of the specified separation is calculated. If the observed increase is less this average, then the interchange is still accepted.

When a specified number of consecutive interchanges do not cause the total energy to fall sufficiently, it is assumed that all pairs of atoms which may be interchanged to reduce energy have been identified. The set minimum separation of atoms is reduced and the procedure repeated. This emulates the effect of falling temperature in the physical process.

This algorithm is applied to building layout by simply replacing the energy by cost and atoms by activity modules. The modules are positioned in the building initially either at random or as specified by the designer. The modules migrate large distances initially while the "temperature" is high. They find the region of space in which their optimum position is likely to be found. As the "temperature" decreases, they cannot jump out of the optimum region but search locally for the optimum position. Nonetheless, at any temperature, they can escape from local optima which are globally sub-optimal because the algorithm allows increase in cost in a controlled manner.

2.7.3 The Dual Graph Formulation

Levin [1964] argued that a representation must simultaneously and homogeneously represent both the requirements and the solution to be useful in the solution of design problems. That is, the representation may contain information about the design requirements only in terms of the forms that are to satisfy them. Levin propounded the "dual graph" approach to floor plan layout, which meets this requirement.

In this context, "graph" means a node and arc diagram rather than a pictorial representation of a numerical relationship. It is customary to refer to the arcs as "edges" and the areas enclosed by edges as "regions". Assume one node is placed in each region of a graph and an edge is added to connect any of these additional nodes that lie in adjacent regions of the original graph. A second graph is formed by this process. This is the "dual" of the original.

Levin was inspired by a philosophical argument but he and other workers developed a successful practical technique from this basis. The technique is designed to be used to develop internal layouts of buildings for human occupancy. It is principally concerned with satisfying the space requirements of the rooms of the building by apportioning the available space between them. Constraints may be imposed on the relative positions of the rooms although this is practically restricted to requirements of adjacency and direct communication (for example, specifying that a door must exist between two adjacent rooms). Physical dimensions of the elements may also be controlled to some extent by specifying the length of the interface between them.

The technique is of limited applicability in conceptual process plant layout because of its inability to consider positional relationships other than adjacency. In plant layout, it is often sufficient to position two elements in reasonable proximity (for example, to reduce the length of piping between them). Requirements for proximity cannot be interpreted as demanding adjacency and genuine adjacency requirements comprise a small proportion of the total relationships that should be considered. The technique is quite simply unnatural for plant layout.

Nonetheless, some description of the technique is called for given its success in solving other layout problems. Graph duality is reflexive. That is, if graph A is formed as the dual of graph B, then given graph A, graph B could be formed as its dual. The objective in this technique is to form a graph representing a floor plan. The nodes of the graph represent corners of walls, the edges represent wall segments, and the regions represent spaces. A pair of contiguous spaces will be represented on this graph as a pair of contiguous regions, separated by an edge.

Clearly, the edges of the dual of the floor plan graph represent adjacencies between the rooms. Because graph duality is reflexive, we can construct the adjacency graph from the floor plan graph, or vice versa. The latter is the more useful practically as the adjacency relationships are explicitly available at the outset of space planning. We construct a graph in which the nodes represent spaces which are required to be adjacent and the edges represent the adjacency constraints themselves. By exhaustively enumerating all possible floor plan graphs from this adjacency graph, all configurations that satisfy the constraints are generated.

In practice, exhaustive enumeration is not feasible in problems which include a realistic number of elements and relationships. Once again, it is seen that a theoretically attractive concept may only be implemented if heuristics are applied to control the combinatorial explosion. Indeed, the dual graph technique amply demonstrates that improving the heuristics employed increases the efficiency of the solution process.

Grason [1968] states the theoretical necessary conditions for it to be possible to generate a floor plan graph from a constraint graph. The constraint graph must be planar, that is, it must be possible to draw it without any edges crossing. Each node in the constraint graph may have a maximum of four edges connected to it and each edge must be directed along a different orthogonal axis. This is a necessary condition for the sub-spaces to be rectangular, a precondition of this technique being applicable. Grason also states certain topologies of chains and circuits which must be absent from the constraint graph.

"GRAMPA" [Grason 1971] employs these necessary conditions as its heuristics. All adjacencies required in the layout are entered. An adjacency is selected to be added to the adjacency graph as an edge. The edge is added in all positions to construct all possible expansions of the graph. Each new graph is checked for feasibility according to the conditions stated above. If feasible, it is retained and developed further by the addition of a new edge, and so on. If infeasible, it is abandoned. Once all adjacencies have been added, each surviving constraint graph represents a potential solution and is converted into a floor plan. A similar technique was implemented (independently) by Yessios [1972].

Pereira [1978] incorporated more sophisticated heuristics into his system. These include conditions that determine whether the graph will lead to a solution that will fit into a rectangular contour. The resulting layouts are more likely to be useful to an architect, the system's intended user. Note that these heuristics embody much higher level knowledge than the graph theoretic heuristics employed in GRAMPA. They discount far more alternatives without rejecting any that may lead to an acceptable solution. Because far more alternatives are rejected, this system produces layouts more efficiently than GRAMPA.

Grason [1971] also suggested that the problem be decomposed into more but simpler sub-problems by representing groups of entities as single nodes as abstractions of the problem. These abstracted nodes would be relatively positioned, then each replaced by nodes representing its constituent elements. The technique would be applied recursively until the nodes representing the individual elements were positioned. This concept offers an alternative (perhaps complementary) to problem reduction by heuristics. The concept was never incorporated into GRAMPA but was adopted successfully by Gilleard [1978].

The space planning systems based around the QAP inherently optimise the layout against an objective function, usually the minimisation of material or personnel movement. Krejcirik [1969] constructed a system based on the dual graph technique to position departments in a manufacturing plant capable of designing the optimal solution. The edges of the constraint graph represented material transfer between the departments. Weights were assigned to the edges according to the effort needed to transfer materials. These values are then used to extract a planar graph of maximum edge weight. Note that constraints were either fully satisfied or neglected in this approach. This definition of optimal implied by this is at least consistent with the dual graph formulation focusing on adjacency relationships.

2.7.4 The Transformation Grid.

The "transformation grid" [Steadman 1970] offers an alternative technique for dissecting the space. It is also intended to position orthogonal elements, although the restriction that they be rectangular does not hold. Notionally, horizontal and vertical lines are drawn across the representation of each element such that all of its vertices lie on an intersection and each line of the grid passes through at least one vertex. The separation of the lines in this minimum grating are adjusted so that each cell becomes a square. This yields a dimensionless generic representation of any entity of the same topology.

The fully developed form of the technique to manipulate these representations [Mitchell 1976] is described here briefly. The total area to be filled is represented by an array. All topologically distinct dissections of the space are generated by rules of three forms. A new row of cells may be added to the array. An existing sub-space may be dissected into two spaces, providing it occupies more than one cell of the array. The third and more complex rule may be applied when two or more rectangles border an outside edge of the space. A new row of cells is added to the outside of the array parallel to the edge. If there are R rectangles bordering the edge initially, the first R-1 rectangles bordering the edge are extended upwards into the new row. The remainder of the row is filled with a new rectangle.

Rules of each form are applied to all existing partial solutions to generate a range of more developed partial solutions. Each of these is retained except those isomorphic with a previous dissection which are discounted. The rules have been shown empirically to exhaustively generate all distinct dissections for up to six component elements.

Until this time, the solution generated is an abstraction. Once all distinct dissections are generated, each is checked to determine whether it represents a physically feasible solution. A graph of required adjacencies is superimposed on the dissected space. Each node in the graph represents one element and is positioned over one sub-space. If the graph can be mapped onto the dissection topologically, the dimensions of each component of an element are used to dimension the squares of the sub-space beneath the node. If the dimensions assigned to the array elements are consistent with those of neighbouring elements, a feasible solution is indicated. Constraints may also be considered in this assignment process, such as certain entities requiring external walls.

A similar system [Earl 1980] represents the incidence relations between line segments and faces by using complex rules which replace existing walls by hypothesizing the position of wall junctions on the grid. This is the most general of the dissection systems, and probably represents the ultimate limit of the paradigm.

A given dissection may be transformed into the subsequent dissection by a single recursive re-write rule [Flemming 1978]. This generates triplets which represent one wall each. The first element of the triplet records whether the wall runs horizontally or vertically. The second element lists the sub-spaces above (or to the left of) the wall. The third lists the sub-spaces below (or to the right of) it. The rule inserts new triplets into the existing set, maintaining the order of unaffected spaces. The system's builder concludes, however, that the single dissection rule is too limited and that his approach is the weakest of the three.

Perhaps the most widely cited example of this formulation [Coyne 1985, Gero 1987] was most limited in both scope and effectiveness! The designers' intention was to demonstrate knowledge-based planning as an effective model of design activity. Space planning is only one of a number of domains in which they built exemplars.

It is important to note that these dissection formulations exhaustively form all topologically feasible dissections then assess each to determine whether it represents a physically feasible solution. Flemming [1978] states that "it appears difficult, if not impossible, to determine both the geometry of a solution and the dimensions of the spaces ... in one integrated process."

All results from workers applying the dissection formulation concur with this view.

The transformation grid may manipulate elements which are not rectangular. This removes a significant disadvantage of the dual graph technique. Nonetheless, the transformation grid remains unattractive for process plant layout for effectively the same reasons - the elements must fill the area, no credit may be gained for their reasonable proximity and again, the un-naturalness.

2.7.5 Symbolic Constraint Representations

All systems discussed so far divide a space into regions of known area but fluid in shape and dimensions. Far fewer systems have been constructed which solve the problem germane to conceptual process plant layout and directly determine the optimal position of items of fixed dimensions. It is noteworthy that all extant systems of this latter type use symbolic rather than numerical representations and fairly loosely structured algorithms.

General Space Planner or GSP [Eastman 1972, Eastman 1973] recorded the constraints on the position of an entity as binary spatial relations of the form "is adjacent to", "is in sight of", and so on. Design Problem Solver or DPS [Pfefferkorn 1975] recognises these and other relationships. These may specify distance between elements (either upper or lower bounds), absolute position (such as, the element should be in the left half of the room) orientation, access area requirements, sight lines and paths to be taken to reach any element.

The benefit of the symbolic representation of the relationships is readily apparent from these examples. They express a much greater richness and wider range of relationship than any of the techniques discussed previously can handle. This expressiveness allows the symbolic representation to match the sophistication of "real-world" layout problems. The symbolic systems are epistemologically appropriate to the problems to which they are applied, the numerical systems are not.

Both GSP and DPS develop their solutions using a "generate and test" architecture. In this, a new partial solution is generated from an existing partial solution. The new partial solution is then tested for compliance with imposed constraints and retained or abandoned as appropriate. In a space planning context, a partial solution constitutes a layout in which some but not all elements have been positioned. Both of these systems employ heuristics to direct the generator into creating an alternative most likely to succeed if it could generate more than one new partial solution. This feature is common to many of the systems described so far.

DPS also attempts to decompose the problem into independent sub-problems. The system examines the constraints and identifies those that will be easily satisfied. This breaks the constraint network into a number of fragments. These fragments are laid out separately as macro objects. This heuristic reduced the layout of a large computer from 9 objects and 49 constraints to 4 objects and 17 constraints, for example. The reader is reminded that this approach was also successful in Gilleard's dual graph system described above.

In GSP, the elements are sequentially positioned by so-called "location operators". A location operator may position an element by projecting the edges of a space and aligning the element with an edge; an operator may position an element at a location along the boundary of another or an operator may position an element such that a point on that element lies a fixed distance from a point on another. In DPS, an element may only be added located at a corner of the available space. This may be either a room corner or a corner created by other objects. Pfefferkorn admits that this is overly restrictive in many circumstances, such as problems requiring small objects to be placed in the centre of a room.

Note that neither system chooses locations using the information embodied in the constraints. The generator chooses a position almost arbitrarily. The tester uses the constraints to assess the suitability of the chosen position. The generator absorbs time creating partial solutions for the tester to absorb time rejecting them immediately. This inefficiency is rectified in the more recent IMAGE [Johnson 1970] and MAS [Masanori 1976] described below.

In GSP, relationships may only be considered if they relate an element being added to those already positioned. GSP is forced to assume that the existing elements are correctly positioned. Eastman acknowledges that this is a deficiency and concludes that the space planning process must involve iterative search procedures with back-tracking [Eastman 1973]. Note that this concurs with the results observed with additive QAP systems. The problem can now be categorically stated to arise not from the use of an inappropriate numerical technique but from the lack of back-tracking.

DPS incorporates not only back-tracking but a sophisticated diagnostic implementation. The design progresses using the depth-first method until a major difficulty is encountered. Then, rather than backing up the tree of partial solutions, the system attempts to diagnose the cause of the difficulty. The diagnostic process is simple - the constraints that were broken when the last set of alternatives were generated are identified. This information is then used to select a remedial action.

Four simple remedial actions are implemented in DPS. A macro object can be created from the objects related by the broken constraint. DPS builds macro objects by positioning their component objects relative to one another so as to satisfy their inter-relationships. The macro objects are then represented identically to any other object in the problem. DPS may modify an existing macro object comprising the objects causing the difficulties if one has been created in an earlier alternative. The object causing the problem can be entered earlier in the solution. The system identifies the cause of the difficulty and positions the object while sufficient freedom still exists. As a last resort, DPS reverts to depth-first search in a systematic examination of all possible alternatives.

The combination of diagnosis and remedial action also greatly reduces the search. In a particular example problem, this approach examined 125 partial solutions before finding the solution. In comparison, the depth-first approach (implemented for comparison) examined 687. It should be noted that the implementation of diagnostic search is supported by records of all partial solutions generated.

CADOO [Marcoussis 1986] is a commercial system structured similarly to GSP and DPS. This is indicative of the efficacy and appropriateness of the techniques described above. CADOO lays out the units comprising a ship's propulsion system. It applies a knowledge base of generic constraints to a description of the propulsion system and ascribes a severity degree to any non-imperative constraint. The constraints represent requirements of proximity, segregation, similarity of orientation, similarity of axis and contact between the units. The system then enters a cycle of planning and placement. A unit is chosen to be positioned by production rules expressing criteria such as "biggest unit first". Potential sites for the unit are then identified for the unit by further rules. A technique similar to that in DPS is employed to handle difficulties in placing later units. The possible sites for the unit are evaluated and sorted into an order of increasing compromise amongst constraints. The best site is selected and the unit is placed. The better alternatives are also noted. In case of failure, a selective back-tracking system assesses whether a unit may be re-positioned at one of its alternative sites. The system positions elements in two dimensions, and is intended to produce a preliminary design for manual refinement using traditional CAD tools.

2.7.6 Increasing Sophistication

The IMAGE system [Johnson 1970] and its descendant, MAS [Masanori 1976] are constraint driven permutational systems. In these systems, the constraints may specify both a separation distance and a numerical weighting. An initial arbitrary arrangement of the elements is made. Each element is tested for unsatisfied relationships. Elements with unsatisfied relationships are re-positioned.

Each relationship carries the predicate by which its satisfaction is assessed upon it. Similarly, the operators to select the new positions for elements are also linked to the instances of the relationships. A very rich description of the relationships may thereby be expressed to these systems. This allows them to consider realistically sophisticated relationships in their problem solving. As such, these systems are major advances on those described in the previous section.

Unfortunately, both systems were found to usually require significant assistance from their operator. The richness of the constraints exceeded the solution capacity of a permutational algorithm.

Previous workers had already constructed systems which considered highly sophisticated constraints. They circumvented the lack of a suitable algorithm by dispensing with automatic layout generation. SEARCH [Bryant 1977] concentrated entirely on the evaluation of layouts designed by a human. PACE1 [Maver 1971], URBAN5 [Negroponte 1968] and Newman's experimental system [Newman 1966] also acted in interactive mode only, with no solution algorithm.

In IMAGE and MAS, if no satisfactory location could be found for an element, all relationships acting on it were partially satisfied by placing it at a least squares weighted average location. This approach breaks down where the relative positions of elements that satisfy one relationship do not intersect with those required by another relationship. In these cases, the more important relationship must be satisfied fully at the expense of the less important relationship. The dual graph system constructed by Krejcirik [1969] described above worked in exactly this manner. It was incapable, however, of partially satisfying all constraints when this would give rise to the globally optimal solution. Some constraints encountered in plant layout must be satisfied fully to gain any benefit, others can be partially relaxed. It is unacceptable to adopt one strategy for resolving conflict at the expense of the other.

2.7.7 Least Commitment Problem Solving

The protocol analysis of space planning presented in Section 2.2 shows that the human designer adopts a markedly different strategy to the algorithms employed in the automated space planning systems described so far. The human progresses by generating partial solutions then integrating them. The automated systems attempt to solve the problem as a whole.

The partial solutions generated by the human may be either complete solutions to localised problems or solutions to the problem as a whole which lack detail. The designer focuses attention on the area of the problem where most is currently known while avoiding drawing inferences based on partial information if possible.

The behaviour of making highly constrained decisions first may be referred to as "opportunistic" problem solving. It has been employed in the speech understanding system HEARSAY-II [Hayes-Roth 1977] for example. In HEARSAY, constraints on possible meanings are scheduled to generate solutions efficiently by applying the constraints with the least uncertainty associated first. This forces the search for the solution through areas of certainty in the search space. Seminal examples of systems that avoid inferences based on partial information are MOLGEN [Stefik 1981a, 1981b] and ARS [Stallman 1977]. This technique is referred to as "least commitment" problem solving and increases efficiency of search by avoiding the increased work of deriving detail when the general structure of the solution is still fluid and uncertain.

The WRIGHT system [Baykan 1986] employs both opportunistic and least commitment problem solving in space planning. The published example of its application is in the domain of kitchen layout. In general the location of work areas (such as a preparation area) is selected first and the positions of the appliances (perhaps a food mixer) may only be selected from within the area.

This is conventional top-down design. However, if the position of an appliance is highly constrained, (a sink is limited to being under a window) the appliance is positioned and the work area (in this case, a washing area) is designed around it. It is claimed that this opportunistic problem solving "efficiently eliminates many dead ends and leads search towards the better solutions".

WRIGHT employs least commitment reasoning by identifying the relative positions of problem elements, expressed symbolically, before calculating the absolute positions. In the current version of WRIGHT, the former task is completed before the latter computationally very expensive task is attempted. A forthcoming version will integrate the reasoning at the two levels of abstraction into one overall opportunistic control structure.

2.8 Current Work

No significant work to develop computer systems either to support the conceptual layout engineer or to automate space planning is known to have been reported more recently than the cited references. Nonetheless, knowledge based design support remains an active research area. Recent examples of process industry applications include knowledge based process synthesis [*eg* Banares-Alcantara 1994, Sileti 1993] and piping design [*eg* Jain *et al* 1992].

Chapter 3 : Experimentation and Results

There are compelling arguments that the best computer tool for the conceptual layout engineer is one which pro-actively participates in the design work. For example, the system developed during this work, called "Plant Layout System" or "PLS", generates three dimensional layouts from process data without its user needing to intervene or assist. An example of this is shown below. Many of the benefits that the results of this work offer only arise from a fully automated system. These benefits are discussed later in Section 10.1.

A computer system will only be able to generate layouts automatically if it embodies substantial quantities of the knowledge of expert layout engineers. In principle, this knowledge could have been elicited and encoded into a conventional procedural program. However, there is a powerful practical argument that success is most likely if a system to automate layout is implemented as an Expert System.

The control structure or "inference engine" is separated from the domain knowledge in any Expert System. The domain expert from whom the knowledge is elicited need only concern himself with domain facts and procedures which remain explicit in the knowledge base. In particular, the knowledge in an Expert System can be inspected and amended readily during its lifetime. This is crucial for a computer system intended for use in conceptual layout.

Firstly, the knowledge base of an Expert System can be constructed incrementally. Initially, limited but realistic functionality can be incorporated. It can then be expanded to meet new and more ambitious targets. At any time, the system can be run and compared against a standard to highlight omissions and inconsistencies. The full scope of the knowledge and procedures used in layout cannot be established in a single pass. Such a task would be simply too ambitious to be economically feasible and would be likely to fail in any case simply due to oversight. Incremental construction would seem essential in this domain.

Secondly, once a knowledge base is "complete", it may be maintained and kept current. It may be kept in line with changes in the types of problems being set to the system or advances and changes in the knowledge to be applied. Plant layout is continuously evolving. New process and plant technologies evolve, the commercial importance of products change and new legislation is enacted. This causes changes in the plants to be laid out, the techniques appropriate in laying them out, and the criteria by which the layouts are judged. The knowledge within an automated layout system must be changed commensurately. This on-going task could prove very costly within a procedural program as the whole program might need to be rebuilt. The structure inherent in an Expert System facilitates this maintenance task.

The conclusions presented in the previous paragraphs were intuitively apparent when this work was started. The results presented in this thesis substantiate these intuitions *post hoc*. They confirm the overall brief and the three specific objectives for the work stated in Section 1.1.

3.1 How PLS Was Developed

The nature of the objectives required that a solution be synthesized rather than, for example, identified by analyzing measured data. The review of related literature in Chapter 2 shows that the techniques for automating conceptual layout are based on unsound principles. The majority for automating space planning in general are inappropriate for process plant layout even though some were successful in their intended application area. The systems which used a symbolic constraint representation (see Sections 2.7.5 and onwards) offer concepts that might be of limited relevance to this work for deriving the plan layout. Of course, this only amounts to a limited aspect of a broad and diverse problem.

Thus, a fundamental strategy for laying out process plant by Expert System had to be formulated from first principles then a system had to be developed to implement this strategy.

The nature of conceptual layout was studied and the major steps in the task as currently practised were identified. These steps suggested the macroscopic components of PLS. This offered a useful starting point for the design. The components were considered individually and techniques that might be used to implement each were considered. An overall integrating philosophy and approach was also considered in parallel with the individual components. It rapidly became apparent that the manual approach is so unstructured that it could not be emulated directly. In particular, an engineer relies heavily on spatial reasoning abilities that would be considered to be unremarkable common sense in a human. It was thought that these were likely to be very difficult to implement efficiently in a computer program. Thus, the design of PLS diverged from manual practice although remaining sufficiently similar so that the users of PLS would understand the basic principles. Procedures that an engineer carries out that would be difficult if not intractable to compute in a single step were distributed amongst several abstract steps. The procedures were replaced by these abstract steps rather than just divided amongst them. These abstract steps were then re-assembled into the components that had been postulated by observing manual practice. Thus, PLS came to embody an approach to plant layout that was developed specifically for it at the detailed level and the designs of the components became tightly interwoven and mutually reliant. This procedure suggested a provisional generic problem solving approach for PLS and this approach was implemented in code. That is, the core of PLS was developed independently of any specific plant. Great care was taken to develop a problem solving procedure that would be capable of laying out any process set to the system.

Thereafter, one process was chosen as the test case for all trial runs of PLS[1]. This process is described in Section 3.2 and was carefully chosen to be representative of conceptual layout problems in general. Repeated use of one test process allowed the author to devote maximum time to developing PLS's problem solving procedure commensurate with the objectives for the original work. An initial knowledge base was constructed which contained knowledge about how to estimate equipment sizes, how to identify constraints on the positions of equipment and similar layout issues for the equipment types represented in the test process. Even though the knowledge base was built in response to the needs of this process, the knowledge was not specific to the process. Knowledge not relevant to the test process was not encoded. Nonetheless, the developed work has all the functionality of a fully expanded system and the knowledge that was encoded can be re-used for many other plants.

PLS was developed iteratively as follows. PLS was applied to the test process. Sometimes, PLS crashed or failed to terminate. If PLS completed a run, the layout was inspected for discrepancies from what would be considered reasonable from an engineering viewpoint. The fundamental cause of any fault or discrepancy was either a coding error, a deficiency in the problem solving procedure or a lack of engineering knowledge. Each problem was tracked to its fundamental cause and the error was rectified. This ensured that the problem solving procedure and the knowledge base were only modified when a fault could be definitely traced to them. In particular, great care was taken to ensure that the procedure was not modified to compensate for deficiencies in the knowledge base so the procedure remained applicable to process plant layout in general. Once the faults that were identified during a run had been rectified, the cycle was repeated.

1. This thesis frequently describes the way in which PLS responded to specific features of the test process. This does not contradict the previous statement of generality in any way. The examples show how the general features of PLS related to a specific instance of a layout problem.

PLS was developed entirely in Common LISP [Steele 1985] on a MicroVAX II computer with 16 MB of memory and 64 MB of swap space. VAXLisp was used as the development environment running under the VMS operating system.

3.2 The Test Process

A process that produces 4000 tonnes per annum of a polymer (notionally poly Vinylidene Chloride) from monomer was selected as the test case for all trial runs of PLS. It was loosely based on a process operated by a major manufacturer. The PFD for the process is shown in Figure 1.

3.2.1 Process Description

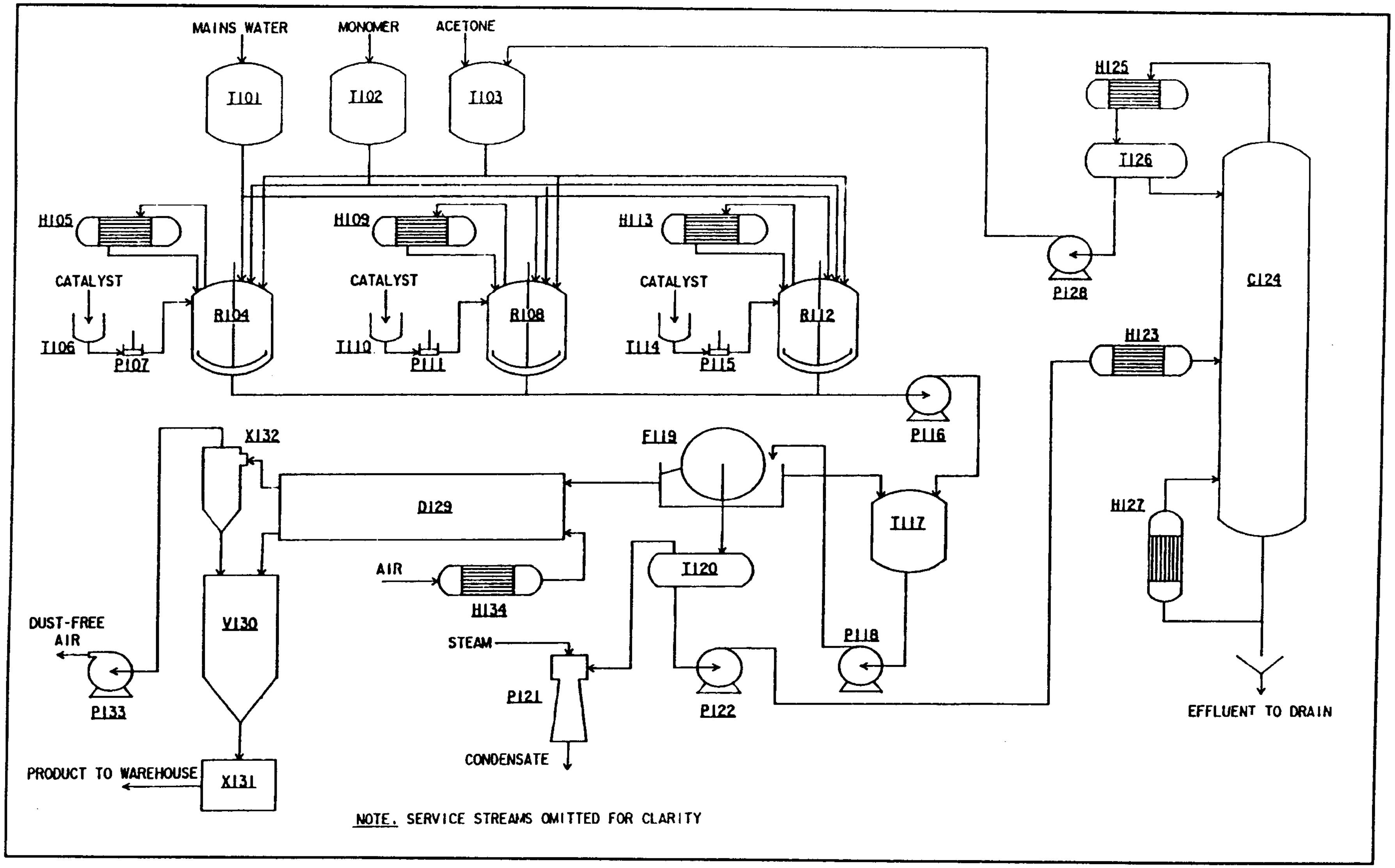
The reaction is conducted in a solution of acetone in water in the stirred tanks, R104, R108 and R112. Each batch takes eight hours to complete. The reactors are charged in turn every three hours from weighing tanks T101 (water), T102 (monomer) and T103 (make-up acetone). The reaction is highly exothermic and is cooled by the reflux condensers H105, H109 and H113. These are single pass on both sides and horizontal to increase efficiency and reduce their size.

Initiator solution is made freshly for each batch in the stirred tanks T106, T110 and T114. The initiator is metered into each vessel by pumps P107, P111 and P115 at a rate to control the reaction rate.

The reactors discharge via the progressing cavity pump P116 to the filter feed tank T117. The slurry contains 10% of both polymer and monomer and is re-circulated through the feed trough of the rotary vacuum filter F119. The level in F119 is controlled by a weir and the excess is returned to T117.

Filtrate and entrained air are separated in T120. Air is withdrawn by the ejector P121 to maintain the vacuum in the filter. The filtrate contains unused monomer, solvent and wash water. It is distilled in column C124 and the acetone and unused monomer are returned to T103. The water is discharged to drain.

Figure 1: Test Process PFD



The filter cake is dried in the rotary co-current drum drier D129. This operates at reduced pressure to minimise dust losses. The air for the drier is heated by steam in H134 and is pulled through the drier by the blower P133. The cyclone X132 protects the blower from entrained polymer.

The dried polymer is held in hopper V130 until fed to the bagger/palletiser X131. The palletised product is transferred to the warehouse by forklift truck.

The data supplied to PLS are tabulated in Tables 1 and 2. These data were deliberately restricted to process parameters. Although Table 1 shows physical sizes for the equipment, these were not input. PLS derived these by applying encoded knowledge in the same way as a layout engineer often conducts some preliminary equipment design. The initial data also included the positions of a bulk tank farm from where the acetone and monomer would be taken (North of the plot) and the product warehouse (South of the plot).

The reader should note that a proprietary monomer, rather than the notional Vinylidene Chloride, is used in the process on which the test process is based. The mass balance is consistent with the physical properties of the real monomer.

Table 1: Test Process Equipment Specifications

Item Number	Process Specification	Working Pressure	Working Temperature	Services	Dimensions (PLS derived)
T101	4.5 m^3 working volume	Atmospheric	Ambient	None	1.7 m diameter 2.15 m length
T102	3.7 m^3 working volume	Atmospheric	Ambient	None	1.7 m diameter 1.7 m length
T103	5.9 m^3 working volume	Atmospheric	Ambient	Feed acetone (from OSBL)*	1.84 m diameter 2.15 m length
R104, R108, R122	14 m^3 working volume	1 bar	30 °C	Cooling water Power	2.60 m diameter 3.00 m length
H105, H109, H113	500 m^2 heat transfer area	1 bar	30 °C	Cooling water	5.0 m tube length 1.14 m shell diameter
T106, T110, T114	0.3 m^3 working volume	Atmospheric	Ambient	Factory water Power	0.4 m diameter 0.6 m length
P107, P111, P115	4.8 l/min throughput	1 bar	Ambient	Power	0.45 m x 0.3 m x 0.15 m (each 2 off)
P116	0.6 m^3/min throughput	2 bar	30 °C	Power	1.19 m x 0.25 m x 0.38 m (2 off)

* Outside Battery Limits

Item Number	Process Specification	Working Pressure	Working Temperature	Services	Dimensions (PLS derived)
T117	14 m^3 working volume	Atmospheric	Ambient	Power	2.3 m diameter 3.23 m length
P118	3.04 l/s throughput	2 bar	Ambient	Power	1.19 m x 0.25 m x 0.38 m (2 off)
F119	96 m^2 cloth area	0.23 bar (drum)	Ambient	Power Factory water	2.0 m x 1.8 m x 1.8 m
T120	-	0.23 bar	Ambient	-	0.7 m diamter 3.0 m length
P121	220 m^3/s throughput	4 bar (steam)	140 °C	LP Steam	0.25 m diameter 1.2 m length
P122	3.16 l/s throughput	2 bar	Ambient	Power	1.0 m x 0.25 m x 0.38 m (2 off)
H123	8.6 m^2 heat transfer area	4 bar (shell) 2 bar (tubes)	140 °C (shell) 60 °C (tubes)	LP Steam	1.0 m tube length 0.4 m shell diameter
C124	6.8 m x 0.55 m packing	1 bar	60 °C (max)	-	1.0 m diameter 11.25 m length
H125	470 m^2 heat transfer area	1 bar	47 °C	Cooling water	3.60 m tube length 1.52 m shell diameter

Item Number	Process Specification	Working Pressure	Working Temperature	Services	Dimensions (PLS derived)
T126	1.3 m^3 holdup	1 bar	47 °C	-	0.9 m diameter 2.15 m length
H127	390 m^2 heat transfer area	1 bar	60 °C	LP Steam	3.75 m tube length 1.41 m shell diameter
P128	1.1 l/s	2 bar	47 °C	Power	1.0 m x 0.25 m x 0.38 m (2 off)
D129	2.0 m x 7.0 m drum	0.8 bar	100 °C	Power	9.0 m x 2.5 m x 2.5 m
V130	10 m^3 working volume	Atmospheric	Ambient	-	2.0 m diameter 3.66 m length (inc. cone)
X131	8.3 kg/s, 1 shift/day	Atmosheric	Ambient	Power	4.0 m x 2.0 m x 2.5 m
X132	7.73 m^3/s throughput	0.8 bar	40 °C	-	1.0 m diameter 3.4 m length
P133	7.73 m^3/s throughput	0.8 bar	40 °C	Power	2.0 m x 2.0 m x 2.0 m
H134	7.65 m^3/s throughput	Atmospheric	100 °C	LP Steam	3.0 m x 2.0 m x 2.0 m

Table 2: Test Process Stream Summary

From	To	Composition (% w/w)					Flowrate (kg/s)		Flow cycle	Density	Phase
		Water	Acetone	Monomer	Polymer	Air	Minimum	Maximum	(min)	(kg/m^3)	
T101	R104 R108 R112	100	0	0	0	0	2.5	2.5	30 in 180	1000	Liquid
T102	R104 R108 R112	0	0	100	0	0	0.97	0.97	30 in 180	1218	Liquid
T103	R104 R108 R112	0	84.74	15.25	0	0	4.92	4.92	30 in 180	835	Liquid
R104	H105	0	100	0	0	0	6.80	6.80	480 in 540	1	Vapou r
H105	R104	0	100	0	0	0	6.80	6.80	480 in 540	790	Liquid
T106	R104	100*	0	0	0	0	$3.7x10^{-4}$	$1.5x10^{-3}$	480 in 540	1000	Liquid
R108	H109	0	100	0	0	0	1	Vapour			
H109	R108	0	100	0	0	0	6.80	6.80	480 in 540	790	Liquid
T110	R108	100*	0	0	00	0	$3.7x10^{-4}$	$1.5x10^{-3}$	480 in 540	1000	Liquid

From	To	Composition (% w/w)					Flowrate (kg/s)		Flow cycle	Density	Phase
		Water	Acetone	Monomer	Polymer	Air	Minimum	Maximum	(min)	(kg/m^3)	
R112	H113	0	100	0	0	0	6.80	6.80	480 in 540	1	Vapou r
H113	R112	0	100	0	0	0	6.80	6.80	480 in 540	790	Liquid
T114	R112	100*	0	0	0	0	$3.7x10^{-4}$	$1.5x10^{-3}$	480 in 540	1000	Liquid
R104 R108 R112	T117	30	50	10	10	0	8.33	8.33	30 in 180	897	Liquid
T117	F119	30	50	10	10	0	2.80	2.80	Continuous	897	Liquid
F119	T117	30	50	10	10	0	1.38	1.38	Continuous	897	Liquid
F119	T120	60.84	26.32	5.2	0	7.5	2.64	2.64	Continuous	13	Gas/ Liquid
T120	P121	0	10	0	0	90	0.22	0.22	Continuous	1	Gas
T120	H123	66.40	27.89	5.74	0	0	2.42	2.42	Continuous	939.7	Liquid
H123	C124	66.40	27.89	5.74	0	0	2.42	2.42	Continuous	939.7	Liquid
C124	H125	0	16.7	83.3	0	0	2.93	2.93	Continuous	1	Vapou r
H125	T126	0	16.7	83.3	0	0	2.93	2.93	Continuous	839	Liquid

From	To	Composition (% w/w)					Flowrate (kg/s)		Flow cycle	Density	Phase
		Water	Acetone	Monomer	Polymer	Air	Minimum	Maximum	(min)	(kg/m^3)	
T126	C124	0	16.7	83.3	0	0	2.20	2.20	Continuous	839	Liquid
C124	H127	97.57	2.01	0.41	0	0	2.20	2.20	Continuous	995	Liquid
H127	C124	97.57	2.01	0.41	0	0	2.20	2.20	Continuous	41	Liquid / Gas
T126	T103	0	16.7	83.3	0	0	0.73	0.73	Continuous	839	Liquid
F119	D129	60	0	0	40	0	0.35	0.35	Continuous	1000	Solid (Wet)
H134	D129	0	0	0	0	100	7.50	7.50	Continuous	1	Gas
D129	V130	0	0	0	100	0	0.12	0.12	Continuous	1000	Solid
V130	X131	0	0	0	100	0	0.13	0.41	1 shift	1000	Solid
D129	X132	4.99	0	0	0.26	94.75	7.73	7.73	Continuous	1.03	Gas
X132	V130	0	0	0	100	0	0.01	0.01	Continuous	1000	Solid
X132	P133	5.00	0	0	0	95.00	7.71	7.71	Continuous	1	Gas

Water compositions marked * include initiator at trace concentration.

3.2.2 Discussion of the Process

The test process comprises 42 equipment items. This is typical of a chemical plant that would be laid out as a single job. A process larger than this would be broken into plots which would be laid out effectively as separate tasks. Thus, the test process is the size of a typical practical layout problem.

The process includes five widely different unit operations: batch reaction (three separate units of R104 *etc*, R108 *etc* and R112 *etc*), vacuum filtration (F119 *etc*), continuous distillation (C124 *etc*), rotary drying (D129 *etc*) and bagging (V130 and X131). This is representative of the diversity of unit operations across the process industry as a whole and tests the generality of PLS (albeit not rigorously as the range is a representative sample). The importance of this is perhaps easiest to see from a counter-example. A series of distillation units could have been set as the test case. This is a realistic process after all, widely used in oil refining. However, a special purpose distillation train layout system would produce results equally as good as if not better than a generally applicable system in this case. Even if the system were supposedly general, the generality would serve no purpose on this process and would not be put to the test.

The wide range of unit operations allows diverse equipment types to be introduced realistically. PLS includes a number of techniques for reasoning about the equipment items. Particular techniques are most appropriate for particular equipment types and *vice versa*. For example, while inheritance of default values might be used for the majority of reasoning about pumps (because pumps are highly standardised), inheritance would be of limited use for a column or dryer because these are designed as bespoke items. The reasoning techniques in PLS and how each is used are discussed in Chapter 5. The range of equipment types in the process provides real needs for each technique.

Furthermore, the diverse equipment types includes some which present specific layout problems. The difference in elevation between the column C124 and the thermosyphon reboiler H127 has to be calculated precisely for H127 to function as a thermosyphon. This layout constraint is unique to thermosyphons. Other examples include the barometric leg between the filter F119 and the filtrate vessel T120 and the need to minimise transport lag between the reactors R104, R108 and R112 and their respective condensers H105, H109 and H113.

Streams of all phases and all two-phase combinations flow within the process. In addition to the many single phase streams, the reboiler return (H127 to C124) is vapour dispersed in liquid; the filtrate barometric leg (F119 to T120) is liquid dispersed in vapour; the reactors discharge a slurry of solid in liquid (*eg* R104 to P116); the filter discharges this as a paste (F119 to D129); and solid is entrained in the air discharged from the dryer (D129 to X132). Each combination of phases impacts on the layout by imposing specific constraints. For example, PLS knows that pipework carrying a slurry has to include large radius bends rather than elbows so that solid will not be disentrained and deposited. PLS left additional space around the reactors to accommodate the large bends in the discharge pipework.

The process includes some batch processing (the reactor units) while the remainder operates continuously. In a batch process, even normal operating conditions vary with time. PLS represents a number of values for each of many attributes and then selects the governing value for each. For example, the reactor discharge streams variously have no flow and full flow. PLS identifies the flow rate at full flow as the governing case before sizing the pipework.

The batch processing sections are important to test PLS's ability to lay out plant operated in either mode. Furthermore, governing cases have to be identified while laying out all plant, whether batch or continuous. For example, the size of relief header piping must be calculated from assumed relief flowrates. There should be no flow in the header during normal operation. The batch sections of the test process also serve as a general test of this aspect of PLS.

3.3 Results

PLS was applied to the test process described above and designed a three-dimensional layout for the plant with no manual intervention. At the end of each successful run, PLS produced a data file which could be read into the widely used process plant modelling system "PDMS" to generate a model. PDMS was then used to view and plot the results. An industrial user of PDMS generated colour shaded perspective illustrations of the layout to be used for project publicity using PDMS's "REVIEW" facility. In principle, any modeller could have been used. PDMS was adopted because it was easily available to the author. Normally, PDMS's standard viewing and plotting functions were used to assess the results of a run. The REVIEW facility was not normally available to the author. These illustrations are shown in Figures 2 to 7.

This layout is entirely conventional and might have been produced by an engineer. It contains no anomalies and would not excite argument in most design offices. Madden [1990] wrote of it:

> "The very ordinariness of [the layout] is the real vindication of [PLS] as a workable engineering tool"

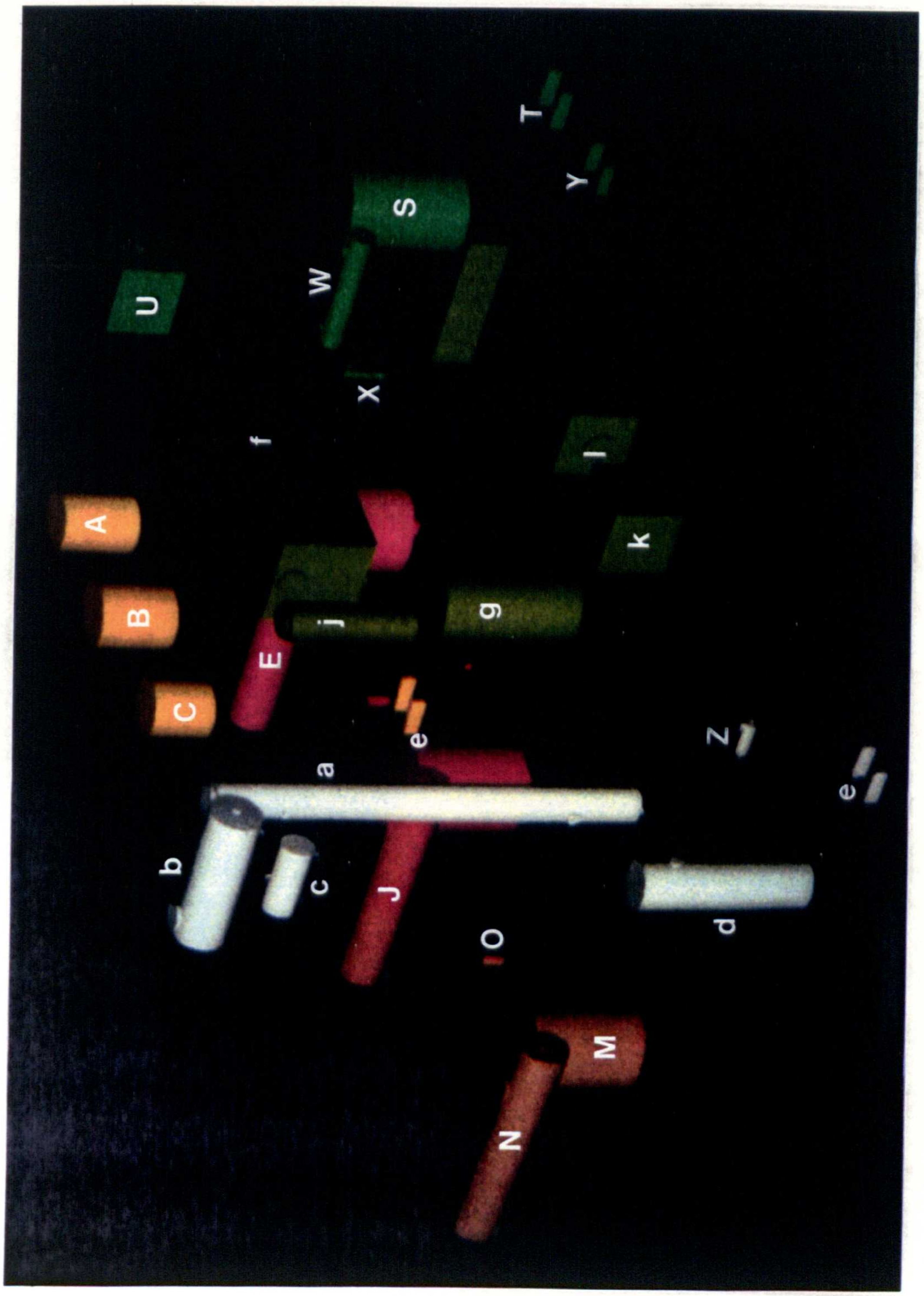
and added:

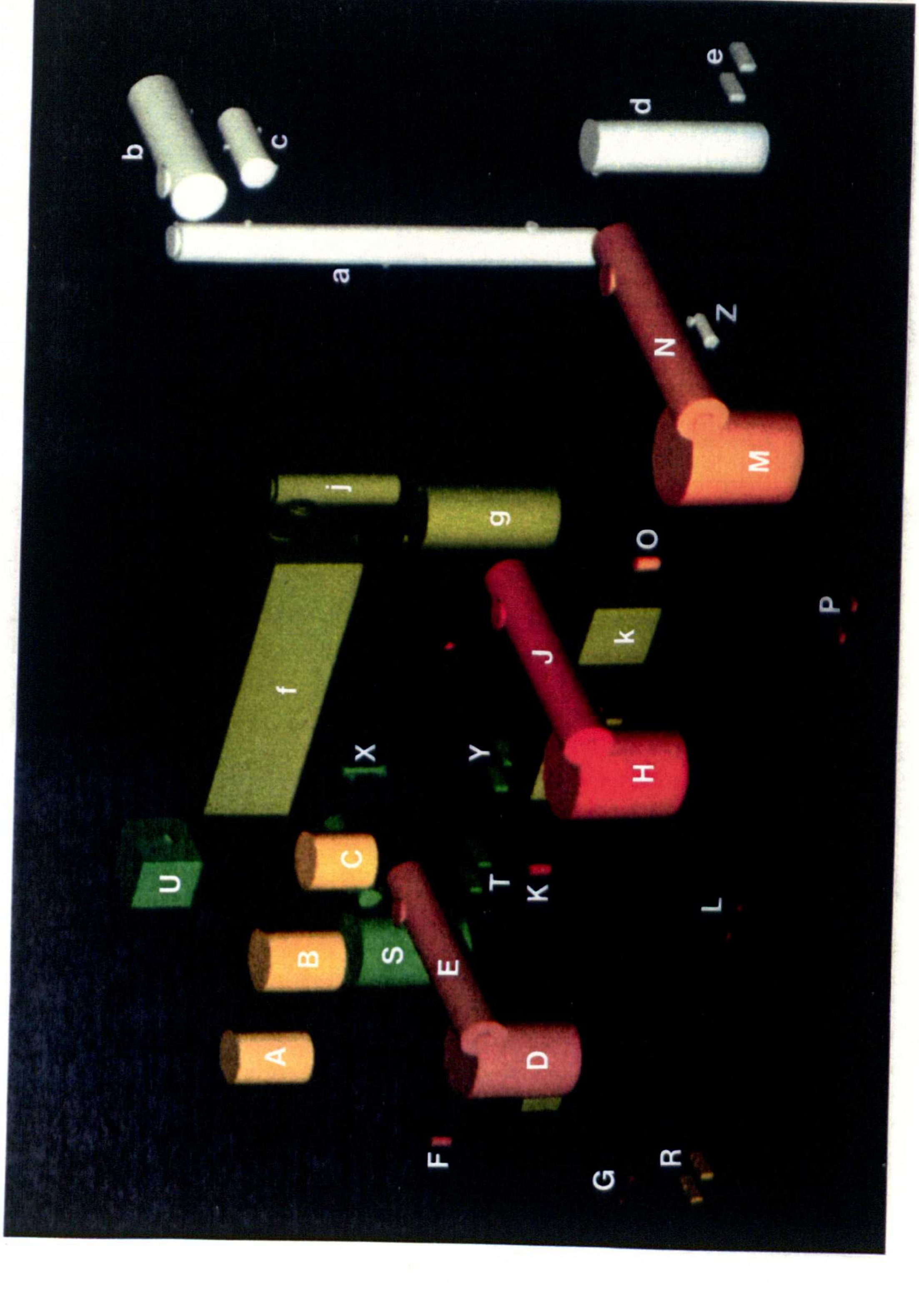
> "[PLS] produces results as good quality, coherent layouts"

The layout is assessed and critiqued from an engineer's perspective in this section. Madden's view is based on a similar assessment. The description of its features also indicates the decisions that PLS had to make to design it. The assessment of the layout is by proxy an assessment of the capabilities of PLS itself.

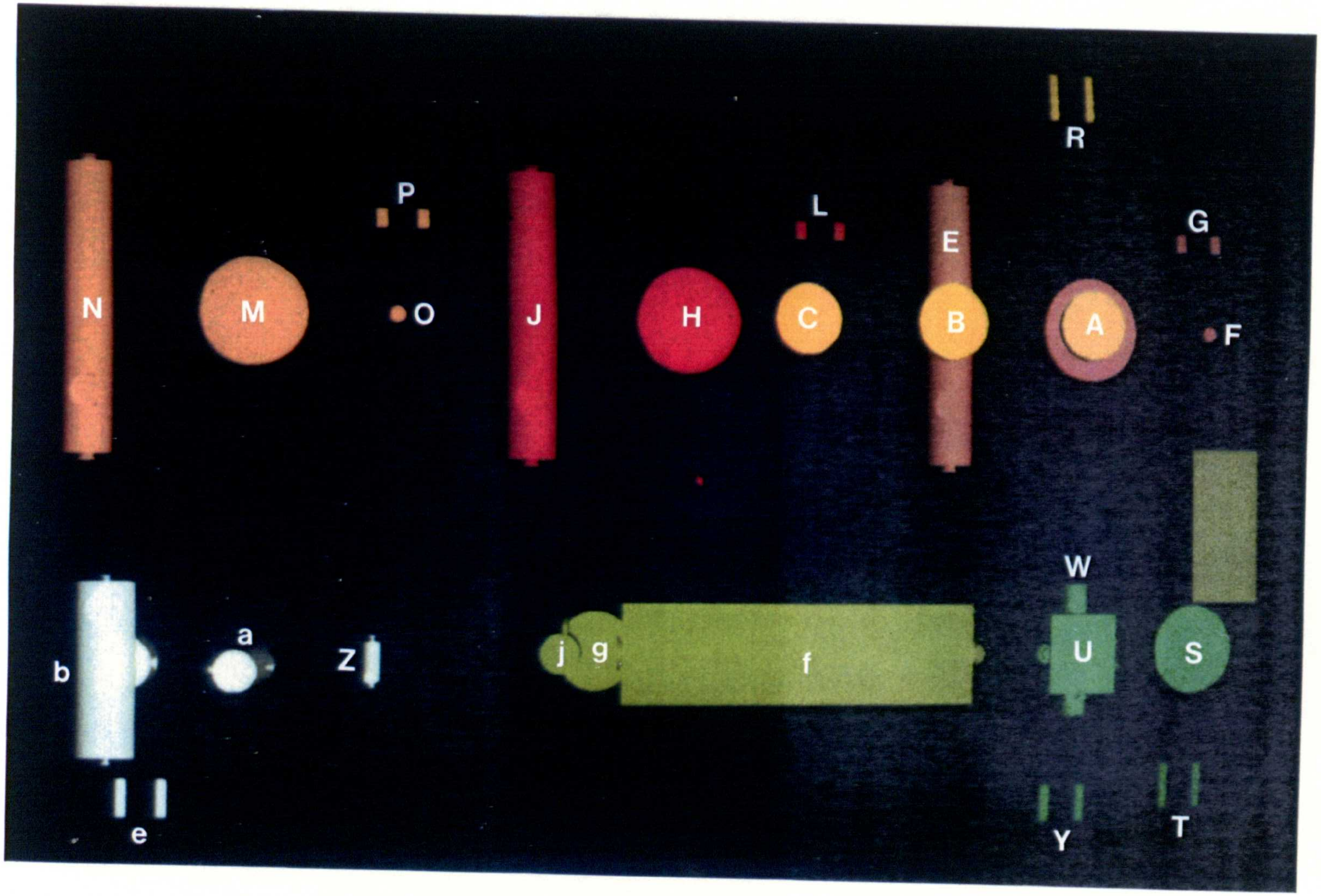
The plant has been laid out to be constructed in a building, commensurate with the need to protect the process operators, certain machinery (the bagger and palletiser) and the dry powder product from the weather. The overall form of the layout is one which is frequently used for plants in buildings. The process equipment are arranged in two rows on either side of a central access way with piping run down the outside of the rows. Process material flows around the layout efficiently. The weighing tanks T101, T102 and T103 are positioned in

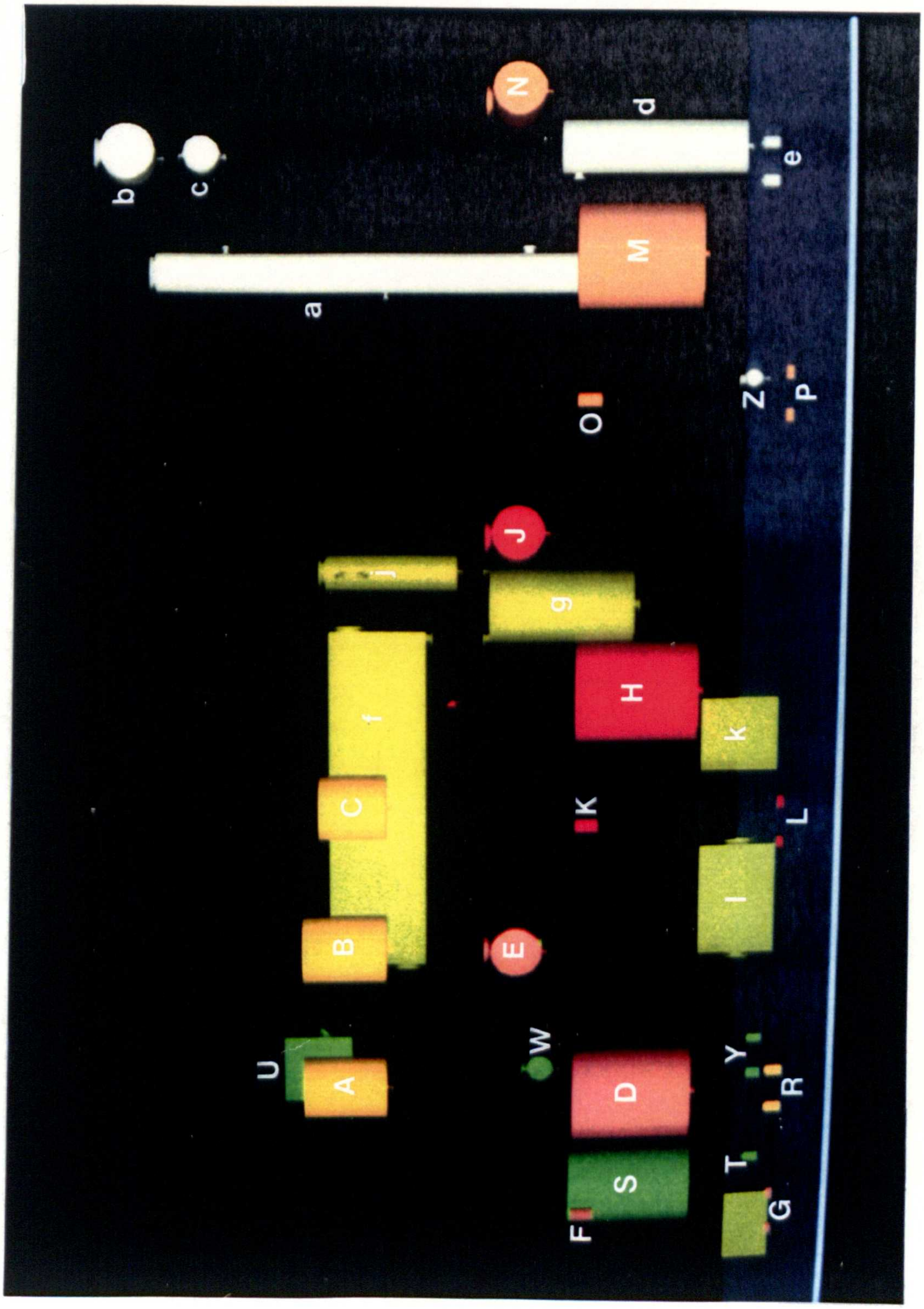
Illustrations of Test Process Layout Generated by PLS

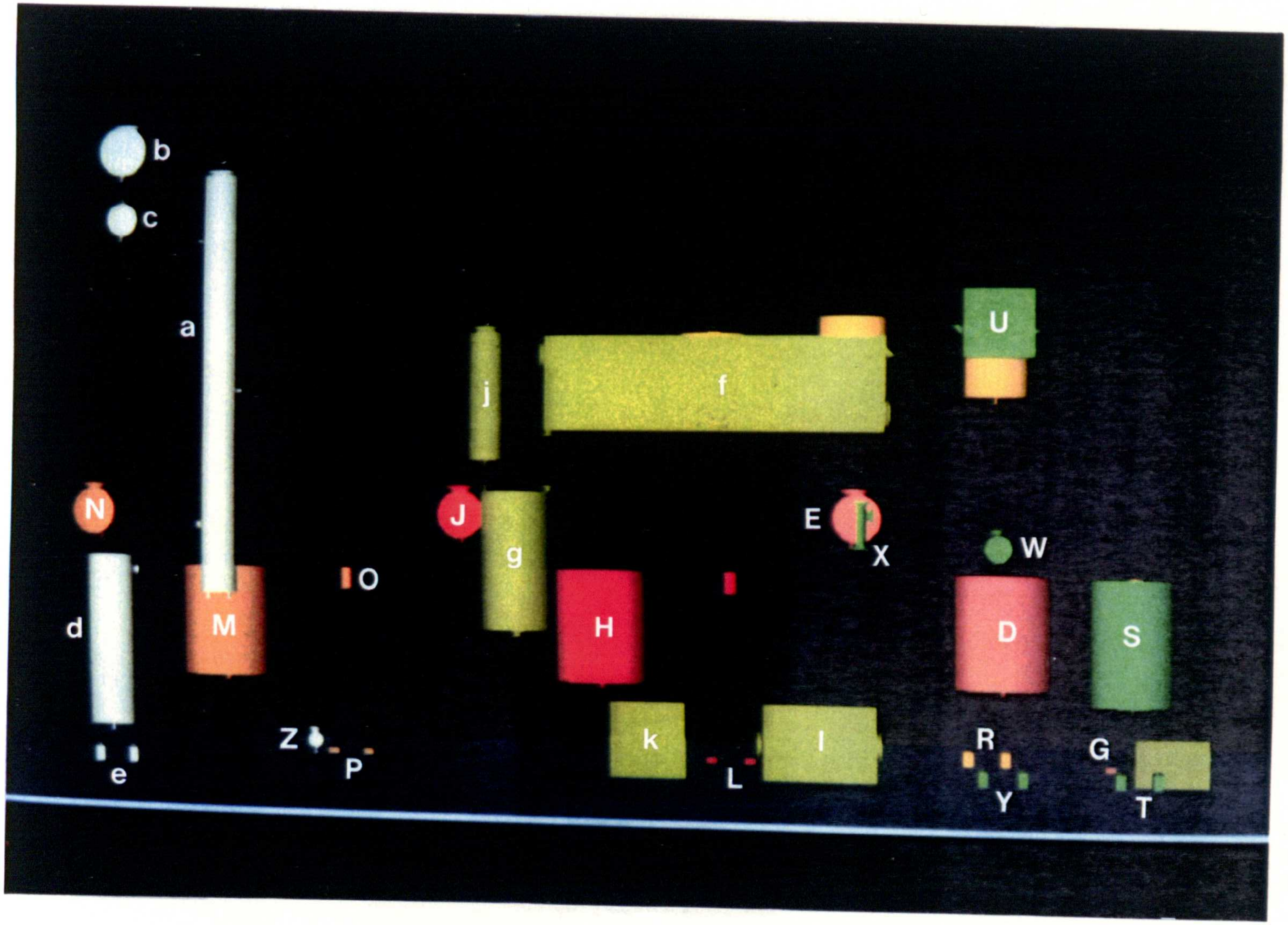

A
B
C
E
j
g
k
l
f
X
U
W
S
T
Y
a
e
Z
e
b
c
J
O
d
M
N

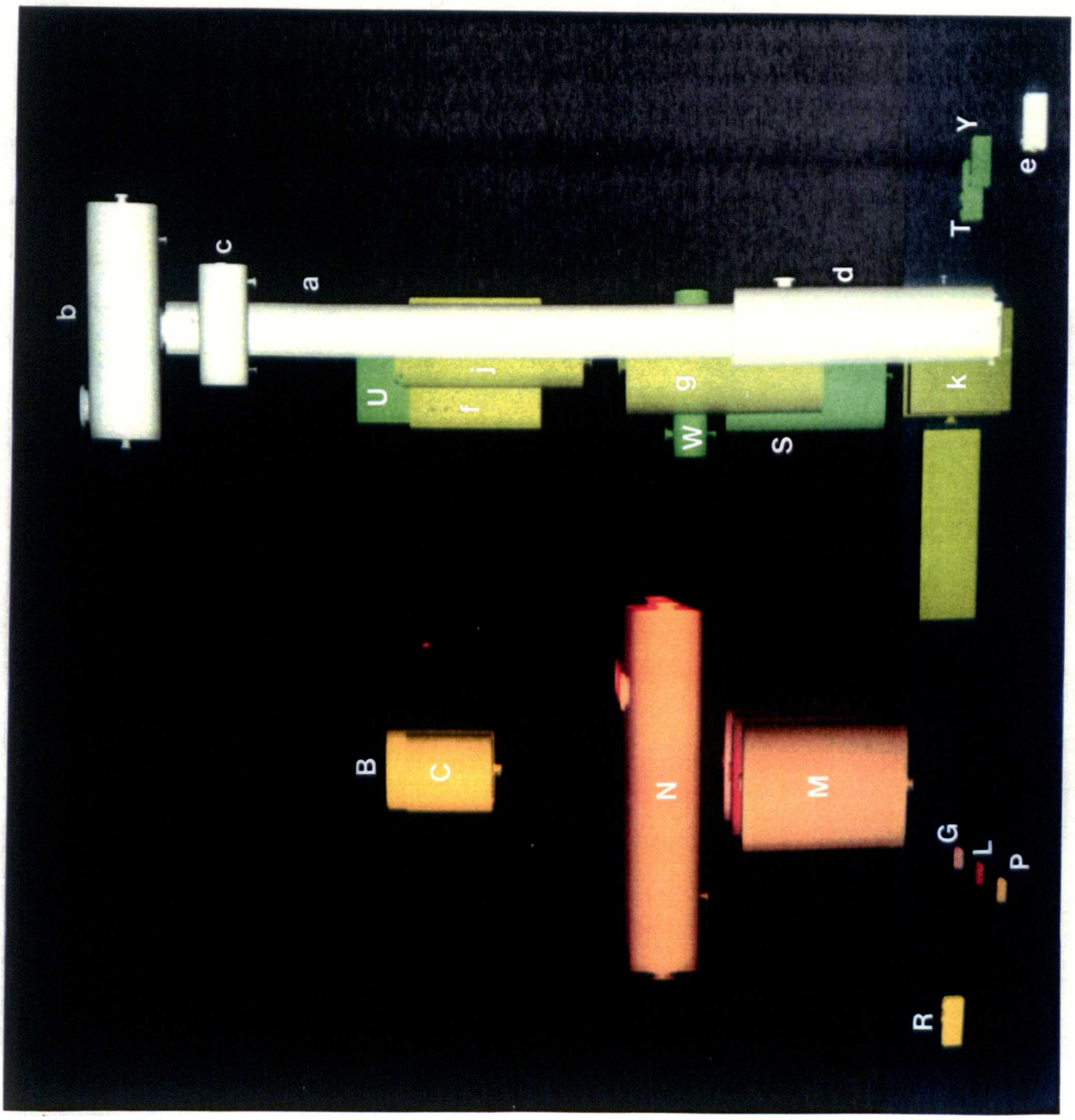


A
B
C
D
E
F
G
H
J
K
L
M
N
O
P
R
S
T
U
X
Y
Z
a
b
c
d
e
f
g
j
k


R
P
L
G
E
N
M
O
J
H
C
B
A
F
W
b
a
Z
j
g
f
U
S
e
Y
T


b
c
N
d
e
a
M
Z
P
O
J
j
g
f
H
k
C
K
L
I
B
E
W
U
A
Y
R
D
T
S
F
G


b
c
a
N
d
M
e
O
Z
P
j
J
g
f
H
k
L
I
E
X
U
W
D
S
R
Y
G
T



b
c
a
d
T
Y
e
U
j
f
g
W
S
k
B
C
N
M
G
L
P
R

A	T101	Water weighing tank
B	T102	Monomer weighing tank
C	T103	Acetone weighing tank
D	R014	Reactor
E	H105	Reflux condenser
F	T106	Catalyst make-up tank
G	P107	Catalyst metering pump
H	R108	Reactor
J	H109	Reflux condenser
K	T110	Catalyst make-up tank
L	P111	Catalyst metering pump
M	R112	Reactor
N	H113	Reflux condenser
O	T114	Catalyst make-up tank
P	P115	Catalyst metering pump
R	P116	Reactor discharge pump
S	T117	Filter feed tank
T	P118	Filter circulation pump
U	F119	Rotary vacuum filter
W	T120	Air disengament vessel
X	P121	Vacuum ejector
Y	P122	Filtrate return pump
Z	H123	Column feed pre-heater
a	C124	Acetone separation column
b	H125	Column condenser
c	T126	Column reflux drum
d	H127	Column reboiler
e	P128	Column bottoms return pump
f	D129	Polymer dryer
g	V130	Dry polymer hooper
h	X131	Bagger/palletiser
j	X132	Cyclone
k	P133	Blower
l	H134	Dryer air heater

A	T101	Water weighing tank
B	T102	Monomer weighing tank
C	T103	Acetone weighing tank
D	R014	Reactor
E	H105	Reflux condenser
F	T106	Catalyst make-up tank
G	P107	Catalyst metering pump
H	R108	Reactor
J	H109	Reflux condenser
K	T110	Catalyst make-up tank
L	P111	Catalyst metering pump
M	R112	Reactor
N	H113	Reflux condenser
O	T114	Catalyst make-up tank
P	P115	Catalyst metering pump
R	P116	Reactor discharge pump
S	T117	Filter feed tank
T	P118	Filter circulation pump
U	F119	Rotary vacuum filter
W	T120	Air disengament vessel
X	P121	Vacuum ejector
Y	P122	Filtrate return pump
Z	H123	Column feed pre-heater
a	C124	Acetone separation column
b	H125	Column condenser
c	T126	Column reflux drum
d	H127	Column reboiler
e	P128	Column bottoms return pump
f	D129	Polymer dryer
g	V130	Dry polymer hooper
h	X131	Bagger/palletiser
j	X132	Cyclone
k	P133	Blower
l	H134	Dryer air heater

the North Eastern corner, closest to the tank farm. Raw materials flow into the weighing tanks and are distributed to each of the three reactors R104, R108 and R112. Reaction products are pumped to the holding tank T117 from each reactor. Note that R112 might appear to be distant from T117 but a convoluted configuration with an unacceptably tortuous central access way would be needed to avoid this. PLS's solution is as good as can be achieved where flow splits and reconverges like this. From the holding tank T117, flow continues through the filter F119 where it splits into two. The liquid stream continues through the column unit and returns to the acetone weighing tank T103, flowing round the Western end of the plant. T103 is positioned most Westerly of the weighing tanks because of the acetone recycle flow. The solids stream flows through the dryer, hopper and bagger and out to the warehouse situated South of the plant. The section of the plant handling solids is positioned closer to F119 than the column unit because solids are far more difficult to transport over significant distances than liquids.

The layout is uncluttered and provides adequate access to the equipment via a central access way which PLS determined was needed. All equipment can be reached directly from the access way. Personnel need not risk injury climbing through equipment and pipework to carry out any operational or maintenance task and can return to the access way quickly to escape if an incident occurs. The access way itself is straight which also greatly facilitates escape. The equipment items are highly visible so operators can observe them easily in passing and therefore quickly become aware of any problem as it arises.

The layout also meets the different access needs of major maintenance tasks. Any equipment can be removed without disturbing other equipment. For example, all exchanger tube bundles can be withdrawn without fouling other equipment, in the main by pulling the bundles into the access way following typical practice. The reflux condensers are offset from their respective reactor vessels so that the vessel heads and agitators can be removed. The weighing tank T101 is positioned on the same centre line as reactor R104 to fit the pitch of the structural members.

However, T101 is sufficiently far above R104 to allow this while leaving sufficient space for removal of the head from R104. Similarly, the tubes of the vertical reboiler H127 can be withdrawn without fouling the drum T126 even though H127 is directly below T126. Similarly, the layout is sufficiently spacious to be operated and maintained safely and economically but it does not occupy excessive space or extend piping runs un-necessarily.

In plant layout, a number of positions are almost equally acceptable for some items of equipment. For example, the ejector P121 is positioned under the dryer D129. This is the only one of a number of otherwise equally acceptable positions chosen because it does not force other items out of their ideal positions. Importantly, PLS has successfully selected a good position for the ejector even though the constraints on the ejector alone do not suggest any position.

PLS does not attempt to route piping or design the plant structure. Nonetheless, allowance has been made for the pipework and structural steelwork in positioning the equipment. The layout is designed so that it is sympathetic to the constraints facing the project engineers. Sufficient space has been left for pipe runs between equipment. For example, the pipe run from reactor R104 to its condenser H105 requires two elbows because of the horizontal offset. Each elbow adds a distance equivalent to 1.5 times the pipe diameter to the minimum vertical offset. PLS has provided for this and for other pipe fittings such as nipples between the elbows and flanges when calculating the elevation of H105.

The span between major vertical structural members in any plant cannot exceed 6 m because this is the maximum practical length of the horizontal members. Equipment that requires support from all sides is positioned in the centre of the 6 m squares that result. In this layout, the reactor vessels R104, R108 and R112 are hung from the first floor and are laid out on a 6 m pitch in the bays of the steelwork. Other equipment requires support at specific points. The dryer D129 is supported at the ends and thus, it is positioned so that its ends align as far as possible with the stanchions.

The plant is laid out on three floors at 4.2 m, 7.2 m and 11.2 m elevation respectively. These elevations balance the needs for access and support. Fork lift trucks and machinery used during maintenance can manoeuvre beneath the first floor. The headroom below the second and third floors leaves sufficient clearance for the operational and maintenance activities needed on those floors. However, the floors are sufficiently close to one another to provide support at most elevations at which an equipment item needs it.

The relative elevations of the equipment items are all sufficient to satisfy flow and other process requirements. The elevation of each vessel that feeds a pump is calculated to ensure sufficient hydrostatic head at the pump's suction nozzle to meet its required NPSH. The reflux drum T126 is sufficiently far above the reflux return nozzle on column C124 to maintain an adequate pressure on the control valve. The elevation difference between the reboiler H127 and column C124 is calculated to provide the pressure differential to drive the thermosyphon. The barometric leg in the filtrate withdrawal line between filter F119 and the drum T120 is satisfied. Note that some equipment items are subject to more than one "chain" of elevation requirements. For example, the filter F119 has to be sufficiently far above the dryer D129 for sufficient slope in the paste chute and also sufficiently far above the filtrate drum T120 to satisfy the barometric leg. The actual elevation of the filter is the greater of the two.

The layout has avoided undesirable local support structures for more than a few items to achieve the elevations set by process requirements. The weighing tanks T101, T102 and T103 do not sit directly on the second floor. The elevation of the second floor is set to suit the dryer D129. This elevation is too low for the weighing tanks. If the second floor were set to suit the tanks, the dryer would be lifted un-necessarily or would need a local structure to support it above the first floor. This would be unsatisfactory given the size and weight of D129. Local support for the weighing tanks is the less un-satisfactory solution.

Access is provided at all levels where operators and maintenance staff work frequently. For example, the reactor vessels R104, R108 and R112 are mounted on the first floor so that their heads are approximately 0.6 m above the floor level. PLS specified this to place the access manhole at what is considered to be the ideal distance above a floor for safe access. The catalyst make-up tanks T106, T111 and T114 are positioned on small structures so their open tops are at chest height so the catalyst can be added in small amounts without the operator either bending or stretching. All frequent access requirements are met by the main floors rather than platforms or ladders for personnel safety.

PLS has laid out a plant which is representative of typical process plant layout problems and has generated, fully automatically, a layout that is entirely satisfactory and conventional from an engineering viewpoint. This result shows that a successful approach to automating all aspects of process plant layout by Expert System has been developed. The approach has been implemented successfully in PLS. All objectives of this work have been achieved fully. The remainder of this thesis elucidates the approach used in PLS which could equally be employed in any other Expert System for this application.

Chapter 4 : The Principles of PLS

PLS is a working Expert System that performs the valuable function of generating automatically three-dimensional plant layouts from process data. PLS implements an approach to automated layout that was developed as an essential element of this work. This approach can be used in any Expert System for this application and the approach is perhaps a more important deliverable than PLS itself. The remainder of this thesis describes this approach.

The approach is presented in the context of how it is implemented in PLS. This is deliberate to stress that the abstract approach can be implemented in working software which can then solve real world problems. However, descriptions of the approach and of PLS are synonymous. PLS is a full implementation of the approach and does not embody any techniques other than those prescribed by the approach. This Chapter highlights the key principles that pervade the approach developed during this work and the overall structure of PLS. This Chapter is intended as an introductory summary to unify and integrate the presentation of the more detailed descriptions of the elements of PLS in subsequent Chapters. The reader should bear this in mind and recognise that detail is limited in this Chapter of necessity.

4.1 The Role Of Constraints

In plant layout, design constraints suggest positions for items of equipment relative to other items rather than absolute positions of each item. The design constraints can be viewed as relationships between the positions of the items. Some constraints demand the proximity of the related objects (such as a pair of items connected by a large-bore pipe made of an exotic alloy) or the separation of the related entities (such as a fired heater and a tank storing flammable liquid in bulk). Some items of equipment are related to entities other than other equipment. These entities include elements of the structure or building in which the plant is built; ancillary elements such as control rooms; and site features which might be involved in the operation of the plant such as a warehouse or might be unconnected neighbours such as a population concentration. PLS reflects the central role of relationships in conceptual layout and adopts them as a unified expression of the spatial requirements implicit in the process data and of the relative positions of the equipment items chosen to satisfy these requirements.

The relationships explicate the spatial significance of the process conditions and features of the equipment items. They express the spatial requirements in a manner that is meaningful and appropriate for the task in hand. Effectively, they distil the factors that are germane to layout from the bulk of the process data. Shuquair [1978] also adopted relationships as the input data to his system, arguing similarly that this would

> "concentrate attention on the main key to the solution, that is, understanding the process"

Relationships are also highly important in the conceptual model of the layout domain. However, this importance must be suppressed in manual methods because of the volume of data needed to allow a proper treatment.

While the relationships impose requirements on the relative position of equipment items, they cannot be used directly to dictate an absolute position. Until the absolute position of one equipment item is known, the absolute positions of all others related to it cannot be determined. However, the positions of these other items will also be subject to constraints from further items. The position chosen for the first item must be chosen to satisfy the constraints acting upon its "neighbours" from elsewhere. In effect, the whole layout must be generated simultaneously.

The technique of "constraint propagation" is ideally suited to solving problems like this in which many sub-problems interact. A constraint propagation system does not choose a particular solution to a sub-problem (in layout, the position of one item). Instead, constraints on the solutions of the sub-problems are identified. The problem-solver progresses by reducing the degrees of freedom in the problem until a solution emerges. The problem-solver moves between the sub-problems opportunistically, making the best use of its then current knowledge of the solution. This meets the needs of the spatial reasoning technique for the conceptual layout domain. Accordingly, constraint propagation was selected as the spatial reasoning component of the methodology developed in this work. The success of the methodology establishes the principle that constraint propagation is an efficacious technique for this domain. Furthermore, relationships can be represented directly as constraints. Thus, the model of the domain embodied in PLS matches the engineer's conceptual model. This facilitates knowledge elicitation and the user's understanding of a system and hopefully, engenders the user's confidence in the system's results.

Davis has enumerated six techniques for constraint propagation [Davis 1987]. The techniques fall into two broad categories. Those in the first category, often referred to as "constraint satisfaction", use the constraints to control the values that may be assigned to attributes in a consistent solution. The problem-solving mechanism might search for an assignment of values that are consistent with the constraints imposed. Alternatively, the constraints might be used actively to propagate a known value to derive a value for an unknown. The seminal use of this style of reasoning is often taken to be the work by Waltz on understanding pictures by identifying the features within them [Waltz 1975]. The techniques in the other broad category work by reasoning about extant constraints to infer new constraints and thereby increase the problem-solver's knowledge of the solution. Davis refers to these as "constraint inference" systems, although the generic term "constraint propagation" has become somewhat ambiguous because many authors use it specifically to mean this technique. Well known examples of systems using these techniques include MOLGEN [Stefik 1981a, 1981b], ENV [Kuipers 1984] and the Quantity Lattice [Simmons 1986].

Constraint inference was also selected as the constraint propagation technique for conceptual layout in this work. The constraints implicit in the process data specify an under-constrained problem in one sense in that they do not convey enough information to lead directly to a solution. In another sense, the problem is over-constrained in that constraints are often encountered which are mutually inconsistent; this is discussed below. Rather, a solution has to be synthesised, albeit to maximally satisfy these constraints. This precludes reasoning by constraint satisfaction alone because the constraints will allow broad ranges of values to be assigned to many of the attributes rather than restricting the attribute to a single value that constitutes the solution. The constraint inference technique in PLS embodies the idea that new constraints should be created to record intermediate information about the relative positions of the objects, initially qualitatively then with an increasing content of quantitative information. These new constraints must be consistent with the constraints expressing the spatial requirements, of course. For example, a constraint might require two heat exchangers to be adjacent to share service piping. This constraint does not contain any indication of the form the solution should take. Increasingly specific constraints are created defining the relative positions of the exchangers until a highly specific constraint is recorded that states, for example, that one of the exchangers is 3 m North of the other. This relative position satisfies the design requirement that the exchangers should be adjacent.

Constraint propagation readily admits so-called "least commitment" reasoning. In this, commitment to a solution or partial solution is delayed if possible until sufficient information has become available to make a decision with certainty. Blind alleys are avoided while searching for the solution. They are not explored, only to be discounted during back-tracking, wasting a large amount of computing time. PLS has been engineered to employ least commitment reasoning widely. This greatly increases problem solving efficiency and is an important principle of

PLS. In an extreme example, it is doubtful whether blind search to form groups of entities (used to simplify plan layout) would even terminate. However, when PLS was applied to the test process, least commitment reasoning was found to eliminate completely back-tracking during group formation.

This work also establishes the principle that the instances of the design constraints can be identified by applying knowledge from a system knowledge base to the process data in a database. This principle sets this work apart from all systems built previously to automate layout. In the former systems, including Shuquair's, the user identified the constraints and stated them in the input data. This limited the use of these systems to skilled layout engineers who possess sufficient knowledge to do this. These are the very engineers who are most able to derive a layout using their traditional manual technique! The user was also forced to invest significant effort in his design. This made these systems unsatisfactory tools to develop provisional layouts, for example to support the evaluation of process options.

In PLS, the constraints are inferred by the knowledge base to great benefit. Further benefit accrues if each constraint identified by the knowledge base is represented persistently by a separate object in the database. This has also been adopted in this work. The constraints can be inspected at any time during the solution process and even after the design is complete. The objects representing the constraints provide a formal audit of a solution and a trace of how the solution was achieved. A process engineer might observe which constraints arise from a feature of the process design to assess the ramifications of the decisions that led to that feature. A plant engineer might inspect the constraints during a layout review to assess whether issues of interest to his discipline have been emphasised sufficiently. The layout might be validated against the constraints considered. Layout engineers cannot record sufficient information to allow their layouts to be audited when using the current manual technique. Conversely, the audit information is an intrinsic feature of PLS. The provision of an audit for the layout confers very important benefits to the system's potential users which will be discussed in Section 10.1.

Furthermore, the persistent representation of the constraints greatly increases the efficiency with which constraint propagation can be implemented. During constraint propagation, each constraint instance is referred to repeatedly. However, there is no need to re-infer the existence of each constraint each time because it is explicitly and persistently represented. This saving is important because the knowledge that infers the constraints is typically complex, reasoning simultaneously about three or four entities describing plant items and streams. For example, approximately one quarter of the time that PLS took to lay out the test process was spent in inferring constraints. Clearly, it would be prohibitive to repeatedly re-infer the constraints. The saving gained by re-inferring the constraints far outweighs the disadvantage of the resulting larger database. This approach is at odds with many other constraint reasoning systems, particularly those based on logic programming. In these, constraints are represented as generic data objects which are matched against the problem description as needed to support a specific inference.

A layout problem is invariably over-constrained and no solution can be generated in which all design constraints are satisfied. Thus, it is inevitable that some of the constraints that are identified from the process data are mutually inconsistent. In PLS, the inconsistencies amongst the spatial requirements are identified during the spatial synthesis phases and constraints are selected to be left unsatisfied. Every inconsistency between constraints is recorded when it is detected. These records are retained in the database and, like the constraints, can be inspected. Adopting this approach significantly enhances the value of the audit information for the layout in which these records are as important as the constraints themselves. They highlight where decisions are made which the user might choose to reverse to generate alternative solutions. More importantly, they manifest the compromises that are embodied in the layout. These compromises might suggest modifications to the process to enhance the layout. For example, PLS was forced to elevate the dryer D129 in the test process so that the solid could be discharged by gravity into the hopper V130. This forced PLS to leave unsatisfied the constraint requiring the dryer to be placed at grade because of its weight. Inspection of the records of inconsistencies would show this and might suggest to the process engineer that a bucket conveyor should be installed to remove the gravity flow constraint between the dryer and hopper. In an extreme case, the engineer might judge that some constraints which are left unsatisfied are critical to the operation of the plant. This would demonstrate that the process were technically infeasible because it could not be laid out.

4.2 Constraint Propagation in Layout

Constraint propagation is used in PLS to calculate the elevation of the equipment items and to derive their plan positions. One objective in constraint propagation in this domain is to identify the set of constraints that can be satisfied together and relax all others. Clearly, a globally consistent set of design constraints cannot be achieved unless all constraints are locally consistent That is, all constraints that relate a pair of items are consistent. Local consistency is relatively easy to assess and inconsistencies are relatively easy to redress. Thus, it is beneficial to construct locally consistent sets of constraints using a computationally cheap algorithm before invoking an algorithm to construct the globally consistent set. The latter is, of necessity, complex and computationally expensive. This principle reduces the number of "pointless" inconsistencies that the global consistency algorithm will identify and reduces the frequency with which it wastefully back-tracks. Thus, PLS adopts this principle and constructs locally consistent sets of constraints first then constructs one globally consistent set.

The ultimate objective of constraint propagation is to derive the positions of the equipment items. In PLS, the positions are recorded as relative positions of pairs of entities rather than as absolute positions of each. The relative positions are also expressed as constraints inferred during constraint propagation. Initially, these constraints are inferred from the constraints which record the spatial requirements implicit in the process data. As constraints are created which record some positional information, additional constraints are also inferred from these. It is clear that two categories of constraint are significant in this procedure. Constraints in one category express spatial requirements. These have been termed "Functional Relationships" or "FRs" during this work and this term will be used hereafter. The constraints in the second category express the relative positions of the entities. These are termed "Spatial Relationships" or "SRs".

Typical constraint-based systems do not enforce any dichotomy between constraints expressing solution requirements and constraints derived as part of the solution. However, the distinction is highly natural in this domain and is an important principle of this work. FRs and SRs play substantially different roles in the procedure for developing the layout and possess markedly different properties. FRs are formal statements of the spatial requirements intrinsic in the process data and it is inevitable that some of FRs are mutually inconsistent. SRs express the form of the solution or partial solution. Therefore all extant SRs must be consistent, by definition. Each FR can be in one of two states, it can be satisfied or unsatisfied. Each SR can only be in one state during problem solving, it either exists or it is completely absent from the database. If a set of SRs are found to be inconsistent, this is a manifestation of a more fundamental inconsistency amongst the FRs. The inconsistency can only be resolved by selecting FRs that must be changed to an unsatisfied state so that the SRs involved in the inconsistency can be deleted from the database completely.

The distinction between FRs and SRs is also important in determining which FRs are to be relaxed when an inconsistency is detected. Satisfying an FR implies that the requirement that it expresses is met in the layout. It is meaningful therefore to accredit a benefit to satisfying an FR. Conversely, SRs define the form of the solution so there is no benefit in a particular SR existing. Thus, only FRs should be considered when constraints are compared against one another to select the more important for satisfaction. The SRs should be neglected completely.

There are also significant implementational benefits from recording the positions of the items as relative positions expressed as SRs rather than as absolute positions written onto the representations of the items themselves. These are presented below.

1. An SR is a composite constraint which stands *in lieu* of potentially many FRs. Furthermore, only a proportion of the many FRs which relate any pair of items typically govern its position. The remainder impose looser constraints and are effectively subsumed by the governing FRs. Thus, SRs reduce the number of constraints that the propagation algorithms must handle, both directly, because they replace many FRs and indirectly, because the redundant FRs effectively disappear from the problem once the SRs have been formed.

2. SRs localise the effects of back-tracking because they define relative rather than absolute positions. This benefit is best seen from an example. Assume that an SR records the fact that object A is 3 m above object B and a second records the fact that object B is 2 m above object C. Assume that the elevation difference between objects B and C is re-assessed, perhaps because one of the FRs between them has to be relaxed. The SR between them must be updated accordingly. However, the SR between objects A and B still stands. The elevation difference between objects A and B has not changed even though their implicit absolute elevations have increased. This would not be the case if elevations were recorded absolutely. Note that this simple example does not show the full benefit of this feature. In many cases, the absolute positions of many items are influenced by the absolute position of a particularly "key" item. However, the relative positions of many pairs of these items are likely to remain the same even though the absolute position of the key item might change. The majority of the impact of this change is localised to the SRs which actually relate the key item.

3. The SRs embody the chains of FRs that govern the position of each item. If an item is found to be subject to inconsistent FRs, an FR must be selected to be relaxed. This FR must be one of those that influence the position of

the item in question. There is no benefit in relaxing any other FR. These FRs can be identified very quickly by looking at the SRs in the chain and considering the FRs that are recorded as governing each. In this context, the SRs are acting as the dependency records in a dependency directed back-tracker undergoing truth maintenance.

These benefits only accrue because constraints are created to record relative positions. This provides further evidence that constraint propagation is a highly apposite technique for this domain.

PLS's approach of achieving local then global consistency is similar in intent to the so-called "Arc Consistency" algorithms [Mackworth 1977] and "Path Consistency" algorithms [Montanari 1974, Mackworth 1977]. These algorithms transform a constraint network into simpler versions which still have the same solution so that this solution can be achieved without traversing all constraints in the network. The important examples of these algorithms are well reviewed by Nadel [1989]. All of these algorithms are intended to simplify constraint networks in which any pair of variables is related by very few constraints but in which the complexity arises because the network can be traversed by many paths. Any pair of variables in PLS is likely to be related by many constraints; much of the complexity of the network arises from this. PLS's procedures to form SRs are simple compared to the Arc and Path Consistency algorithms, some of which are highly complex. Nonetheless, PLS's approach suffices for this domain.

During this work, it has been found that the SRs can be propagated by mechanisms that are very simple compared to those used in more typical systems. PLS capitalises on this as a design feature, essential because of the intrinsic complexity of the constraint network in a layout problem. This has been possible because the generic constraints that are important in this domain can and have been identified. For example, PLS need only propagate constraints which record relative elevation to calculate the elevation of equipment items. This is analogous to Allan [1984] identifying the set of 12 relationships which are important in temporal reasoning. Thereby, it has been possible to identify all operators that are germane to these constraints and restrict the operators that are applied to the constraint instances to a very small set. Because only very few operators are needed, and because these can be predicted in advance given the task in hand, the constraints are propagated by dedicated algorithms for maximum speed. The operators are hard-coded into the algorithms during their implementation. This strategy has proved highly satisfactory in PLS and argues that a generalised constraint interpreter should not be employed in this domain.

4.3 Decomposing Layout into Phases

The FRs exhaustively define the spatial requirements that have to be met in a solution. However, they do not provide any explicit information as to the form of the solution nor do they even constrain it sufficiently to make it "obvious". A solution must be synthesised, a complex task. This work establishes a series of complementary but separate phases into which the task can be decomposed to render the problem tractable. PLS embodies this decomposition and progresses through the phases, representing the problem in appropriate abstractions, until the equipment items' spatial positions are generated.

4.3.1 The Pre-Processing Phase

Prior to the spatial synthesis phases, a pre-processing phase is required. The input data must be checked, the physical sizes of the equipment items must be calculated from the specification of their process duties and the FRs must be identified and recorded.

In the prototype of PLS, the process to be laid out is defined in a formatted file, much like a simulator input file, for example. PLS reads the file and builds a model of the process in its database. Before PLS attempts to generate a layout for the process, it validates these initial data. It does so using rules which search for both logical and engineering inconsistencies in the model. For example, rules are used to check the mass and heat balances around each item of equipment. It was found that incorrect specifications of connectivity accounted for nearly all errors in the input data. Checks on the mass and heat balances test the connectivity model fairly rigorously as well as validating many of the process data themselves. If errors are detected, PLS directs the user towards the likely cause in the input data. If PLS were re-engineered for use in a commercial design environment, it would be beneficial to interface it to a process database or intelligent schematics system to eliminate this data preparation step. This might obviate the need for the data to be checked.

Two of the benefits of using an Expert System for conceptual layout are only realised if the system estimates the physical sizes of the equipment from their process duties. This task is time-consuming for the layout engineer working manually and can actually absorb more time that the design of the layout itself. The task is routine and therefore, highly amenable to computerisation. Unless this task is automated in the Expert System (as it is in PLS), the time taken to carry it out manually before the system is invoked would preclude the use of the system to support preliminary cost estimates and feasibility studies. It would be simply impractical to invest significant man hours in this work while there is risk that the process option being studied might not be adopted. Furthermore, automating this routine task frees the engineer to spend more time optimising the layout itself, a more beneficial use of his time. This task is automated readily in PLS by applying relatively simple generic domain knowledge from the system knowledge base to the statement of the particular layout problem.

Similarly, the design constraints are rarely stated in the process data. They are, however, implied by these data. For example, in the test process, no design constraints were explicitly defined in the data input to PLS. Again, generic domain knowledge is applied to these data to identify the FRs and make them explicit. The constraints are inferred during this pre-processing phase.

Effectively all of the generic knowledge needed for conceptual layout is repeatedly applicable to plant after plant if coded appropriately. To be most widely usable, it must consider the generic properties of types of process equipment and process media. Thus, for example, generic knowledge in PLS's knowledge base asserts that streams with large volumetric flowrates must be carried by large bore pipes, that large bore pipes are expensive per unit length and that the length of expensive pipes should be minimised. These completely general assertions may be applied to any process to be laid out to identify all streams within the process with a large volumetric flowrate and infer that these require a large bore pipe whose length should be minimised. From this, it is inferred that the items connected by each of these streams should be close and an FR is created in the database. It is an important principle of this work that knowledge should not be expressed at the level of specific items or the juxtaposition of particular pairs or sets of items. PLS's knowledge base does not contain an assertion that a condenser and the column that it serves are related by a constraint requiring proximity, for example. Thereby, the knowledge can cover a wide range of situations and be applied to a wide range of processes. In the test process, the knowledge just exemplified applies equally to identify FRs relating the condensers and the batch reactors, the vacuum filter and its ejector and the dryer and its air heater. The knowledge is predicated on a particular combination of fundamental conditions. These conditions recur from plant to plant. Many plants contain large bore pipes for example. The knowledge that understands the significance of large bore pipes applies to each of these plants. Therefore, it should be widely applicable and reusable. It is for this reason that the knowledge base built to support testing of PLS is not specific to the test process, even though it contains no knowledge that was not required to lay out that plant.

4.3.2 Determining Elevations

Strictly speaking, layout should be thought of as a three-dimensional problem with elevation and plan layout tackled simultaneously. It is standard practice when working manually to establish the elevations of the equipment first then attempt the plan layout. This is necessary to reduce the complexity of the problem and actually has little impact on the quality of the layout. It is uncommon that the plan layout can be improved by changing elevation. For example, the layout of the test process cannot be improved at all by adjusting the elevations of any items of equipment. When this is possible, it is usually at the expense of increased structural costs. This strategy is adopted in PLS as a powerful simplifying assumption. The elevations of the items of equipment are calculated independently of developing the plan layout.

It is significant that elevations are calculated before the plan layout is derived in this methodology. All FRs which govern the elevation of an object have to be considered wherever the related object might be in the plant. Thus, the calculation of elevation cannot be localised within a physical area of the plant. Furthermore, some FRs which govern the plan positions of items arise because of the items' elevations. For example, the weighing tanks T101, T102 and T103 are incorporated into a group so they can share the support structure needed to elevate them above the reactors. These FRs only come to light once the elevation of the weighing tanks have been calculated. Thus, there is no benefit in attempting plan positioning prior to calculating elevations and a strong argument against. This should be taken as a general principle of this domain to be adopted in any system to automate layout. A related principle dictates that the knowledge that identifies FRs representing constraints arising from the elevations of the items should be applied at this time.

4.3.3 Forming Groups

In conceptual layout, the constraint network might comprise many thousands of instances for one plant. In the test process, PLS identified 1200 constraints for example. This whole network must be considered simultaneously during plan layout. In PLS, the plan layout task is decomposed into two separate but complementary sub-tasks. In the first, sets of entities which must be close in the final layout are collected into groups. These groups are then collected into increasingly more expansive groups and this procedure repeated iteratively. In the test process for example, PLS formed three groups which correspond to the reactor units. It then collected these three groups and the group of weighing tanks to form a larger reaction group. In the second sub-task, the members of each group are positioned relative to one another, only considering the FRs which relate the members either to other members of the same group or to other entities.

Thereby, the number of entities that must be positioned simultaneously is markedly reduced. Therefore, the number of constraints that must be propagated simultaneously is also reduced. Without this or some similar approach, it is likely that even the smallest of plants would prove intractable for any practical algorithm.

This approach accurately reflects a phenomenon that is widely observed in layouts produced manually. Groups of equipment are observed within any properly laid out process plant. The group might be a concentration of items with a common requirement, such as for support or a service, or might be the items comprising a process unit, such as the column, heat exchangers and pumps comprising a distillation unit. The group might exist as an entity in its own right, such as a compressor house. Groups themselves might also be collected into larger scale groups, such as distillation units being collected to share a high structure or load bearing ground.

In some cases, the FRs that relate the members of a group represent a need to share a facility or service. However, it is important that a group is not thought of as being a set of entities with a common need such as this. Entities are collected into a group solely because they need to be close to the other members, even though the need for closeness arises from the need to share a facility or service.

The approach has been discussed so far as a means to reduce the scale of the task that the plan layout algorithm must undertake at any time. This is indeed one of its purposes. However, the approach actually allows four very powerful simplifications to be made. It is important to note that superficially, these simplifications might seem to be designed to mitigate deficiencies in the plan layout procedure. However, the intention is to distribute a computationally difficult task (plan layout) between two simpler abstract steps (grouping and then simplified plan layout).

It is highly important nonetheless to note that FRs must be able to influence a layout even though the equipment items that they relate are not members of the same group. This establishes three principles that must be adhered to within this approach. Firstly, these FRs must still cause the groups that include the entities to be collected into more expansive groups. This might leave the entities reasonably local to one another in any case. Secondly, the FRs must be considered while deciding the relative positions of the groups which contain the entities during plan layout. Finally, the FRs that relate a group member to a member of another group must be allowed to influence the internal layout of the group. This ensures that the groups are not laid out to be locally optimal but rather, to fit together in the overall layout.

A Problem Reduction Strategy

The whole constraint network which might comprise many of thousands of instances must be considered simultaneously during plan layout. It is unlikely that an algorithm could be devised that could manipulate this many constraints and yield a solution in a realistic time. However, the majority of constraints act over relatively short distances and exist within concentrations of entities. Plan layout can be considered to be a series of localised sub-problems integrated into a single overall solution.

The process of forming groups establishes coarse scale plan layout information. The groups define which proximity relationships will be satisfied in the solution. The task of the plan layout phase which follows is then simplified to specifying the detailed juxtaposition of entities that are known by then to be local. Thus, the process of forming groups divides the difficult task of developing a layout into a series of easier complementary sub-tasks.

The groups themselves are also powerful as abstract entities. They reduce the size of problem that the plan layout algorithm has to solve concurrently. The plan positions of the members of a group are derived relative to other group members. That is, the groups focus attention on determining the correct relative positions of a small number of entities and only a small number of constraints have to be considered. The constraints that relate the members of the group are important to the task in hand and are considered duly. The constraints relating entities that are not members of the group are not relevant to the task and are neglected. Furthermore, each group stands *in lieu* of its members while the groups are relatively positioned to develop the overall structure of the plan layout. The members have no individual identity and "inherit" an approximate position from their parent group. This approximate position is then refined when the members are positioned relative to one another. Thus, at any time, many fewer entities than the total number of items of equipment in the plant are being positioned. This use of groups is similar to the technique suggested but not adopted by Grason [1971] but later adopted by Gilleard [1978].

Partial Satisfaction of FRs

The trade-off between increased separation of entities and the level of satisfaction of the constraints that relate them can be treated very simply in conceptual layout. A constraint that requires proximity can be in one of three states of satisfaction. The constraint is fully satisfied if the related entities are as close as possible, partially satisfied if the related entities are reasonably close and unsatisfied otherwise. Effectively, any reasonable degree of proximity is equally as beneficial, but adjacency is significantly better. This accurately models the domain. For example, consider two items of equipment connected by a pipe. If the two items are very close, the pipe can be run directly from nozzle to nozzle. However, if the items are further apart, the pipe will most likely be run via a pipe rack. Tubing is consumed in the runs from each nozzle to the rack. The run along the rack also consumes tubing but this contributes a surprisingly small proportion to the overall length unless the items are very widely spaced. Thus, the relationship between the total cost of the pipe and separation of the items can be approximated well as a stepwise relationship which correspond to adjacent, reasonably close and distant. The importance of proximity is, of course, the inverse of the relationship between cost and separation.

Clearly, a constraint that requires the entities that it relates to be adjacent cannot be partially satisfied. If the entities are adjacent, the constraint is satisfied fully. However, it is not the nature of constraints in this domain to demand adjacency, even though colloquially, it is widely held that pairs of items of certain types must be adjacent. For example, many engineers would state that a thermosyphon reboiler must be adjacent to the column that it serves to minimise pressure drop in the pipework. However, the perceived need for adjacency is false. There is no harm in a small item being positioned between the column and reboiler provided that they remain as close as possible (probably dictated by the space needed to remove the reboiler tube bundle).

In PLS, the principle has been adopted that all FRs that require proximity are guaranteed to be at least partially satisfied if the entities that they relate are members of the same group. This arises directly from the definition of a group stated above. Effectively, FRs are satisfied in two steps. If the FR is sufficiently important to warrant at least partial satisfaction, the entities that it relates are collected into a group. If the FR is to be fully satisfied, the entities are juxtaposed appropriately during the plan layout phase. Even if the constraint does not receive explicit attention during the plan layout phase, it is already partially satisfied because the entities that it relates are members of the same group. Thus, no explicit action need be taken to partially satisfy an FR during plan layout. This is important because it simplifies the implementation of the plan layout algorithm. More importantly, this yields a meaning or criterion for partial satisfaction and a means to achieve it which actually models the domain. The distance that the entities can be separated before the FR cannot be considered to be satisfied differs from one type of FR to the next. This issue is discussed in Section 8.2.1.

This is extended to provide a treatment for constraints which only impose coarse spatial requirements. In plant layout, some constraints that require closeness and all constraints requiring the entities to be distant only impose coarse spatial requirements on the related entities. The former are satisfied fully if the entities are within the same general area, the latter if the entities are not in the same general area. No additional benefit arises by placing the related entities particularly close to or far from each other.

For example, a process might include two vessels with large motors fitted to their agitators. If the majority of the process area were classified electrically as Zone 1, it would be beneficial to position the vessels together in a well-ventilated bay. Standard motors could then be fitted rather than expensive flame proof alternatives. Once the vessels are placed in a bay, the constraint is fully satisfied and their exact juxtaposition does not affect the benefit of it being so.

Groups provide a very natural treatment for constraints that impose these coarse spatial requirements. Once the entities related by such FRs have been collected into the same group, the FRs expressing coarse spatial requirements will be satisfied. These FRs can be neglected during the plan layout phase. Similarly, if a constraint requires the related entities to be distant, each of the entities is placed into a separate group. Thereby, the task of positioning the two entities far from one another is separated from the task of positioning each entity relative to the other entities to which it has to be close.

Groups as Macroscopic Entities

In some cases, it is inappropriate to place an entity in the "best" position to maximally satisfy the FRs acting on it alone. Rather, a number of entities must be positioned so that the FRs relating them all are equally satisfied. For example, consider a pump and its standby pumping the bottoms stream from a column. Constraints require the pump and the standby to be close to the column to minimise pressure loss in the suction pipework. A further constraint requires the pump and the standby to be adjacent to share a plinth. In this case, the optimal configuration is one in which the pumps are placed alongside each other to share the plinth. Thereafter, the pair of pumps should be positioned with the centre-line of the pair aligned with the centre-line of the column. Thereby, neither of the

FRs which relate the pumps to the columns are fully satisfied but both are satisfied as much as possible.

This configuration is achieved if both pumps are collected into a group. This group then acts as a macroscopic entity. The two constraints that originally related the pumps to the column now relate the pump group as a whole to the column. These define the position of the pump group relative to the column and the pump group is therefore positioned correctly.

Plan Meta-Knowledge

Layout engineers appear to rely heavily on stereotypical solution fragments which they have acquired over many projects to develop plan layouts. Each stereotype captures a typical configuration of a limited area of a plant. The methodology developed during this work relies heavily on meta-knowledge that captures the principles of these stereotypes. "Meta-knowledge" is knowledge that directs an Expert System to use its other knowledge more efficiently or powerfully. Groups provide a local context which corresponds to the areas over which this meta-knowledge can be applied satisfactorily. For example, one stereotype enforces a linear arrangement on the group members to align them with a pipe rack or access way. The meta-knowledge that corresponds to this stereotype might be used to position all equipment items in a plant. Even so, it would be unacceptable to lay out the whole plant by applying the meta-knowledge to all of it. The optimal layout at a macroscopic scale might be topologically distinct from a single line of equipment alongside a pipe rack. For example, if flow were to diverge, an overall layout that was Tee-shaped would be desired. Thus, although the stereotype might be entirely satisfactory if applied repeatedly to small areas of the plant, it would break down if applied globally.

4.3.4 Plan Layout

The derivation of the plan layout of the plant is the most creative task for the engineer, and proved to be the most difficult to automate. Progress was made by employing meta-knowledge that captures the principles of stereotypical solution fragments which experienced layout engineers acquire over many projects. The topology of FRs within the group is analyzed to determine which meta-knowledge should be applied to lay out its members.

During plan positioning in PLS, the members of one group are positioned relative to one another at a time. The effect of FRs that relate a group member to members of other groups are also considered. Thereby, the internal layout of the group reflects all FRs imposed from throughout the plant so the groups fit one another when assembled to create the overall layout. However, this approach greatly reduces the number of equipment items are positioned simultaneously. For example, the test process comprises 42 items of equipment. The group with the most members is formed around the vacuum filter F119. This has 6 members, the filter F119 itself, the feed vessel T117, the suction vessel T120, the ejector P121 and two groups. These groups comprise the pump P118 and its standby and pump P122 and its standby respectively. This approach also reduces the number of FRs that PLS has to consider simultaneously. Approximately 1200 constraints were identified from the test process. However, only approximately 50 constraints relate the members of a group to one another and to other entities external to the group.

This approach can only be employed satisfactorily if the boundary of the group crossed by each FR that relates a member to a member of another group is known. This can only be known from the relative positions of the related groups. However, the space occupied by each group has to be known before their relative positions can be determined. Thus, the sizes of the groups are estimated, then the groups are relatively positioned, before the equipment within them is laid out.

The size of groups whose members are equipment items can be estimated with greatest confidence. The equipment has known and fixed size. Thus, the dimensions of these groups are estimated first. The effects of FRs on members of other groups can be neglected for these estimates and each group can be laid out in isolation. Examples worked by hand by the author showed that a group's size is effectively the same whether the external FRs are considered when it is laid out or not. The major effect of the external FRs on the internal layout of the group is to change the order of the members along the group. This might add or remove scope for small items to be tucked into a small gap in a position where the FRs acting upon them are also satisfied. However, it is only the small items that can be positioned to make use of otherwise empty space and their position has little effect on the overall size of the group. This procedure is repeated to estimate the size of the increasingly expansive groups.

Once the sizes of all groups are known, their relative positions are derived. The largest scale groups are positioned first since they are constrained by fixed site features such as the position of the warehouse and tank farm in the example process. The members of each of these groups are positioned within them and so on until the individual items of equipment have been positioned within their groups. The positions of the members of each group are expressed relative to the local origins of their groups at this time. Once every group has been laid out internally, the positions of their members are translated to coordinates relative to a global origin.

Chapter 5 : Representational Issues

Brachman *et al* [1979] suggested that the representation of the knowledge and data representations used in an AI application should be described from four somewhat independent viewpoints. Each viewpoint represents a different level of abstraction and the names "epistemological", "conceptual", "logical" and "implementational" levels were suggested.

The specific domain knowledge and entities represented in the application are considered at the epistemological level. Issues considered at this level include the types of object (*eg* vessels and pumps); their relationships (*eg* connectivity); their specific attributes (*eg* flowrate and temperature); the knowledge applied (*eg* equipment connected by a high temperature stream should be close to minimise heat loss); and the partitioning and control of knowledge.

The conceptual level viewpoint considers the generic forms and structures which are used to represent the specific domain knowledge elements and entities. The more abstract issues that are considered include the generic representation of the domain objects (*eg* are they represented as single entities with many attributes or described by a number of isolated and independent facts); the types of the underlying relationships between the entities (*eg* similarity between instances, instances as aggregations of others and entities as generic exemplars of instances); the general form in which knowledge is expressed (*eg* rules, inheritance hierarchies and equations); and the general control of the application of knowledge (*eg* whether rules chain forward or backward, whether inheritance searches breadth- or depth-first and whether inference is monotonic or non-monotonic).

The logical level focuses on the logical completeness and correctness of the inference mechanism. Issues such as speed of the system, memory usage and robustness are addressed at the implementational level.

PLS's structure is typical of an Expert System. The knowledge that layout engineers use is recorded in the knowledge base. The database records the data that describe the specific process that PLS is to lay out. Any intermediate information that PLS derives is also recorded in the database. This Chapter describes the forms in which the knowledge and data are represented in PLS, and the mechanisms that PLS uses to reason with its knowledge, from the viewpoint of the epistemological and conceptual levels. Note however that PLS formalises some entities that the engineer only loosely acknowledges, such as FRs and groups. These entities are described in depth in later Chapters wherein their representation is included.

5.1 Representing the Data

Many facts are required to adequately describe each of the equipment items comprising the process that was laid out by PLS. For example, 35 facts are required to describe C124, the distillation column in the test process. Some facts record symbolic concepts, for example stating that a column is oriented vertically, others record numeric values, such as a specific internal temperature for the column. Similarly, PLS has to record many facts to describe each stream of process medium flowing between the equipment, such as the stream's temperature and flowrate.

Equipment items have a physical realisation and it is not surprising therefore that engineers perceive each as an entity. They also perceive streams in this way, even though streams are actually abstract concepts (they are distinct from the physical pipes that carry them). Streams are given identifying numbers, each has a graphical representation on the flowsheet and so on. Engineers perceive the facts germane to an entity as being an aggregation of attributes rather than a series of effectively disconnected descriptive statements. This argument is supported by the format in which a process design is formally documented, for example. Each of the principal entities comprising the process is described on a separate *pro forma* process data sheet. Engineers are entirely comfortable with this format which is used universally.

A "frame based" representation was selected for PLS to match the engineers' perception. The basic concepts of a frame based representation are due to Minsky [1975]. He argued that the apparent speed and power of mental processes suggested that fragments of knowledge in a human's memory must have more structure than exists in sets of disconnected facts. He introduced the concept of the "frame", a data structure that represents a stereotypical situation, formed on the basis of previous experience. The brain retrieves frames whenever a new situation is encountered to represent expectations about the situation. Minsky's concepts were implemented as a representational technique for use in Expert Systems. The frames are data structures that aggregate the factual descriptions of situations or entities with the procedural knowledge needed to supplement the factual descriptions. Winograd [1975] discussed the relative merits of declarative and procedural representations. He concluded that the former have many advantages but particular types of knowledge are better implemented in a procedural way. For example, PLS collects together the 35 facts about column C124 in a frame that represent the column. These comprise the declarative aspects of its representation. PLS also includes procedural knowledge about columns, such as how to calculate an estimate of their height from the number of trays in them. This procedural calculation is tightly integrated into the frame as an essential element of PLS's overall model of a column.

Some of the frames in PLS's database describe the entities that comprise the process and site such as the equipment items, streams and site features. PLS also derives facts about the entities, such as the physical sizes or coordinates of the equipment items. It adds slots to the frames to record these additional facts. PLS also creates new frames as it reasons. Some of these represent concrete entities such as access ways and structural steelwork members that PLS determines must exist in the plant. Others represent abstract entities such as groups or constraints. PLS uses frames as the representational structures for all data. The starting database for the test process occupied 203kB. This grew to 860kB by the time the solution had been achieved.

PLS holds the whole database of frames in memory while it works. It writes the frames to a formatted ASCII file at the end of a session then parses this file at the beginning of the next session to re-construct the database. PLS parses a textual description of the process to establish the database initially.

The features of PLS's frames are described below. A data model of the process and the layout was developed. Two particular aspects of the model are noteworthy - the representation of parts of entities and the representation of connectivity. These are highlighted in the description. The representations of the entities which PLS creates to record the results of the various phases of spatial synthesis, such as groups and constraints, are described in later Chapters.

PLS's frames are typical. Each frame represents a separate entity. The example below shows the frames for the reactor, R104, its agitator and the agitator's motor. The notation used in these examples is defined in Appendix A.

```
{S-1    [is-a vessel]
        [name R104]
        [geometry cylinder]
        [operating-volume 14.03]
        [weight 17000]
             :
             :
             :
        [top-agitator
             {S-2   [is-a agitator]
                    [head anchor]
                    [shaft-power 100]
                    [drive
                          {S-3  [is-a motor]
                                [required-power 100]
                                [voltage 440]}
                          ]}
                    ]}
```

The frame S-1 represents the vessel itself. Each fact about an entity is recorded in a "slot", each of which has a slot name and slot value. This frame is typical in that it has a number of slots. The slots with names "name" and "geometry" have symbolic values. The slots named "operating-volume" and "weight" have numeric values. These slots record the facts that the vessel's name is R104, that it has cylindrical geometry, contains 14.03 m^3 of reactants and weighs 17000 kg when full. The slots have no fundamental significance to PLS. Their semantics are defined entirely by the context in which they are retrieved and the subsequent use of their contents. The semantics are not explicit in the frame itself. Thus, dimensional units are not recorded explicitly by the numeric values because units are part of the slot's semantic. The only exception to this are the slots named "is-a". These record the cross-reference between the instance frames and their respective class frames. Thus for example, frame S-1 is an instance of the "vessel" class. The is-a slots are significant to the inheritance mechanism in PLS.

PLS does not differentiate between integer and real numbers. The underlying language in which the frames are implemented is LISP. It has a very flexible type system so it is not necessary to specify the data types for implementational reasons. The semantic difference between an integer and a real number might be significant but is not explicated.

Parts of an equipment item are often as real physically as the equipment items themselves. Typically, the parts exist independently of the entity of which they are a part. The agitator in this example is as tangible as the vessel on which it is mounted and has an existence of its own, for example. Thus, it is appropriate for PLS to represent the parts as frames that are distinct from, but related to, the frames that represent the items themselves. This also allows PLS to ascribe generic properties to classes of parts. PLS can reason about the parts separate from the objects of which they are parts when appropriate. For example, the generic properties of motors are identical irrespective of what they are driving. In the test process, PLS reasoned about the motors on pump P116 and on the agitator of reactor R104 to identify them as potential ignition sources without any need to consider what type of equipment each was driving. PLS could apply knowledge that matched the real world situation to the appropriate entities. This would not have been possible if the facts that described the motors were recorded in slots on the frames for P116 and R104 themselves. Thus, in this example, the "top-agitator" slot contains a frame rather than an atomic numeric or symbolic value. The frame, S-2, records the facts that describe the agitator. The "drive" slot on frame S-2 has a frame as its value. This frame, S-3, describes the agitator's motor but as an object in its own right rather than as a set of slots on the agitator.

PLS expresses the relationship between a part and the object of which it is a part in the name of the slot of which the part's frame is the value. The name "top-agitator" given to the slot on the vessel frame S-1 expresses the relationship that the frame S-2 represents an agitator mounted on the top of the vessel. This technique was conceived during this work so that complex relationships between the part and the object of which it was a part could be expressed succinctly. This differs from other frame languages in which frames are indirectly referenced via "owner" and "part-of" slots. PLS can draw inferences about the parts of entities as easily as it can about the properties of entities. For example, a vessel might have a number of agitators, mounted on its top and sides. PLS differentiates between these by placing agitator frames in either of two slots on frames for vessels. These slots could be called "top-agitator" and "side-agitator". PLS can recognise that the agitator on the vessel shown above is top mounted directly from the name of the slot which contains the agitator frame. Thereby, for example, PLS can easily infer the constraint on the vessel's position that arises because free space must be left above vessels with top agitators so the agitator can be removed.

It is essential that PLS represents the streams within a process to be laid out in as much detail as the equipment items. It is also essential that PLS represents the connectivity of the streams accurately and unambiguously. The descriptions of the topology, chemical composition and physical conditions of the streams are key to defining the operation of the process and the conditions in the equipment items themselves. In layout, many constraints on the positions of connected objects arise from the properties of the streams that connect them. PLS represents each stream as a frame which records the physical conditions of the stream. For example, each stream frame has slots to record facts such as its stream number, temperature, pressure and continuous phase. While connectivity might have been recorded as a property of the connected items, recorded on their frames, connectivity can only be fully understood if the properties of the stream itself are also recorded. Many texts on object oriented database design, such as Rumbaugh *et al* [1991], state that relationships that are qualified by attributes of their own must be reified[2] if their representation is to be unambiguous. Much of the knowledge in PLS about streams makes little more than passing reference to the equipment items to which the stream connects. For example, one rule states that two equipment items should be close if connected by a stream which contains particulate solids in suspension. This rule considers the properties of the stream in detail but makes no reference to any property of the connected equipment items. As well as being the correct solution from a computing viewpoint, this

2. "Reify" means to consider or make an abstract concept real or concrete.

particular choice of data model for streams matches engineers' long standing perception of streams as entities separate from the equipment that they connect.

The frames for streams are fundamentally similar to those for equipment items. The example below shows the frame for the stream connecting the weighing tank T101 to the reactors R104, R108 and R112.

```
{S-4    [is-a stream]
        [stream-number 1]
        [temperature 293]
        [continuous-phase liquid]
                :
                :
                :
        [contents
                {S-5    [is-a component]
                        [substance-name acetone]
                        [proportion 100]}
                ]
        [branches
                {S-6    [is-a branch]
                        [connects-to
                                {S-?    [name T101]
                                        :
                                        :
                                        }
                                ]
                        [direction in]
                        [flow-period 60]
                        [flow-cycle 180]
                        [flow-proportion 1.00]}
                {S-7    [is-a branch]
                        [connects-to
                                {S-1    [name R104]
                                        :
                                        :
                                        }
                                ]
                        [direction out]
                        [flow-period 60]
                        [flow-cycle 540]
                        [flow-proportion 1.00]}
                :
                :
                ]}
```

In general, a stream can flow from and to one or more equipment items. In PLS, the stream frames themselves do not contain any record of the items connected by the stream. Rather, each has a slot named "branches" with frames which are instances of the "branch" class as its value, S-6 and S-7 in this example. Each of these "branch" frames has a slot with name "connects-to" which has the frame representing the connected equipment item as its value. Thus, S-6 records the connection to the weighing tank T101 and S-7 records the connection to the reactor R104. This also exemplifies a slot with more than one frame as its value, and indeed the slot actually contained two more branch frames for the other two reactors. The "direction" slot on the branch frames records the direction of flow with respect to the stream. Thus, fluid flows from T101 to R104 (and the other reactors). The other three slots named "flow-period", "flow-cycle" and "flow-proportion" respectively record the time during which fluid flows through this branch, how frequently this occurs and how much of the total stream flow passes through the branch. Thus, fluid flows from T101 for 30 minutes in every 180 minutes and all flow passes through this branch. Conversely, fluid flows into R104 for 60 minutes in every 540 minutes although again, all flow passes through this branch. This structure can represent streams in which a continuous flow diverges or converges and also streams in which flow is switched between the parallel branches. This would not have been possible if the frames representing the items themselves were placed in the slots of the stream frame.

The composition of the stream is also recorded by using one frame to represent each component. In this example S-5 represents the acetone component of the stream. This representation allows the frames to represent the "one to many" relationship between streams and components - one stream has potentially many components. The number of components that PLS can record for a stream is not restricted to an arbitrary upper limit. Additional "component" frames may be created and added to the contents slot until every component of the stream is recorded. This would not have been the case if slots had been included in the stream frame to record component names and proportions (*eg* "component1-name", "component1-proportion", "component2-name", "component2-proportion", and so on). The representation of one to many relationships used in PLS is similar in concept to that widely used in relational databases. The approaches differ in detail however because of the difference in underlying technologies. In a relational database, each type of entity is recorded on a separate table rather than represented by a class of frame. The cross-references between the entities are implied by recording the unique key of the entity of which there is one against the entries in the tables for the entities of which there are many. The relationship is actually made explicit in the queries that retrieve data from the database. The approach used in PLS is more natural and direct than the approach use in a relational database.

The most important and innovative feature of PLS's frame language is this use of slots with frames as their values to establish arbitrary relationships between frames. This feature was named "recursive embedding" in PLS and had no published precedent at the time it was conceived. Recursive embedding allows the plant to be modelled in PLS's database in a way that is both practical and intuitive. The builders of many frame systems have judged that the relationship between a part and the object of which it is a part is highly important and must be recorded frequently. Therefore, they have provided a specific construct in their language that is used to represent the relationship wherever it occurs. By building the construct into the underlying language, these workers have provided it as a resource to the knowledge engineers who might use the language. However, the construct records a general relationship and the subtleties of a specific relationship between two frames might well be lost. Recursive embedding is a radically different philosophy, capable of expressing any relationship required by a builder of a knowledge base with all of its semantic content. It offers the knowledge engineer total freedom to set up relationships at will. PLS makes three specific uses of recursive embedding. These are enumerated below to abstract the principles and explicate "design rules" which suggest when the technique would be of benefit.

- Recursive embedding is used to represent the parts of objects. The parts are represented as separate frames to the objects of which they are a part. The function or purpose of the part is expressed in the name of the slot which holds the frame on the frame which represents the object of which it is a part. Recursive embedding is beneficial if a part must be described by a number of facts, if there is a need to reason about a part independently of the object of which it is a part or if there is a need to distinguish between parts which are intrinsically similar but serve a different purpose.

- Recursive embedding is used to represent the specific relationships between streams and objects to construct a sufficiently rich model of the connectivity and topology of the process. An intermediate "branch" frame is used to carry the reference from a stream frame to one equipment item frame to expand the description of the stream with respect to the particular equipment item. Recursive embedding is beneficial if an entity has properties that only apply with respect to one of many entities referenced from it. Intermediate frames can qualify the reference and state the properties that are unique with respect to the referenced entity.
- Recursive embedding is used to represent one to many relationships between entities. It is beneficial if an entity is best modelled with some of its properties represented by an aggregation of an arbitrary number of other entities.

5.2 Reasoning in PLS

PLS progresses through two distinct phases of reasoning while it develops a layout. In the first, PLS reasons about the equipment items and streams comprising the process to estimate values for unspecified process data; to estimate equipment sizes and develop coarse physical models of the items and then to infer the constraints on their positions. In the second phase, PLS reasons about the constraints to postulate the groups of equipment items; determine their elevations and finally to position them in the plan layout. Clearly, knowledge of a different nature is used in each phase. The knowledge in the first phase embodies the general principles of plant layout elicited from engineers and layout texts. In the second phase, the knowledge encompasses spatial reasoning and the implementation of the layout approach devised for PLS. Nonetheless, the knowledge used in both phases is represented using a common underlying representation language.

Early frame systems emphasised the process of "matching" an instance frame against all class frames to establish its class as a central part of their inference procedure. "KRL" [Bobrow 1977] is probably the best known example of this.

These systems were designed to accept a loose description of a situation or entity. They made sense of the description by identifying the stereotypical frame in their knowledge base with which it corresponded. This was very much in the spirit of Minsky's original concept. In contrast, PLS is typical of modern frame systems which have almost universally discounted matching. In PLS, the user specifies the class of each instance frame as part of the description of the entity in the initial process data. PLS then applies generic knowledge from its knowledge base to the frames to derive new facts about the process and the layout. The majority of the knowledge in the conceptual layout domain is procedural in form. PLS includes two representations for procedural knowledge in its language to reflect this emphasis. These are so-called "procedural attachments" which are invoked in goal directed reasoning and highly complex data driven rules. Note that the latter are markedly different in concept to traditional examples of this approach. PLS's knowledge representation language is the subject of this Section.

PLS does not include a mechanism by which an instance frame inherits declarative knowledge from its class frame. This contrasts with effectively all other frame based systems which embody this as a key feature. In these systems, some frames represent particular instances of problem elements while others represent classes of these problem elements. The class frames record facts that could apply to all instances of their class[3]. The description of an instance is complemented by the instance frame "inheriting" the generic facts from its class frame. These facts then become available as though they were part of the instance. For example, a class might be defined to represent ANSI "4 in x 3 in x 10 in" pumps. This would record the bore of the suction nozzle as 4 in and the bore of the discharge nozzle as 3 in. Assume that an instance frame were created to represent a particular pump on a flowsheet and that this instance frame is stated to be a member of the ANSI "4 in x 3 in x10 in" pump class. The instance frame would inherit the values of the suction and discharge nozzle bores from its class frame

3. Class frames represent values that must be true for all instances of the class in some frame languages. In others, the class frames record default values for use on the instance frames if a value is not specified.

if they were not stated when the instance frame were created. However, it was found that it is rare that values can be inherited usefully in the layout domain. Very few types of process equipment are sufficiently standardised for the classes to be anything more than broadly generic. In the main, an item of process equipment is designed for its specific duty. For many types of equipment, the only standards that are published do no more than prescribe accepted design procedures and calculation methods. The designs of some types of equipment are constrained by standards that either specify preferred values for key attributes or prescribe a number of configurations for key features that can be combined effectively at will. The Tubular Exchanger Manufactures Association [1978] or TEMA standard for tubular heat exchangers actually exemplifies both aspects. It specifies preferred shell diameters and lengths and the number of tubes that can be fitted into a shell of a given diameter. It also lists allowed configurations of tubes and heads, shell types, and flow patterns. The TEMA standard divides the configurations into meaningful classes. It does not specify classes for complete exchangers and the configurations can be combined in many ways. The layout engineer is interested in the exchanger as a whole, needing only to know its approximate size. In particular, the TEMA standard offers no "off the shelf" solutions for given heat duties, for example. This situation applies to most equipment types.

There are two exceptions to the preceding argument. Centrifugal pumps are often selected from ANSI [1984a, 1984b] or American Petroleum Institute [1986] standards. Each entry in either of these standards almost completely specifies the performance and physical design of a pump for a given duty. That is, if the required flowrate through a pump is known, its detailed design can in effect be read from these standards. The other exception is more general. All instances of each of a number of classes share a characteristic that impacts upon how they should be treated in a layout. Some knowledge is predicated on this characteristic, irrespective of the class of the instance to which it is being applied. For example, all pumps and all thermosyphon heat exchangers share the characteristic of pumping liquids, potentially to an increased elevation. The knowledge that identifies streams in gravity flow is predicated on the item from which the stream flows not having a pumping effect. In principle, every instance that has a pumping effect could inherit a flag value to indicate this. However, other techniques are essential in PLS to meet major needs of the conceptual layout domain that inheritance of values could not. These techniques, specifically the use of lookup tables and "procedural attachments" are as effective as inheritance in these special cases. These are described below. The only argument for including a conventional inheritance mechanism would be tradition. Clearly, this does not offer any justification in practice! Accordingly, inheritance of values is omitted from PLS. It would be highly recommended that the approach taken in PLS should be adopted in any Expert System for this domain.

Because it is inappropriate for frames to inherit values which are not specified in the input data, PLS has to derive them. It is for this reason that the initial database is very small compared to the size of the knowledge base, which occupies 1.3 MB. The database is not expanded by large class frames recording inheritable values. Rather, the knowledge, which is mainly procedural in conceptual layout in any case, is concentrated in the knowledge base.

Goal directed reasoning is highly appropriate to expand the process data and estimate the sizes of the equipment items. In PLS, goal directed reasoning is implemented using procedures which instance frames can inherit from their class frames to derive values for their attributes. These so-called "procedural attachments" are fairly commonplace in frame systems, although sometimes referred to as "if-needed demons". In PLS, they are invoked if the slot that should record the fact is absent from the instance frame. Slots are only added to the instance frame when a value is to be recorded. The procedural attachments retrieve their antecedent values from the instance frame and calculate a consequent value that is then written into the previously missing slot. The following typical example of a procedural attachment is written in the syntax of PLS's language. This procedural attachment calculates the shaft power required by a pump.

```
[procedure shaft-power pump
     [pattern
            [=self
                   [required-head =head]
                   [mass-flowrate =flow]
                   [hydraulic-efficiency =efficiency]]]
     [action
            [assign =power [=head * =flow * 9.81] / =efficiency]]
            [=self
                   [insert [shaft-power =power]]]
            [return =power]]
```

The procedure is introduced by the keyword "procedure" which is followed by the name of the slot to which the procedure is attached (or stands *in lieu* of) and the name of the class it is to be used with. Thus, this procedure is invoked when an attempt is made to read a missing "shaft-power" slot from a pump instance frame. The body of the procedure comprises two parts, the "pattern" and "action", introduced by the respective keywords. The pattern defines from where antecedent data are to be retrieved. The action specifies the calculation and the value to be returned.

All words preceded by an "=" symbol, such as "=self" or "=head", are variables. Variables can have either an instance frame or a slot's value assigned to them as appropriate. The variable "=self" is special in that PLS assigns the instance frame on which the procedure is running to be its value.

The form "[=self ....]" is a prototype for a frame for which PLS searches in its database and on which the enclosed forms are executed. In this case, the "=self" variable specifies the instance frame that the procedure should act upon and PLS need not search for it. Each of the enclosed forms represent one slot of the frame. PLS inspects these in turn. The order in which the slots are searched is independent of their position in the instance frame itself as the slot names provide indexed access. PLS inspects the "required-head" slot of the pump frame and retrieves its value (if it exists) and assigns this value to the variable "=head". If this slot has a value (or a procedural attachment is triggered which derives the value), PLS continues in turn to the "mass-flowrate" slot then the "hydraulic-efficiency" slot. If PLS fails to find any of the slots, it halts execution of the procedural attachment and no value is returned. This shows how the pattern is used as a means to specify what data should be retrieved from the database.

If PLS finds all of the required slots on the pump instance frame, the pattern is said to match and the procedural attachment continues into the action section. This commences with the "[assign ...]" form which calculates the value of the power using the values assigned to the variables in the pattern. The result of this calculation is assigned to the "=power" variable. In this example, the value of the required power is recorded in the database by the form:

```
    [=self
            [insert [shaft-power =power]]
```

The "[insert ...]" form creates a "shaft-power" slot and writes the value assigned to the "=power" variable into it. The slot is created on the frame assigned to the variable "=self". Finally, the "[return =power]" form instructs the procedural attachment to return the value assigned to the "=power" variable. It is not strictly necessary for a procedural attachment to create a slot and record the value that it derives. It is the value that it returns that is used *in lieu* of a value read from the slot. However, the value remains available for subsequent use if it is recorded and repeated invocation of the procedural attachment is avoided.

Procedural attachments ensure that all values needed by the rules invoked to infer constraints are available. They also ensure that PLS only derives values if they are actually needed by these rules. For example, PLS includes a rule that infers constraints relating equipment items that require a high voltage supply so that these items can be grouped to minimise the length of high voltage cables. This rule reads the value of the voltage required by a motor from the motor frame. A procedural attachment is defined for motors that derives the most appropriate voltage given the electrical power. A second procedural attachment for motors derives the required electrical power from the shaft power required by the object driven by the motor. The procedural attachment exemplified above derives the shaft power for a pump, and so on. Now, assume that the rule that identifies equipment items that are related because they require a high voltage supply attempts to read the voltage from a frame representing a motor driving a pump. If this value is not recorded on the motor's frame, the procedural attachment is invoked. The procedural attachment attempts to read the value of the electrical power from the motor frame. If this second value is absent, the procedural attachment that derives it would be invoked, reading the value of the shaft power from the pump frame. Similarly, this might cause the procedural attachment that derives the shaft power for the pump to be invoked. This is a form of goal directed or "backward chaining" inference. PLS only invokes a procedural attachment if doing so helps it to achieve its goal at the time of providing a particular value to the constraint finding knowledge. This goal is set by the constraint finding knowledge attempting to read an absent value. This is a typical example of a situation where this style of reasoning would be widely considered to be entirely appropriate.

The example above hints at the value of tying a procedural attachment to a particular class. The procedural attachment for motors that derive the required electrical power reads the required shaft power from the frame representing the object driven by the motor. This procedural attachment is appropriate whatever type of equipment a motor drives. However, the procedural attachment to calculate the required shaft power differs markedly depending on the type of the driven item. For a pump, the power is calculated from values of mass flow rate and liquid density. For an agitator, the power is calculated as a function of the speed and diameter of the agitator. Both the pump and agitator classes have a "shaft-power" attribute and both classes have procedural attachments defined that derive a value for this attribute. The procedural attachment is different for each class however. Thus, rules can look in the same place for a value of the required shaft power on an instance of either class. If the procedural attachment is needed, the inheritance mechanism of PLS decides which form of it should be invoked. The class of the frame is used to control exactly how an inference is drawn. This is known as "polymorphism" in Object Oriented programming (see [Stefik 1986]), and indeed, this aspect of PLS owes as much to the Object Oriented approach as it does to frame reasoning. The distinction between the two approaches is somewhat blurred and of theoretical interest in any case.

Although frames cannot inherit values in PLS, it includes a mechanism for looking up values in tables. The entries in these tables can comprise a number of related values. Three tables were required for use with the test process, although any one of these amply demonstrates the principles of the technique. Each entry in the first table records a tube bundle length, shell diameter, tube pitch and heat transfer area per shell for a tubular exchanger of a standard size, taken from the TEMA standards. Entries in the second correlate a preferred combination of diameter and tangential length for a vessel with the vessel volume, taken from the DIN standard [DIN 1979]. The third table correlates the physical dimensions of centrifugal pumps and their throughputs, taken from the API standard. These tables allow flexible access to the data they contain. An entry can be retrieved by applying a predicate to any of the values or combination of values that it might contain. The predicates themselves can include relational operators to specify complex criteria of the entry to be returned. For example, a predicate might be defined to search the table of heat exchanger sizes to find the smallest diameter shell for a heat exchanger with 1 in OD tubes on a square pitch, a heat transfer area greater than 54 m^2 but a tube bundle length less than or equal to 2.44 m or 8 feet. (The answer is 686 mm or 27 in). Procedural attachments including predicates like this example read these tables and retrieve the entry that is most appropriate. This mechanism provides PLS with an elegant representation for these standardised values. The tables store the information in a very compact and easily accessible and maintainable form compared to coding it directly into the procedures. Because the retrieved values are copied from the tables into the slots of frames, the values can be accessed very quickly when read subsequently. The table mechanism was highly successful in PLS and certainly proved to be a very effective alternative to inheritance.

Procedural attachments provide PLS with a goal directed reasoning capability. PLS also reasons using rules which are applied in a data driven manner. In any data driven rule system, including PLS, rules are invoked and take their action if specific data are present in the database. The system draws as many conclusions from available data as possible. This is exactly the behaviour that PLS needs while inferring and manipulating constraints. It is essential in PLS that all constraints are inferred so that the spatial synthesis algorithms can determine which are to be satisfied. This is an intrinsic feature of the constraint propagation techniques used in PLS. Furthermore, even the information that a constraint exists but cannot be satisfied is valuable to the user.

Data driven reasoning is widely used in Expert Systems. Each of PLS's rules have the general form that is universal in this approach:

if *precondition* **then** *conclusion*

The rules in PLS record individual fragments of generically applicable knowledge. The rules do not refer explicitly to particular items of equipment or streams, but rather, to generic types. For example, PLS applies a rule containing knowledge about pumps to every instance of pump in its database.

The following example of a rule is also written in PLS's syntax. The example is designed specifically to illustrate the principles of PLS's rule language. It performs the somewhat contrived function of printing the names of equipment items connected by a liquid stream carried in a large bore pipe. Practical rules in PLS were substantially more complex but the complexity would impede the clarity demanded of an introductory example.

```
[rule print-largebore
     [pattern
          [=stream
               [class stream]
               [continuous-phase liquid]
               [pipe-bore =bore > 300]
               [branches
                    [=in-branch
                         [direction in]
                         [connects-to
                              [=in-object
                                   [name =in]]]]
                    [=out-branch
                         [direction out]
                         [connects-to
                              [=out-object
                                   [name =out]]]]]
               ]
          ]
     [action
          [print "Pipe from " =in " to " =out " is large bore"]
          [print "Its bore is " =bore " mm"]]
     ]
```

The rule follows similar principles to the procedural attachment shown above. The rule is introduced by the keyword "rule" which is followed by its name, "print-largebore". Again, the body of the rule comprises a pattern section and an action section. These define the precondition and the conclusion of the rule respectively.

PLS generally attempts to match each rule against every instance frame in the database. This introduces a loop into the execution of the rule which allows it to consider every frame. In this rule, the variable "=stream" is assigned as its value the frame being considered at any time.

The "[class stream]" form in the pattern specifies that the rule should only be applied to instances of the stream class. The "class" keyword is widely used in PLS's rules. It speeds the matching process because it directs a rule to the frames to which the knowledge is intended to be applied. PLS makes no attempt to match the rule against the other frames. A rule might only consider a small proportion of the total number of frames in the database thereby. For example, consider a rule containing knowledge that might apply to each of the 60 streams in the test process. Even if this rule were run when nearly all of the approximately 1200 frames representing constraints had been added to the database, the rule would only attempt to match against 60 frames. The saving is obvious. The "class" keyword also adds expressiveness to the language by offering the rule writer a means to clarify to which types of entity the encoded knowledge is germane.

The next two forms in the pattern are similar to those in the pattern of the procedural attachment, and inspect slots of the stream. However, in this case, they specify conditions that must be met for the rule to match. The "[continuous-phase liquid]" form inspects the "continuous-phase" slot on the frame under consideration. The rule will reject this frame unless the slot contains the value "liquid". In this example, 300 mm is considered to be the minimum pipe size that constitutes a large bore. The construct "[pipe-bore =bore > 300]" specifies the requirement that the "pipe-bore" slot must contains a value greater than 300 for the rule to consider this frame further. The value of the slot is retrieved and assigned to the "=bore" variable. Sophisticated constructs such as this are essential in PLS to express the typically complex and subtle engineering knowledge. The patterns of procedural attachments can also express conditions and this is used in many real examples.

Stream frames have a slot called "branches" which contains "branch" frames. This rule retrieves the frames from the slot to gain access to the frames that represent the equipment items that the stream connects. The following section of the rule searches the "branches" slot for each branch with inward flow.

```
[branches
        [=in-branch
                [direction in]
                [connects-to
                        [=in-object
                                [name =in]]]]
```

This section of the code introduces a second loop as it assigns each frame in the "branches" slot to the "=in-branch" variable in turn. If the frame's "direction" slot has value "in", it is a branch with inward flow. The rule retrieves the frame that represents a connected equipment item from the "connects-to" slot of the branch frame. In principle, another loop is introduced here but the semantic of the database calls for this slot to contain one frame only. The value of the "name" slot is retrieved from the equipment item's frame and assigned to the variable "=in".

A similar process loops through the branches with outward flow in the next section of code. Note that the rule compiler nests the code to loop through the outgoing branches inside the loop introduced for the incoming branches. Thus, the action section is reached once for every combination of incoming and outgoing branch.

The action section in this example is very simple. It contains two "print" statements. The first prints a message such as:

"Pipe from T101 to R104 is large bore"

The second might print:

"Its bore is 300 mm"

Once the action section has been executed for every combination of the branches on the first stream frame, the outermost loop of the rule steps it onto the next. Thus, the rule traverse the database and prints messages for every stream that meets the conditions expressed in the pattern.

The rule language in PLS has two noteworthy features. One noteworthy feature, exhibited clearly in the example above, is that the rules search the instance frames in the database for their preconditions. It is somewhat atypical, although certainly not unique, to apply rules to complex data objects such as frames. More often, each fact in the database is recorded as a separate datum. However, frames are used in PLS to represent the problem elements and therefore, the slots of the frames constitute the facts to be reasoned about. Similarly, if a rule concludes a new fact about an existing object, a new slot is added to the object's frame. If the rule concludes that an entity should exist, a new frame is created. In practice, most of the entities inferred by rules are constraints.

The preconditions for many of the rules in PLS are complex and usually require a number of facts to be considered about an entity. It is not uncommon for PLS to need to examine 10 facts about an entity while matching a rule to it. PLS applies its rules very efficiently because the frames "package" all of the facts about an entity together. Once PLS retrieves a frame while attempting to match a rule, each fact can be retrieved directly by inspecting the appropriate slot of the frame. The slot names can be thought of as indices directing PLS straight to the facts.

To exemplify the complexity of knowledge needed in conceptual layout, consider the rule that infers that one object should be above another because a liquid stream flows from the first to the second under gravity. This rule searches for combinations of three entities that meet the following conditions. Two of the entities have to be equipment items, the third has to be a stream. The stream has to connect the two items. The bulk phase of the stream has to be liquid and the pressure differential between its ends has to be insufficient to drive flow upwards. The item from which the stream flowed cannot have a pumping effect, because this makes gravity irrelevant as the driver of the flow. Once the rule identifies a combination of two items and a stream that meet these conditions, it draws its conclusion. In this case, the rule creates a frame in the database for the FR representing the constraint that the source object must be above the sink object.

The above example can also be extended to show how rules and procedural attachments work synergistically. The rule in the example retrieves the pressure differential from the stream's frame. This pressure differential is calculated by retrieving the pressures from the frames representing the connected items. The calculation is encoded as a procedural attachment for streams. The procedural attachment reads values of pressure from the connected items' frames and writes the pressure differential onto the stream's frame. However, it only does this if the rule attempts to retrieve the value of pressure differential from the stream frame. If the stream or items have already failed to meet another criterion, the rule does not need this information and the procedural attachment is not invoked. Similarly, the rule determines whether an item has a pumping effect by attempting to retrieve the value of a "pumping-effect" slot from the item. A generic procedural attachment is defined for equipment items in general which returns FALSE for this. That is, in general, objects are considered not to have a pumping action. However, more specific procedural attachments are defined for classes whose instances might have a pumping effect. These more specific procedural attachments over-write the generic version and are invoked in preference. These procedural attachments return TRUE if appropriate. For example, a heat exchanger would not normally function as a pump unless it were being used as a thermosyphon reboiler. The procedural attachment for heat exchangers considers evidence such as whether the flow through the exchanger is being partially vaporised to determine whether the exchanger is a thermosyphon and returns TRUE or FALSE as appropriate.

The other noteworthy feature of the rules in PLS is the mechanism by which they are invoked. Unlike a standard rule language, the rules encoded in PLS are not invoked or controlled by any complex control regime or rule interpreter. Rules are invoked explicitly in PLS by other rules. When a rule is invoked, it attempts to match against every instance in the database unless it is restricted by the "class" keyword. Once the rule has traversed the database, its action is complete and PLS calls the next rule, or recalls the last one. This architecture is well suited to fast matching and data generation and also well suited to the purposes of the two reasoning tasks for which the rules are used.

The rules that infer constraints are independent of one another. No rule used for this task requires data as its antecedent that another rule derives as its consequent. Although the rules are data driven, they do not "forward chain" in the classical sense. Once every rule that infers constraints has traversed the database once, this task is complete. If a rule fails to match a particular frame when it is applied, the conclusions of subsequent rules will not affect this. Thus, there is no need to re-run the rule in case it has been enabled subsequently after failing to draw a conclusion. Note that the assumption that all antecedent data is available is safe because of the backward chaining action of the procedural attachments.

Rules are also used to implement the later phases of PLS such as group formation, calculation of elevation and plan layout. These tasks are carried out by algorithms devised specifically for each task. These algorithms are implemented in suites of rules which explicitly call one another and thereby exercise tight control over the reasoning process. In this role, the rule language is used as much as a high level procedural programming language as a knowledge representation language. The algorithm is implemented explicitly using the intrinsic control of the rule language. The rule language is ideal for this purpose because the programmer can express the interactions of the code with the database in rule preconditions and conclusions. These give the programmer access to the low level database query and update functions at an appropriate level of abstraction.

In a conventional rule language, the rule source is treated as data that are matched against the database by a rule interpreter. In PLS, the rules are invoked explicitly, and a conventional rule interpreter is not required. Thus, each rule is compiled into a LISP function. When a rule is invoked, the corresponding LISP function is called. A rule compiler reads the rule source code and generates LISP as its output. The output is then compiled by the LISP compiler to binary code. The compilation takes place outside PLS. The binary code is found to run at least two orders of magnitude faster than the rule source loaded directly and interpreted into LISP at run-time. The compiler was built by Reynolds in consultation with the author. Note that the compiler in PLS is a compiler in the classical sense. The term is often used in the context of rule languages to mean a fast matching algorithm such as RETE [Forgy 1982]. These look for changes in the facts in a system's memory to avoid matching every fact against every rule repeatedly.

Some operations in PLS's approach to layout are intrinsically "non-monotonic", that is, adding a new fact can cause an existing fact to become invalid. For example, the fact that an entity is 3 m above another no longer holds if a new fact that the entity is 4 m above the other is asserted. Although PLS follows a "least commitment" reasoning strategy which substantially eliminates back-tracking, it must draw assumptions to progress in some situations. For example, PLS must assume elevations of items while iterating to adjust the elevations of the equipment items and floors with respect to each other. Thus, PLS must operate non-monotonically at times and uses two strategies to handle this. The first strategy is exemplified by PLS's handling of the change of elevation offset. The code that calculates elevation is programmed to explicitly replace a previous offset with a new offset. That is, a new fact is not actually asserted but rather, the existing fact is modified. This is a solution to one aspect of the so-called "Frame Problem", identified by McCarthy [1969]. The Frame Problem has a number of manifestations, in this situation the problem of predicting all effects of an action or event. The solution used in PLS is similar to that used by McCarthy in his "Situation Calculus". This also included explicit knowledge to specify of the ramifications of a change to the database. The Situation Calculus exhibits the principal potential difficulty with this solution. The number of axioms needed to maintain consistency in the database far exceeds the number needed to represent the actual knowledge being reasoned about. This difficulty does not arise in PLS. In PLS, very few types of change to the database have ramifications which might need to be propagated. Thus, it is easy to specify exhaustively the knowledge that might be required to manage the ramifications.

PLS uses conventional back-tracking as its other strategy to handle non-monotonicity. In the prototype, PLS back-tracks chronologically because this is by far the simplest mechanism to implement. That is, PLS retracts all facts derived since it made the decision that it now wishes to reverse. PLS invokes back-tracking sufficiently infrequently for this grossly inefficient approach to be acceptable in a prototype. However, it is clear that a properly constituted Truth Maintenance System should be included if PLS is re-engineered for full-scale commercial use. This requirement is discussed further in Section 10.6.5.

5.3 Values that Vary with Time

In a batch process, the values of many parameters vary with time in normal operating conditions. The rate of a reaction will usually decrease as the reaction continues, for example. The values of some parameters of any process, whether batch or continuous, also vary with the operational state of the process. For example, there should be no flow in a relief header during normal operation but one would expect the flow rate to be substantial during a release. The positions of the equipment cannot change with time to suit the prevailing conditions, of course. One layout must be designed using the governing values of each parameter although the governing values of all parameters need not be taken from one consistent operating state. For example, the elevation of a vessel above its discharge pump is calculated using the liquid level when the vessel is nearly empty while the area of the vessel's wall that is wetted for heat transfer is calculated using the liquid level when the vessel is filled to its normal level.

A governing value need not be the value that has the largest impact on the layout. The governing value is the value that is used to derive a particular constraint. Indeed, values of the same attribute at different states may actually give rise to constraints that have contradictory effects on the layout. In extreme cases, it might not be possible to generate a layout that satisfies all of the critical constraints. Nonetheless, PLS records every inconsistency between constraints. The records of the inconsistencies show if critical constraints are being left unsatisfied and therefore, that the process design is infeasible.

PLS represents values that differ with time or operating state very simply. It records the values at each potentially significant state in separate slots with names that imply both the basic attribute and the state. For example, the vessel's liquid levels at the normal operating and empty states are recorded in slots called "normal-level" and "emptied-level". Although simple, this technique ideally suits PLS's needs and would be equally appropriate in an Expert System for commercial use in this domain. The governing value of an attribute depends on what the attribute is and importantly, what inference is being drawn. PLS selects the governing state for an inference by applying domain knowledge (albeit implicit). Slot names, which include the name of the state, coded into the rule manifest this knowledge explicitly. In the example of vessel liquid level, the rule to calculate the wetted area for heat transfer implies the domain knowledge that the value of liquid level at normal operating conditions is governing. The rule manifests this knowledge by retrieving the value for the liquid level from the "normal-level" slot.

Chapter 6 : Constraints in Conceptual Layout

The central role of constraints in PLS has already been introduced earlier in Section 4.1. This Chapter expands on these principles. We have already seen that two conceptually different categories of constraint are significant within PLS. "Functional Relationships" (FRs) record the spatial requirements and "Spatial Relationships" (SRs) record the relative positions of entities while the layout is developed. However, FRs can be further sub-divided into more specific categories that are meaningful within the domain. These categories are discussed in this Chapter. The persistent representation of the constraints is important and this has been done. Finally, a layout is invariably over-constrained in that mutually inconsistent spatial requirements act on the equipment items. Accordingly, FRs must be relaxed to allow a solution to be achieved. In PLS, the FRs to be relaxed are selected using a unique qualitative technique which is explained in this Chapter.

6.1 Constraints as Diadic Relationships

Many objects might be subject to a common design constraint. For example, a number of heat exchangers might all share a need for the same tube pulling facilities. This common constraint could be represented as a single polyadic FR. However, it was found during this work to be more convenient to form a set of diadic FRs, one for each pair of related objects. In particular, a polyadic FR would require a complex data structure to allow it to record separately whether the relationship between one pair is satisfied. Conversely, each diadic FR in a set refers to one pair of entities and therefore, it is clear which one relationship of the many in the set is represented by the FR.

Similarly, the position of an entity might be defined relative to two or more other entities. For example, one might wish to record that item A is 3 m North of item B and 2 m South of item C. Again, this relative position could be recorded in a single polyadic FR. However, much of the benefit of using SRs to record relative positions is lost unless the SRs are also diadic. In particular, each SR stands *in lieu* of a locally consistent set of FRs and records the relative position of a pair of entities so that minimal work is required if the relative position of one with respect to a third is changed. Clearly, these benefits are only consistent with SRs recording diadic relationships.

It is not essential to limit FRs and SRs to recording diadic relationships within the methodology developed during this work. However, the practical benefits are considerable and this has been adopted throughout PLS.

Note that a relationship can be postulated in principle that is intrinsically polyadic. This relationship records that an entity is between two others. However, in practice this relationship is not important in PLS. It has not been found to be useful as a precursor to inferring more precise relative positions at any time during this work. While the relationship could be inferred from other existing SRs, this would not increase the information available and therefore, would serve no purpose. Thus, the existence of this relationship in principle does not contravene in practice the principle expounded above.

6.2 A Taxonomy of Constraints

The important conceptual difference between FRs and SRs has been discussed in Section 4.2. It was found during this work that FRs should be further sub-divided into more specific categories. Within the conceptual layout domain, each design constraint expresses one of two spatial requirements. It may require the related entities to be close or demand their segregation. The properties and behaviour of a constraint differs markedly depending on the spatial requirement it expresses. This distinction is also important to define the way in which the constraints are manipulated by a computer system. It is an important principle of this work that

FRs are categorised to reflect this distinction. Instance of one category represent relationships which constrain entities to be close and might actually constrain their relative positions. These shall be referred to as "physical FRs" hereafter. Conversely, "segregation FRs" represent relationships which constrain entities to be distant from one another because they are mutually repelled. A third category of FR is also used extensively in this work. "Logical FRs" are not an intrinsic feature of the domain and do not record any spatial requirement directly. However, they are a powerful device to record relationships which are used to simplify the procedure of identifying clusters of strongly related entities.

Conversely, all SRs record the same information at a conceptual level and do not influence the layout of themselves because they record partial solutions rather than impose design constraints. The partial solutions are recorded as relative positions of the entities. These relative positions might be expressed initially as a direction, perhaps to record that "item A is North of item B". As constraint propagation progresses, additional detail becomes available and a distance might also be recorded, perhaps stating that "item A is 3 m North of item B". However, it is important to note that this additional information is recorded on the same SRs that record the directions. That is, there is no value in distinguishing between SRs which record direction only and those which record direction and distance. To do so would complicate the representation un-necessarily. Therefore, SRs are not sub-divided further.

The three categories of FR are reviewed below to describe their properties and features and the mechanisms used to infer the existence of their instances. The practical details of how FRs and SRs are represented are described later.

6.2.1 Physical FRs

An FR classified as "physical" represents an instance of a relationship which arises from a constraint that requires the related entities to be close or perhaps even imposes more precise spatial requirements. The constraint might represent a common need, for example relating two items which require frequent attention from the process operators. Alternatively, the constraint might represent a need for proximity for physical reasons, such as the constraint that requires two objects connected to a high voltage supply to be close to minimise the length of high voltage cable in the plant.

Instances of physical FRs are inferred readily by applying generic domain knowledge to the process data. It is appropriate to record this generic knowledge in forward chaining rules. Each of these rules recognise the causal factors of one type of physical FR and create instances of that type as their conclusion. PLS includes 42 rules to identify instances of physical FRs amongst the equipment comprising the test process. This set is not exhaustive, but presents a representative sample of the categories of physical FR and includes examples with a wide range of properties. Examples of these rules include:

> "If two heat exchangers require the same service, then a physical FR requires them to be close to minimise service pipework for economy."

or

> "If a stream contains a vapour close to its dew point, then a physical FR requires the connected equipment items to be close to prevent condensation in the pipework."

Clearly, these rules exemplify the general principle as well as specific knowledge embodied in PLS's knowledge base.

6.2.2 Segregation FRs

"Segregation FRs" represent constraints that require the related objects to be positioned apart from one another. Almost all segregation FRs that can be envisaged appear to arise to enhance safety. Certainly, all examples in the test process do. As with physical FRs, segregation FRs are also readily identified by applying generic domain knowledge to the process data. Again, it is appropriate to express this knowledge in forward chaining rules. An example of one of these is:

"If an object is a potential ignition source and another object contains flammable material in bulk, then a segregation FR requires the objects to be distant to prevent ignition of a leak."

6.2.3 Logical FRs

"Logical FRs" are a very powerful abstraction that can be used to identify clusters of strongly related entities within a constraint network. In this work, they are used as an essential element of the procedure to form "logical groups", that is, groups that correspond to process units or similar. Indeed, it may be beneficial to read the description of the principles and requirements of logical group formation in Section 8.2.2 in parallel with this Section.

Logical FRs do not record a constraint that is significant directly in the terms of the domain. In layout, the constraints that are directly significant require either proximity or segregation. Rather, they are derived from the more fundamental constraints, in this case expressed in physical and segregation FRs. Logical FRs act as meta-knowledge to simplify the satisfaction of the fundamental constraints. It is meaningless to consider any direct benefit that accrues when a logical FR is satisfied. The logical FR merely re-expresses the requirements that more fundamental constraints impose. That is, benefit accrues if a logical FRs is satisfied because the fundamental constraints that it reflects are also satisfied as a side effect. These properties are clearly markedly different to those of the fundamental constraints. Therefore, it is essential that logical FRs are classified separately from these fundamental constraints, in this work the physical or segregation FRs.

The following formal definition of a cluster of entities has been adopted in this work. This can also be read as a formal expression of the principle underlying logical groups. Each cluster of strong constraints is circumscribed by a local minimum in the strengths of the physical FRs. A key item in each cluster is positioned at a local maximum and the strength of the FRs declines with distance from the key item. Effectively, the gradient of the strength of the FRs is positive from any member towards the key item. The increase in strength of the FRs towards the key item pulls each member into the cluster.

Logical FRs record in which direction from any entity the gradient of the strength of the FRs is greatest. The direction is expressed in the form of a relationship between the entity and an adjacent entity in the constraint network towards which the gradient is greatest. This establishes the principle that logical FRs are notionally directed.

Logical FRs are such a powerful abstraction because they allow very simple algorithms for finding the members of the clusters of constraints. While simple, these algorithms embody all the control and sophistication required to identify complex combinations of clusters, such as autonomous clusters within other, more expansive clusters. That is, logical FRs re-express the problem in a manner that is highly amenable to computation. However, logical FRs are very easy to identify and therefore, the cost of this simplification is low. Indeed, it was discovered during this work that logical FRs relating individual items of equipment can be identified using "compiled" knowledge embodied in relatively simple rules. Compiled knowledge captures the expected outcome of a potentially complex chain of reasoning and executes it in a single step. The details of the chain of reasoning are lost or might even be unknown. For example, one of the rules that identifies logical FRs states:

> "A transfer pump should be grouped with the item from which it draws its feed."

This rule does not explicate any notion of why this should be so but forms logical FRs where they would be expected nonetheless. A set of fifteen such rules was developed and encoded in PLS. This set is sufficiently broad to identify every instance of logical FR relating equipment items in each of three diverse plants, the test process and two others which were analyzed as paper exercises.

A general analytical rule can also be used to identify logical FRs. This rule arises directly from the underlying principle of logical FRs. It states that an object should be grouped with a second object selected from amongst all those to which the object under consideration is connected to which it is most strongly attracted by physical FRs. The strength of attraction is measured by comparing the physical FRs running parallel to the streams upstream and downstream of the object being assessed. However, the logical FR is only created if the second object is more strongly attracted to a third object to which it is connected. This condition is not met in two situations. The first arises if the second object is an ancillary of the object under consideration. In this case, the logical FR will be created when the pair are considered the other way round. The second arises if separate groups should be formed around each object. In this case, no logical FR should be formed at all.

It is important to note that this analytical rule is markedly simpler than a procedure to identify clusters of entities working directly from an analysis of the constraint network as a whole. Furthermore, the analytical rule is used almost exclusively to identify logical FRs between groups as they are formed. No compiled knowledge could be elicited that would identify these logical FRs. In the main, groups tend to have many connections. Typically, more than one of these exhibits properties that would suggest that a logical FR might be formed. The analytical rule distinguishes between these candidates. However, the rules embodying compiled knowledge can identify the majority of logical FRs even though the set of rules is restricted to finding FRs relating equipment items. Equipment items are far more numerous and more inter-connected than groups. Thus, this set of rules covers all of the by far most frequent situations in which logical FRs must be identified.

More than one process stream flows through some objects. For example, a heat exchanger might transfer heat between two process streams. The principle is adopted that one logical FR should be formed for each flow. Thus, for example, two logical FRs are formed for a heat exchanger.

6.3 Representing Constraints

The persistent representation of constraints is an important principle of PLS. It allows FRs to be inspected to provide an audit of the layout, an important benefit to a system user. It also greatly enhances efficiency compared to re-inferring the constraints each time they are referred to during spatial synthesis.

Much like streams, each constraint has properties and attributes of its own. Thus, like streams, a structured representation is clearly appropriate. The structured representation is provided in PLS by representing constraints as frames because frames are particularly convenient. The constraints are reasoned about extensively during the spatial synthesis phases of generating a layout. The rule language is designed to work in conjunction with frames. Therefore, it can be applied directly to the frames representing the constraints and therefore, can be used unmodified to encode the spatial synthesis knowledge.

In practice, each category of FR has a corresponding class of frame defined for it. A further class is defined for all SR frames. The classes are used passively to partition the FR instances in the database. Thereby, the spatial synthesis algorithms can then retrieve just the instances in the categories that are appropriate to the task in hand. An example of an instance frame for a physical FR which constrains the reactors R104 and R108 to be close to reduce the length of high voltage cable required to connect their motors follows.

```
{F-1    [is-a physical-FR]
        [type economic]
        [name cable-length]
        [criterion 4kV]
        [min-distance 2]
        [max-distance 10]
        [branches
                {F-2    [is-a fr-branch]
                        [object
                                {S-?    [name R104]
                                        :
                                        } ]
                        [feature
                                {S-?    [name R104-M1]
                                        :
                                        } ]
                        [groups
                                {S-?    [name G8]
                                        :
                                        } ]
                        [elevation-end top]}
                {F-3    [is-a fr-branch]
                        [object
                                {S-?    [name R108]
                                        :
                                        } ]
                        [feature
                                {S-?    [name R108-M1]
                                        :
                                        } ]
                        [groups
                                {S-?    [name G9]
                                        :
                                        } ]
                        [elevation-end bottom]}
                ]}
```

This frame is identical in principle to those presented previously. It comprises slots which record atomic values, some numeric and others symbolic, and other slots which have frames as their values. Instance frames for the other categories of constraint are very similar and will not be exemplified separately.

All instances of each type of physical and segregation FRs have specific properties which arise from its nature and causal factors. Some properties are generic to broad categories of relationship, others to a very specific set. The example FR relates two objects driven by high power motors, expressing their mutual attraction to minimise high voltage cabling. The relationship has the general property of increasing plant economy if satisfied, a more specific property of minimising the length of cable, and the very specific property of minimising the length of 4 kV cable. These properties are important because they determine the phases of spatial synthesis an FR should influence, determine an FR's need for satisfaction and identify objects with similar needs which can be met by positioning them close to one another. The causal factors of an FR are recorded in three slots, written onto it when it is created, called "type", "name" and "criterion". These slots express the causal factors increasingly specifically. In this example, the "type" slot of an FR instance has "economy" as its value, the "name" slot has "cable-length" as its value and the "criterion" slot has "4kV" as its value. Three slots are used so that generic properties can be explicitly recorded on the FR as well as very specific properties. When PLS needs to consider the generic properties of an FR instance, it can read the information directly from the frame. If the FR frames only recorded the most specific information, PLS would be forced to infer the implicit generic properties from the explicit specific properties. The explicit records of the more generic information also indicate clearly which FRs are thought to be similar in general by the system builder. This conveys important information to someone inspecting the knowledge base.

On a logical FR, all three of these slots are given the value "logical". The causes and properties of all logical FRs are identical so there is no meaning in any distinction between "types" of logical FR.

Within the domain, some FRs record more precise spatial constraints than proximity or separation. In this example, it is assumed that the reactors are constrained to be at least 2 m apart and at most 10 m apart. These values are recorded in the "min-distance" and "max-distance" slots. It is doubtful whether this FR would impose such a precise constraint but the slots have been included to exemplify this feature. Other FRs constrain the elevation offset between the related items. This is recorded in the "min-elevation" and "max-elevation" slots. In some cases, FRs constrain both the horizontal and vertical separation of the items. The frames for these FRs include whatever combination of these four slots is appropriate. Similarly, the minimum and maximum separations of the entities related by an SR must be recorded upon it at some time during constraint propagation. Thus, frames for SRs also include these slots. They might also include a "direction" slot which takes as its value the name of one of the cardinal directions of the compass to record the direction of the SR relative to the local coordinate system of the group of which the related entities are members. Some segregation FR frames record a minimum separation of the entities considered adequate to eliminate the hazard if this can be calculated or read from standard Tables [*eg* Mecklenburgh 1985, Appendix C]. Other segregation FRs only express coarse spatial requirements and this is inappropriate. No segregation FR imposes a maximum separation - any separation greater than the minimum, if specified, satisfies the FR. Logical FR frames never record any precise spatial information because this would be meaningless.

The frames for constraints include two "branch" frames in PLS - similar to those for streams. Each branch frame holds one of the entities that the constraint relates. Because constraints are always diadic, each constraint frame holds two branch frames. In this example, the FR instance holds the branch frames F-2 and F-3, which hold the frames for R104 and R108 respectively. Slots are written into the branch frames to record information germane to each of the related entities. In this example, the branch frames include slots called "elevation-end" which take the values "top" and "bottom" respectively. This is typical of an FR or SR that relates to elevation, although it is included in the example above purely to demonstrate the principle. The branch frames of logical FRs include a slot called "logical-end". This slot is given the value "master" on the branch frame that records the object towards which the gradient of the strength of attraction is positive. The slot is given the value "ancillary" on the branch frame that records the object from which the gradient is positive. These terms are mnemonic and reflect the frequent situation in which the gradient slopes upwards from an ancillary towards the "master" that it serves.

Different names are used for these slots so that the spatial synthesis algorithms can identify the axis in which the constraint acts. For example, while the branch frames on an elevation SR include slots called "elevation-end", the branch frames on SRs formed during plan layout include slots called "plan-end". These take "to" and "from" as their values to specify relative to which entity the direction is measured. Thereby, each spatial synthesis algorithm can readily identify the constraints that it must consider.

Even though all constraints are diadic in PLS, FRs influence the positions of the groups of which the entities are members. Thus, in practice, when PLS collects an entity, whether an equipment item or previously formed group, into a more expansive group, PLS updates the appropriate branch frames of FRs to show that they also relate the more expansive group. In this example, the FR relates R104 and R108 which are members of groups G8 and G9 respectively. These groups are recorded in the "groups" slots of the respective branch frames.

Some physical FRs arise from the properties of specific physical components of objects. In this example, the constraint arises from the need to minimise the length of the cable length between the motors fitted to each reactor. This constrains the positions of the reactors and also their orientation so that the motors are as close as possible. Thus, it is essential that physical FRs can record the component that gives rise to the constraint alongside its parent object to model adequately the domain. The component is written into the "feature" slot of the branch frame alongside the object in the "object" slot. In this example, both branch frames have "feature" slots which have the instance frames for the motors as their values. The branch frames of SRs can also include a "feature" slot.

Segregation FRs do not impose constraints of enough precision to warrant this level of detail.

6.4 Representing Relative FR Importance

Previously it has been stated that a layout is invariably over-constrained. The set of all FRs that impose spatial requirements invariably contains mutual inconsistencies. Accordingly, some FRs must be selected to be relaxed to leave a set of FRs which are all consistent. This selection can only be made if the importance of satisfying the FR instances can be compared. In practice, the importance of FRs varies widely in conceptual layout. Some must be satisfied if the plant is to function at all. Some FRs are of such little importance that they are satisfied in a design only if it is opportune to do so. The plant might be more economical or easier to operate if others are satisfied. Value judgements are required to select which FRs should be satisfied.

In some domains, it is appropriate to generate the solution by optimising a single parameter against an objective function. In these domains, it is then appropriate to represent and manipulate the importance of constraints numerically. For example, ISIS [Fox 1983, 1986] and later CONSYST [Silverstein 1990], were constructed to plan and schedule production processes. Some constraints represent dependencies between processes where one process must be completed before another may start. These cannot be relaxed and therefore their importance need not be represented. The need to satisfy them may be taken as read. All other constraints effectively represent the financial impact of scheduling decisions and trade-offs. The importance of these may be accurately quantified against a single coherent metric, in this case effectively financial. Numerical values are assigned to classes of constraint to represent their importance. These values are inherited by their instances. The solution is generated by searching to maximise these values, treating problem solving as single parameter optimisation. This matches the objective in scheduling of maximising profitability of a facility.

In conceptual layout, the benefit accrued from satisfying many FRs might also be directly quantified. For example, an engineer might estimate the additional financial cost incurred if an FR were left unsatisfied. In many cases, this would be very natural and accurate. If this were true of all FRs, PLS could determine numerically which FR instances are to be satisfied. This is feasible even given the large number of FRs to be considered. For example, an algorithm has been developed which determines the optimal number and positions of members in bridge structures [Mistree 1981]. This problem involves literally many thousands of variables and very complex relationships between them. However, it is not possible to quantify the benefit of satisfying many other FRs. Some may be quantified in principle but the data are lacking. Others have no natural quantification.

The decision to stack heat exchangers exemplifies a lack of data. It is widely held that there is an economic benefit in this. However, analysis of the readily apparent economic factors shows that they nullify each other [Bush 1972]. It appears that a real but "hidden" factor exists which the engineer takes into consideration but which could be missed by an optimisation approach.

The effect of transport lag caused by long piping runs on the control of the process exemplifies a constraint that is highly important and yet not amenable to quantification. A domain expert would have to estimate a value to incorporate into the optimisation. However, estimated values of importance are likely to be imprecise and unreliable. Generally, if a subject is asked to order stimuli according to the magnitude of a given stimulus dimension, the subject will only be able to use five to nine ranks effectively [Hink 1987]. This is too imprecise for input data to an optimisation function given that the FRs vary so widely in importance. The bands are too broad.

The accuracy of subjective values of importance should also be questioned. The value is in effect the probability of the FR proving to be more important than another in a comparison. Studies show that subjects over-estimate low objective probabilities and under-estimate high objective probabilities [Lichtenstein 1977]. The bias tends to decrease in easy tasks and experts tend to show less bias than non-experts [Lichtenstein 1982]. However, weighting FRs in layout would not seem to be an easy task and even experienced layout engineers would not be expert in this. It is not the way they work.

The principle has been adopted in this work that it is unsatisfactory to record the relative importance of constraints numerically in this domain. The previous argument shows that it is difficult to ascribe accurately absolute numeric values of importance to FRs. However, it is relatively easy for a domain expert to state which of two different FRs is more important. For example, layout engineers would agree that an FR relating two objects connected by a large bore pipe should be satisfied at the expense of an FR relating two objects connected by a small bore pipe. This ability is exploited to support this principle and provide a non-numeric technique to determine which FRs should be satisfied in preference to others. This technique is introduced in a simplified form here. Layout engineers were asked during knowledge elicitation to compare a pair of FRs and state which of the FRs is more important. The relative importance was tabulated in PLS's knowledge base. The procedure was repeated for many other pairs of FRs. In practice, no relative importance could be decided for some pairs of FRs. Thus, a partial ordering was achieved. The table which recorded the engineers' preferences is therefore named the Partial Ordering Table or PO Table.

When an inconsistency between FRs is identified, the relative positions of the FRs on the PO Table shows which to satisfy. If no preference is recorded directly for a specific pair of FRs, it is assumed that relative importance is transitive and a series of FRs which can be compared is established between the entries for the FRs. While it has never been proved that qualitative relationships are transitive, intuitively the assumption seems completely reasonable.

The technique used in PLS is more sophisticated than the simplified form described above. Inconsistent FRs may occur in sets for two reasons. Firstly, it is likely that any pair of entities will be related by more than one FR. This was found to be universally true in the test process. Thus, whenever an entity is involved in an inconsistency, typically more than one FR will need to be relaxed. Secondly, it is often the case that an inconsistency can only be resolved by relaxing FRs that relate a number of entities. For example, during group formation, an entity is assigned to the group to whose members as a whole it is most strongly attracted. All the members might contribute FRs to the set which attracts the entity. In a further example, while elevation is calculated or plan layout derived, an inconsistency might need to be resolved by moving more than one entity. For example, assume that item A is constrained to be 4 m North of item B and 3 m South of both items C and D. For the sake of this example, assume that items B, C and D are all fixed. Thus, the inconsistency can only be resolved by relaxing either the FRs between item A and B or the FRs relating item A to both items C and D.

The methodology developed during this work adopts the principle that each set of FRs that can be satisfied concurrently are collected into a so-called "link set". It is these link sets that are actually compared on the PO table. No other system that arbitrates contradictions between constraints non-numerically is known that employs an analogous technique or even a technique that attempts to meet similar objectives. The most usual technique is that due to Marcus [1986], exemplified more recently in PLAKON [Syska 1988].

PLAKON implements two strategies for arbitrating contradictions amongst constraints. "Relaxibility factors" (*sic*) are assigned to all classes of constraints. In global relaxation mode, when a contradiction is identified, PLAKON successively disables constraints with the highest value of relaxibility from the problem until a consistent solution can be achieved. Once PLAKON, in global relaxation mode, has identified one constraint that cannot be satisfied, it modifies the constraint network globally until a solution is possible. Constraints might be disabled that do not contribute to the contradiction. PLAKON does not continue to develop the solution in the majority of the problem space once further progress becomes impossible at one point. That is, PLAKON cannot trade off constraints to achieve the best global solution. PLAKON can also operate in its local relaxation mode. In this, constraints are organised into so-called "constraint bundles". These are sets of constraints that all represent the same domain dependency but restrict values more or less severely. In the case of a conflict, the conflicting constraint is swapped for a less restrictive alternative from the same bundle. This leads PLAKON to develop a solution which comprises many local optima rather than one global optimum.

It is important to note that even a numeric technique would be likely to encounter difficulties in the conceptual layout domain. Consider, for example, the FRs attracting pumps toward one another into a pump bay. It is common practice to install perhaps eight or ten pumps together in a pump bay. Smaller pump bays are observed much less frequently and it is most unlikely that two pumps would be installed in a bay. It appears that a "critical mass" must be reached before the decision to collect the pumps together is taken. The relationship between number of pumps and likelihood of their being collected appears to be highly non-linear. By implication, the relationship by which the importance of satisfying the FRs relating the pumps is derived from the number of FRs is also highly non-linear. A relationship like this is likely to prove difficult to derive accurately and also difficult to compute. This difficulty is circumvented entirely by the link set technique.

The link set concept is used in PLS as follows. When the PO Table was elicited, pairs of prototypical sets of FRs were compared. These prototype sets specify either a number of instances of the same type of FR or a specific combination of instances of a number of types. The PO Table includes a number of prototype sets comprising the same type of FR but in differing numbers. These express the (usually increasing) importance of satisfying increasing numbers of instances of the same FR. Sets of one FR are also considered so that solitary FRs can still be compared. When an inconsistency is detected between FRs, those FRs involved in the inconsistency are formed into link sets. These link sets are then compared with the prototype sets on the PO Table. The relative importance of satisfying each link set is retrieved from the PO Table exactly as described above for single FRs. Thereby, importance of satisfying link sets of various combinations of FR can be compared directly because this is defined explicitly by the position of the prototype link set on the PO Table. Furthermore, there is no need to derive a relationship between the composition of a link set and the importance of satisfying it.

The criteria by which FRs are selected for inclusion in a link set depend on the problem solving phase in which the inconsistency is detected and the detailed nature of the inconsistency. These criteria are essential adjuncts to the technique and are discussed in the appropriate following Chapters. However, the underlying principle is that the algorithm active at the time is responsible for applying appropriate domain knowledge which records these criteria.

The need to satisfy an FR depends on its causal factors which are recorded on each FR instance in its "type", "name" and "criterion" slots in PLS. The prototype link sets on the PO Table are defined in terms of these values also. A link set is considered to match a prototype set when the three values for one element of the prototype set correspond to the values read from an FR instance in the link set. Wild card values can also be specified in the entries for the prototype link sets. These match any value read from an FR instance. For example, an entry in a prototype set might require an FR instance to have "economic" as the value of its "type" slot for it to match but wild card values for the "name" and "criterion" slots allow any values of these to match. Any FR instance of type "economic" matches the entry in the prototype set. Wild card values can be specified for the "criterion" slot only or the "name" and "criterion" slots together. PLS attempts to match FR instances in a link set against fully specified entries in the prototype sets, then attempts to match FRs against prototypes containing a wild card for the "criterion" slot, and finally against prototypes with wild cards for both the "name" and "criterion" slots. This allows generic prototype link sets to be specified on the PO Table to match any FRs which arises from similar considerations but which differ in detail. For example, the heat loss per unit length of two pipes could be the same if one is small bore and at very high temperature and the other larger bore but cooler. The FRs that relate objects connected by either type of pipe would have the same values for their "type" and "name" slots but a different value for their "criterion" slots. A generic prototype link set with a wild card for the value of the "criterion" slot would match either FR and show that they are equally important to satisfy.

Generic prototype sets which specify the value of the FR's "type" slot but include wild cards for the values of the other slots are used in the PO Table built for the experimental work. These prototype sets establish default levels of importance for broad categories of FR. Additional prototype sets which specify more closely the FRs that they match are added to the PO Table to indicate that specific FRs of this general type are considered to be more important than the default.

In principle, a link set might arise which is important to satisfy but which comprised an unforseen combination of FRs. No prototype set would exist on the PO Table which adequately represents the importance of this combination so the importance would be under-estimated. In practice, a feature of the domain mitigates this difficulty. It is found that the prototype sets that are most important to satisfy contain a few important FRs rather than a large number or particularly synergistic combination of individually unimportant FRs. That is, it appears that a link set has to contain important FRs for the link set as a whole to be important. Therefore, the individual importance of the members of an important link set "protect" it from being left unsatisfied because its obscure combination cannot be matched against the PO Table.

In PLS, a separate frame is created to represent each inconsistency between FRs. These frames record the two link sets formed from the inconsistent FRs and the link set chosen as more important after their comparison on the PO Table. These frames are retained in the database to provide the audit records of the inconsistencies. The PO Table implies the design assumptions which lead to the particular choices of which FRs should be satisfied. Thus, it forms an essential element of any audit for validating a layout produced by PLS. Therefore, the PO Table is copied into the database to be retained for the database's lifetime. This ensures that all information needed to validate the design remains available.

6.4.1 Indeterminate Comparisons

The PO Table might not record which of two link sets should be satisfied. When some pairs of link sets are compared during elicitation of the PO Table, the engineer might express no preference as to which should be satisfied. Sometimes, the link sets might be genuinely equal in importance, other times it might not be meaningful to compare them as they express disparate and incomparable factors. The PO Table is constructed with the prototypes sets corresponding to these link sets positioned "in parallel". When link sets that correspond to these prototype sets are compared on the PO Table, it is not possible to draw a conclusion. In the methodology, one link set is selected arbitrarily as the more important. In principle, each of the link sets could be selected in turn and a layout generated for each. As a simplification to speed development of PLS, it only develops the first solution to completion. This issue is discussed further in Section 10.6.5. The frame that is created to record the inconsistency is marked with the fact that the inconsistency was resolved arbitrarily so it can be identified easily if the audit is reported. The methodology requires that the PO Table is modified dynamically to reflect the preference implied by the arbitrary selection. This ensures that the same choice is made should similar link sets be compared later. This maintains consistency throughout the solution. In PLS, the modifications are applied to the copy of the PO Table loaded into the database for this layout and hence, are not transmitted to other users. Design then continues until a solution is reached.

6.4.2 Segregation FRs

Segregation FRs are satisfied when the related objects are positioned sufficiently far from one another. A pair of objects might be related by a segregation FR and also by physical FRs which demand their proximity. Thus, it is a feature of the domain that FRs relating a pair of objects can be inconsistent with themselves rather than with FRs relating one of the pair of objects to a third. For example, consider a distillation unit separating crude oil. The reboiler in this unit might well be fired to vaporise the very involatile column bottoms. A number of physical FRs demand the proximity of the column and reboiler, for example to minimise heat loss from the vapour return line and to minimise pressure drop in the line. A distillation column contains a large volume of flammable vapour which would be ignited by the fired reboiler were it to leak. A segregation FR demands that the column and reboiler should be distant to minimise the risk of a major fire. The physical FRs are clearly inconsistent with the segregation FR even though all of the FRs relate the same objects. In a situation such as this, either all of the physical FRs or the segregation FR should be satisfied.

In these cases, two link sets are formed from sub-sets of the FRs which relate the pairs of entities. The physical FRs are collected into one, the segregation FR (or FRs if there is more than one) into the other. The two link sets are then compared on the PO Table to determine which is more important to satisfy. Prototype sets comprising segregation FRs are added to the PO Table in exactly the same way as those comprising physical FRs. There is no concept of "inverse" importance for the segregation FR prototype sets. The segregation FRs act in what might be termed loosely an inverse of the action of physical FRs. However, the PO Table records which link set will be allowed to act, not the action itself. Thus, if the segregation FR link set is more important than the physical FR link set, the former is placed higher in the PO Table. In the example above, PLS would form one link set comprising all of the physical FRs and a second with the segregation FR as its member. It would match the two link sets against the PO Table. For the sake of the example, assume that the prototype link set for the segregation FR was placed above the prototype link set for the physical FRs on the PO Table during elicitation. This implies that the segregation FR is more important to satisfy than the physical FRs, and the segregation FR would be selected for satisfaction. Having made this decision, PLS would then, and only then, acknowledge the spatial significance of the segregation FR and position the column and reboiler apart from one another.

Chapter 7 : Calculating Elevation

In a process plant, the elevations of the equipment items are set principally to satisfy the minimum and maximum height differences that must be allowed between the items if they are to function. The engineer performs complex but routine calculations to establish the allowable minimum and maximum elevation differences. This is one of the major time-consuming elements of the manual procedure for layout. The calculations require information from the process design, physical property data and preliminary mechanical designs of some of the items, especially the positions of their nozzles. The engineer spends as much time marshalling these data as performing the calculations themselves. PLS automates this time-consuming task and calculates the elevation of each item of equipment in the plant and positions floors at the appropriate elevation.

Two factors dominate the engineer's perception of calculating equipment elevation. Process flow causes constraints on the relative elevations of equipment items. Many of these constraints arise to take advantage of gravity flow. These constrain the discharge point of the feeding item to be above the inlet of the fed item. That is, these constraints do not require any precise elevation offsets. Some constraints demand relative elevations of the items to ensure the pressure at a point is sufficient. For example, the pressure at a pump's inlet must be sufficient to meet its NPSH requirements and the pressure at a control valve in a gravity reflux line must be sufficient so that the valve operates in its linear region. Other constraints demand relative elevations to maintain a differential head between two items, such as between a distillation column and a thermosyphon reboiler. In general, the relationships which arise from process flow push items upwards from the items below. However, constraints which specify a narrow band of elevation offsets will "pull" the lower item upwards to achieve the permitted maximum offset if the upper item cannot be lowered. For example, the minimum elevation of the column above the reboiler is calculated to provide sufficient differential head so the thermosyphon effect will drive circulation. However, if the column is raised to satisfy other elevation requirements, the reboiler must also be raised so that the pressure drop in the pipework of the reboiler loop does not become excessive.

The second dominant factor is the desire to minimise the elevation of every item of equipment. The cost of a structure to support a plant increases rapidly with its height. However, the constraints which arise from process factors generally drive items upwards. An engineer adopts the strategy of positioning all items at the minimum elevation at which the process requirements are satisfied to minimise structural costs. PLS also adopts this strategy.

The notion of grade itself is as important as the other two factors in a rigorous treatment of elevation calculation, although it is taken for granted by engineers. Grade is important for two reasons. Firstly, the elevation calculation is under-constrained unless grade is acknowledged. Without grade, relative elevations can be calculated correctly but there is no datum against which absolute elevations can be calculated. Secondly, the process elevation constraints typically break into a number of separate chains. In the test process, the reactor units and weighing tanks formed one chain, the filtration and drying units a second and the distillation unit a third. All chains include grade, however, which unifies them.

Accordingly, PLS explicitly represents "grade" as a frame. The elevation code can act upon the "grade" frame without distinction between it and the frames for equipment items. The constraints which relate the "grade" frame anchor the constraint network to a fixed point. PLS creates an FR to relate every equipment item to grade when it infers the FRs during its pre-processing phase. PLS does not assume that the origin of an equipment item is on its base. It is convenient to permit the knowledge engineer to choose the origin for each type of equipment. In some cases, the bottom of an equipment item must be raised above ground level. For example, pumps are invariably installed on low plinths and columns are mounted on skirts to leave space for pipework. In effect, these items require support below their lowest perimeter rather than flush with it. The minimum elevations of the origins of these items above ground level are set so that appropriate space is left below the item. These FRs also serve to balance the effect of other FRs which might cause an item to be lifted. Because of these FRs, PLS lifts an otherwise unsupported or inaccessible item onto a higher floor after comparing the cost and the benefit rather than as an easy solution. The values written into the "type", "name" and "criterion" slots of the FRs to grade reflect the cost of lifting the item above the minimum so that the comparison can be made on the PO Table.

Any item which is not at grade must be supported by the plant's floors or by local structures built upon the floors. Access must also be provided to the items for their operation and maintenance from either a floor or a local platform. In general, floors are preferred for both support and access. Local structures complicate the structural steelwork of the plant and therefore cause the plant's capital cost to increase. They also clutter the plant. This impedes removal of equipment for maintenance and convolutes escape routes in the event of an emergency. Platforms are only positively encouraged as access to tall items which protrude above the top of the structure. For example, manholes in the side of tall columns are often reached from platforms suspended from the columns themselves. Accordingly, floors are positioned at a height were they can support and provide access to the most items. As a consequence of this, floors cannot be positioned until the approximate heights of the items are known. The provisional elevations of the items constrain the elevations of the floors more than *vice versa* and as few floors are created as necessary. Once the floors have been created, the elevations of some items are adjusted to make best use of the floors. During this adjustment, some items might need to be lifted to maintain process relationships. This procedure is standard practice for the layout engineer. PLS also embodies it and calculates elevations in two phases. Initially, it calculates elevations for the items to satisfy process flow constraints. Thereafter, PLS assesses the heights at which support and access is required, introduces floors at these heights and makes any necessary adjustments.

In PLS, FRs record the required elevation offsets between specific features of the equipment, such as manholes or nozzles, rather than between the origins of the equipment items. The FRs record the features in the "feature" slots of their branch frames. Other FRs record the elevation offsets of the features of the equipment items. Many features can be adjusted with respect to their item to best suit the layout. For example, column side manholes should be spaced regularly, but it is not critical exactly which tray is accessed by each. The manholes can be adjusted somewhat so that they align with floors provided for other purposes. PLS's representation reflects the freedom enjoyed by the layout engineer and allows sufficient flexibility so that PLS can assign an elevation to each feature independently. This principle applies more to those features which are significant when PLS determines the heights at which floors are to be positioned. However, it is carried through to those FRs which record process constraints to give a unified and consistent model.

7.1 Propagating Process Constraints

All constraints which arise from process flows cause items to be elevated. The effects of these constraints are intrinsically undesirable because they can only cause structural costs to increase. The cheapest elevation configuration arises when the fewest of these constraints are imposed. In general, a layout engineer only considers elevation constraints arising from process flows if they must be satisfied for the process to function. No benefit can accrue from satisfying any others so they are neglected. This philosophy is adopted in PLS. FRs are only formed to record process elevation constraints that impose critical elevation offsets. A ramification of this is that elevation FRs cannot be relaxed if PLS detects an inconsistency when the constraints are propagated. This occurs if the FRs governing the offset of a pair of objects are incompatible or the height of an item is defined by two or more chains of SRs which do not converge at a consistent height. It shows that the process cannot be laid out. This gives rise to PLS's very simple procedure to propagate elevation constraints that arise from process flow.

PLS propagates the elevation constraints in two steps. The first establishes the minimum and maximum elevation differences between pairs of features to meet the most stringent FRs that relate them. An SR is created to record the minimum and maximum elevation differences allowed by the FRs. During this step, PLS checks that the elevation FRs relating the features are locally consistent. In the second step, PLS propagates the locally consistent SRs. It infers new SRs that record the elevation differences between pairs of features that are not directly related by elevation FRs. PLS first propagates the SRs that record the elevation of features related to grade. This fixes the elevations of these features against a known point. PLS then propagates the SRs from these newly fixed features, and so on. It continues until an SR has been created to record explicitly the elevation difference between each combination of items which are indirectly related by existing SRs. Davis [1987] cautions us that constraint systems that reason about differences can spend a lot of time deriving information that has no practical value. In temporal reasoning, for example, there is no point maintaining the value of the difference between the date of a contemporary conference and the date of Julius Caesar's death then reducing this difference by one week when we receive more precise information about the date of the conference! However, PLS is likely to make use at some time of a high proportion of the elevation SRs that it infers. Whenever PLS considers adjusting the elevation of an item to suit a floor, it needs to identify all items which must be moved as a consequence. The elevations of many items depend on that of a key item, which is typically one of those which is lower in the plant. It is likely that PLS will consider moving each of these key items at some time while positioning floors. Thus, this indiscriminate approach to propagating the SRs is appropriate in this domain.

The Algorithm for Local Consistency of Elevation FRs

1. PLS searches for a physical FR in its database that imposes an elevation offset of a pair of features of equipment items.

2. PLS retrieves the SR that records the elevation offset of this pair of features. PLS updates the SR's "min-elevation" and "max-elevation" slots if the values specified on the FR are more restrictive than those recorded on the SR. If the SR has not yet been formed, then PLS creates it and sets its "min-elevation", "max-elevation" or both slots to the values read from the FR.
3. PLS iterates around steps 1 and 2 until it has considered every physical FR in its database.

The Algorithm for Global Consistency of Elevation SRs

1. PLS searches for a feature with an elevation offset specified relative to grade. This feature will be referred to as the "base feature". PLS sets accumulators of the minimum and maximum absolute elevations of features to the values read from the SR's "min-elevation" and "max-elevation" slots.
2. PLS searches for an SR which records the elevation offset of another feature above the base feature. This will be referred to as the "current feature". PLS updates the accumulators by adding the values read from the "min-elevation" and "max-elevation" slots of this new SR.
3. PLS searches for an SR that records the elevation difference between the base feature and the current feature. PLS updates this SR's "min-elevation" or "max-elevation" slots with the values of the accumulators if these are more restrictive than the current values. If the SR has not yet been formed, then PLS creates it and sets its "min-elevation", "max-elevation" or both slots to the values of the accumulators.
4. PLS recursively invokes step 2 with the same base feature as in step 1 but with other current features which are progressively further up the chain of SRs.
5. PLS iteratively invokes step 2 for each SR that records elevation offsets of features above the current base feature. The recursion inside the iteration creates an SR between grade and every feature which is above the original base feature.

6. PLS resets the base feature to a feature above the original base feature, resets the accumulators and repeats step 2. PLS creates SRs between every pair of features in the chain by repeating step 6.
7. PLS repeats step 1 for a new base feature.

If the local consistency algorithm encounters an FR which specifies a minimum offset which exceeds the maximum offset written onto the SR or an FR which specifies a maximum offset less than the minimum recorded of the SR, PLS has detected a local inconsistency amongst the FRs. Similarly, if the global consistency algorithm calculates a minimum elevation difference between two features that exceeds the maximum difference recorded on the SR or calculates a maximum difference that is less than the recorded minimum, PLS has detected a global inconsistency. In either case, no consistent solution exists. PLS is programmed to stop and report this to the user. The user must then modify the process design to eliminate the cause of the inconsistency.

7.2 Positioning Floors

The floors in a process plant support the equipment items, provide access to the items but can also foul parts of the equipment or its pipework during operation or maintenance. The elevation with respect to an equipment item at which a floor must be positioned to provide access or support, or at which a floor must not be positioned because it fouls a feature of the item, impose design constraints on the heights of the floors themselves. These constraints nearly always specify bands of elevation differences. Some of these bands specify fairly specific distances. For example, the centre-line of a manhole in the side of a column should be 0.75 m to 1.2 m above the access platform. This enables a fitter to step through the manhole rather than crawl or climb into the column. Similarly, if a vessel is supported from below, the bottom of the vessel should be a sufficient distance above the floor so that the pipework connected to its bottom nozzle does not foul the floor. However, the distance should not be increased above this because the vessel would require a more substantial support skirt if it were. Other bands are wider. For example, it should be possible to reach any part of the tube bundle of a shell and tube heat exchanger from a floor when the tube bundle is withdrawn for maintenance. Similarly, lugs can be positioned anywhere on the side of a vessel to support it. The support constraint for this vessel is met if a floor is positioned anywhere between its top and bottom. The lugs are positioned once the relative elevation of the vessel and floor is known.

Any floor will satisfy these design constraints. The constraints demand that an entity provides the function of a floor at a particular height rather than defining a relationship between an item and a particular instance of a floor. PLS reflects this and forms FRs which record the height difference between a notional or generic floor and the equipment items. PLS selects the most appropriate real floor to replace the generic floor in each FR once the real floors have been introduced. In the analysis, "grade" is treated as a floor, albeit with some special properties that reflect the domain.

Every item of equipment in the process imposes at least one constraint on the position of the floors, because every item must be supported. Many items actually impose a number of constraints. For example, each reactor in the test process required support somewhere up its height, access to the manhole in its upper end, access to the agitator motor mounted on its top and clearance below for the discharge pipework. Thus, many bands of elevation in which floors are required are distributed throughout the height of the plant. Because these bands are numerous, it is likely that at least one constraint will require a floor at any elevation. Clearly, a floor cannot be introduced to meet every constraint. The heights at which floors are needed have to be rationalised so that only a reasonable number of floors have to be introduced. It is inevitable that the floors will not satisfy all constraints, wherever they are positioned. The elevations of items and the positions of features on the items have to be adjusted to make best use of the floors. Neither an engineer nor PLS can inspect the constraints and identify the optimal heights for the floors directly. Rather, the side effects of deciding on a particular floor height must also be considered. In particular, the height of a floor might be increased slightly. This would also lift the items that have constraints satisfied by the floor at its original height. Then, the floor might also satisfy constraints imposed by items that are currently just "out of reach". The alternative to lifting the floor is to introduce another floor and lift these items so that the next floor satisfies their constraints.

PLS formalises this approach in its procedure for positioning floors. It postulates the height of a floor by applying plant layout knowledge that the first floor is at least 4 m above grade and subsequent floors are separated by 3 m. These separations are recorded in the knowledge base and can, therefore, be changed to suit the practice in a branch of the industry which does not adopt these typical figures. PLS determines which FRs are satisfied by a floor at a particular height. While doing so, PLS adjusts the positions of features of equipment items to make best use of a floor at this height. PLS then identifies the additional FRs which would be satisfied if the floor were lifted slightly; in practice to the height at which it just falls within the band of elevation offsets of the first FR that is unsatisfied. PLS also identifies items which would need to be lifted with the floor, either because the floor would no longer satisfy a constraint on an item or to maintain process constraints. PLS compares the benefit of lifting the floor with the benefit of leaving it at its current height, making the comparison by forming appropriate link sets. If the benefit of lifting the floor is greater, PLS repeats the procedure to determine whether it should be lifted yet higher. If the benefit of leaving the floor at its current height is greater, PLS sets the final height of the floor to its current height and introduces the next floor. By this approach, PLS compares the incremental benefit of lifting a floor. Thus, it can neglect the common influence of FRs that are not satisfied at either floor height. When PLS applied this procedure to the test process, for example, it postulated a height for the first floor of 4 m. This met the support requirements of the reactor vessels, R104, R108 and R112 and of the holding tank T117. When PLS considered the floor at 4.2 m, it found that the floor would still support these vessels which would not need to be lifted. It also found that the floor would also support the reactor reflux condensers, H105, H109 and H113 and provide access to the manholes on the reactors, the tube bundles on the condensers and the catalyst make-up tanks, T106, T110 and T114. However, if the floor were lifted further, it would not provide access or support to additional items. Consequently, PLS positioned the first floor at 4.2 m.

Features need only be lifted to maintain process constraints if the distance that the lower item is being lifted exceeds the slack in the elevation difference between them so that the minimum difference cannot be maintained. In PLS, the minimum and maximum differences are recorded on the SRs. Thus, PLS can readily calculate the slack and identify whether it is sufficient. This demonstrates the value of SRs in general as a record of relative positions, and in particular, the value of creating SRs between every pair of items which are related by process constraints. If the lower feature is lifted, PLS reduces the maximum difference on the SR to account for the slack that has been used.

Strictly, the procedure to determine the height of a floor positions it to achieve a local maximum in the number and importance of the FRs satisfied. For example, assume that PLS finds that the current height of a floor is more attractive than a slightly increased height. It will position the floor at its current height. However, if the height of the floor had been increased slightly more, it is possible that PLS might have found this to be far better, but this possibility is not explored. However, the procedure is entirely appropriate because it exploits an intrinsic feature of the constraints. The constraints that are most important to satisfy are those which arise from a need for support, and in particular, support for major equipment items. These constraints are almost invariably satisfied by a range of floor elevations that are almost as wide as the typical spacing between floors. Thus, if one floor does not fall into the range, it is likely that the next floor will do so and therefore, the constraint will be satisfied.

Different criteria determine whether constraints which arise from access, support and clearance requirements are satisfied. PLS distinguishes between the requirement expressed in an FR by defining three generic floor entities called "access-floor", "support-floor" and "fouling-floor" respectively. An FR created to relate a feature of an equipment item to a generic floor actually relates the feature to one of these entities. If a floor falls within the elevation band of an FR which relates "access-floor" or "support-floor", the FR is considered to be satisfied. If the FR is not satisfied, PLS can either move the feature with respect to its item or move the item itself so that the elevation band aligns with a floor. Conversely, if a floor falls within the band of an FR which relates "fouling-floor", the FR is considered to be unsatisfied. PLS must either move the feature or the item until the floor is outside the elevation band.

The Algorithm to Position Floors

1. PLS creates a frame to represent the first floor of the plant and assigns it the standard elevation of a first floor.
2. PLS forms a link set which represents the benefit of leaving the floor at its current height.
3. PLS searches for the FR which is not currently satisfied but which would need the floor to be lifted the smallest distance to be satisfied. PLS takes the bottom of the range of elevations which would satisfy this FR to be the next provisional elevation for the floor. PLS forms a link set which represents the benefit of lifting the floor to the next elevation.
4. PLS compares the link sets.
5. If the link set which represents the benefit of leaving the floor at its current height is more important, then PLS records the current elevation on the frame for the floor, creates a new floor and assigns it an elevation of the height of the current floor plus the standard height difference between floors. PLS iterates to step 2.
6. If the link set which represents the benefit of lifting the floor is more important, then PLS sets the current elevation to the next elevation and adjusts the elevation of appropriate items. PLS iterates to step 2.
7. PLS continues to iterate until all items of equipment are supported.

Once PLS has determined the heights of the floors in the plant, it can then identify the requirements for access and clearance which are not met by these elevations. PLS forms frames to represent platforms which provide access to the features that cannot be accessed from floors. It identifies these features by searching for FRs to "access-floor" which do not align with real floors. Similarly, PLS forms frames to represent gaps that must be left in the floors to clear features. It identifies these features by searching for FRs to "fouling-floor" which do align with real floors. PLS forms these frames to convey to the structural engineers the important information that platforms and gaps are required. This information would otherwise be implicit in the layout solution. Thereafter, PLS has completed the calculation of the elevation of the equipment and floors.

Engineering principles dictate whether the position of a feature can be adjusted relative to its item to make best use of a floor and when an item must be lifted if the height of a floor does not satisfy constraints. These principles are important domain knowledge. They give rise to the criteria which PLS adopts to determine which items it should consider to be amongst those which need to be lifted. They also give rise to the criteria by which PLS determines the membership of the link sets which it uses to compare the benefits of lifting a floor to the benefit of leaving the floor where it is.

Support

An item of process equipment is supported either by a skirt or legs standing on a floor or by lugs mounted on the side of the item which is then hung through a floor. Clearly, the mountings must align exactly with the floor. The layout engineer sets the position of the mountings relative to the item. Thus, PLS records an identical minimum and maximum offset between the mounting feature and "support-floor". The latitude to move the mountings is left in the FR between the mounting feature and the item's origin. This model is the most accurate reflection of the engineer's perception and also provides maximum information for the mechanical engineer who needs to know the exact position of the mountings to calculate the stress in the item's shell during detailed design. This position is recorded on the SR between the mounting feature and the origin of the item. Every item of equipment must be supported, but the support points for each are always in one plane. That is, if an item stands on one floor, it will not also be hung from the floor above. Once a floor is aligned with the support features, the floor meets all support needs of the item. If a floor is lifted incrementally above its current height, it will support additional items which would otherwise need to be lifted onto the next floor. PLS includes the FRs between each of these additional items and grade in the link set which represents the benefit of lifting the floor. PLS also includes the FRs between grade and other items, related to this item by process constraints, in the link set. Once PLS has aligned a real floor with a support feature, it must maintain this alignment because there is no other way to support the item. Initially, PLS adjusts the position of the support feature with respect to its item while it tries the floor higher in the plant. There is no particular benefit in leaving the floor at its current height because the item is not yet being lifted. However, once all latitude to move the feature has been absorbed, benefit does accrue if the floor is left at its current height. Thereafter, PLS includes the FRs between this item, and the other items related to it by process constraints, and grade in the link set which represents the benefit of leaving the floor at its current height. The FRs between the items and grade are excluded from the link set which represents the benefit of lifting the floor further.

Access

An item is often designed so that access is provided to one of its features by the same floor which supports the item. For example, vessels with a top manhole are typically specified to be supported hung from a floor with the support lugs mounted near the top of the vessel. Thus, the floor that supports the vessel also provides access to its manhole. In some cases, the feature to which access is required cannot be moved relative to the item - the manhole must be mounted on the vessel's top. In other cases, the feature can be positioned with some latitude relative to the item. For example, a temperature probe can be mounted anywhere below the liquid level on a vessel. If a floor at its current height does not provide access to a feature, the engineer will attempt to move the feature relative to its item so that the floor falls within the required height offset of the feature. While so doing, the engineer might also exploit any latitude to adjust the height of the item itself on its mountings. However, the engineer would never leave an item unsupported to satisfy a need for access. When PLS assesses whether a floor satisfies FRs to "access-floor", it reflects this practice. It will adjust the position of the feature on the item as far as the FR between the feature and the item's origin allows, and also adjust the height of the item on its supports if the FR between the item and the support feature allows. If the feature is successfully aligned with a floor, PLS records the position of the feature with respect to the item. When a floor is lifted incrementally, benefit accrues from this increment for each feature which requires access which PLS can align with the floor. The FR between each feature and "access-floor" is added to the link set which represents the benefit of lifting the floor. However, if PLS continues to lift the floor, it might be unable to align the floor with the feature. Benefit accrues if the floor is left at its current height once this occurs. Therefore, the FR between the feature and "access-floor" is added to the link set which represents this benefit.

Clearance

Clearance between an item and floors is mainly required above and below the item for pipework and above the item so that components of the item can be removed. For example, in the test process, PLS left clearance above the filter F119 so that its drum can be lifted off for maintenance and left clearance below the reactors R104, R108 and R112 for the product discharge pipework. The space left for clearance is often provided intrinsically by the item's support. For example, a horizontal shell and tube heat exchanger is usually supported above the floor on legs which intrinsically leave sufficient space for the pipework from the nozzles on the bottom of its shell. However, it is not unusual for clearance to be required elsewhere. For example, clearance must be left for the discharge pipework of the reactor vessels in the test process between the vessels and the floor below the floor from which the vessels are hung. It is rare that the feature that requires clearance can be moved relative to the item - a bottom discharge nozzle of a vessel must be mounted on the vessel's bottom! The minimum and maximum elevation offsets recorded on the FR between the feature and "fouling-floor" record the exclusion band in which the floor will foul the feature. Note that PLS does not form FRs that record an exclusion band which covers the full height of a tall item such as a vessel or column. It is common for tall items to penetrate a number of floors in a process plant. If a floor at its current height fouls a feature, the engineer will attempt to move the feature relative to its item so that the floor falls outside the excluded height band in the rare cases that this is possible. The engineer might also exploit any latitude to adjust the height of the item itself on its mountings. However, the engineer would never leave an item unsupported to provide clearance for a feature. When PLS assesses whether a floor satisfies FRs to "fouling-floor", it will adjust the position of the feature on the item if the FR between the feature and the item's origin allows, and also adjust the height of the item on its supports if the FR between the item and the support feature allows. If the feature can be successfully moved clear of the floor, PLS records the position of the feature with respect to the item. Benefit accrues from leaving a floor at its current height if lifting it incrementally would cause the floor to foul a feature. Therefore, if the floor would fall into the height range excluded by an FR between a feature and "fouling-floor" if the floor were lifted, the FR is added to the link set which represents the benefit of leaving the floor at its current height. If a feature does fall into an excluded band, the floor may be moved clear of it if PLS continues to lift the floor. In this case, the FR between the feature and "fouling-floor" is added to the link set which represents the benefit of lifting the floor. If a feature is fouled by "grade", PLS lifts the item. This corresponds to accepted good practice in plant layout of elevating items to clear the ground rather than digging pits to gain the clearance.

Chapter 8 : Group Formation

An important tenet of this work is the principle that sets of entities which must be close are collected into groups to permit four very powerful simplifications during plan layout. These simplifications have been discussed previously. Groups are used within other automated plant layout systems. In Al-Asadi's work [1980], the groups are formed manually whereas Shuquair's system [1978] forms groups automatically. However, these must then be refined extensively by the user before they can be passed to the plan layout routines. The principles which have been developed during this work allow a totally automated technique. This is far more sophisticated in both the data structures and method than Shuquair's approach and is thought to be unique in its efficacy to simplify plan layout generation.

Three types of group have been identified during this work which will be referred to hereafter as "physical", "logical" and "segregation" groups. The distinction is important because each type of group corresponds to a markedly different reason why the members should be positioned close to one another. All three types of group have the same spatial significance during plan layout, an aggregation of entities that must be close in the final layout. Thus, once the groups have been formed, they can be handled during plan layout without any distinction between their types. The three types are discussed below.

Physical Groups

The members of a physical group should be local because they share a need for the same service or facility. If these items are concentrated together, the service need only be provided at one place in the plant. In some cases, the "service" is provided by a site feature, such as an area of ground capable of bearing heavy loads. In these cases, physical groups are formed to concentrate the entities that must be positioned at this site feature. An engineer working manually explicitly forms groups that are analogous to PLS's physical groups. Indeed, groups of this type are likely to be demanded by engineering standards.

Logical Groups

It is very common for equipment items that comprise a process unit to be positioned close together in well designed layouts. Numerous constraints attract the equipment items which comprise a process unit to one another and many of the FRs which inter-relate the unit's members express strong requirements for proximity. Effectively, a cluster of constraints inter-relate the members of each process unit in the plant. The members of a process unit gravitate towards one another because of this concentration of strong, attractive constraints between them. However, the engineer does not set out to form groups of this type explicitly. Rather, they would appear to emerge in the final layout as a result of an almost tacit design objective of keeping all members of a unit operation local to one another. It appears that the engineer forms these groups by recognising the roles that the specific items of equipment are playing and thereby identifying them as members of the unit. For example, the engineer might reason that "that heat exchanger is a reboiler so it must be part of that distillation unit". Logical groups formalise this tacit objective.

Segregation Groups

Certain entities are collected together so that they may be segregated from the rest of the process to eliminate a hazard. Some of the entities that are collected might be hazardous themselves. These can be positioned close to one another if they are compatible. Other entities might be individually innocuous but might have to be positioned with one of the hazardous entities for one or both to function. Groups grow as the hazardous items are concentrated and their ancillaries are positioned with them. In the manual technique, this occurs almost as a side effect of the engineer attempting to segregate the hazardous entities. This behaviour is formalised in segregation groups.

Many entities are potential members of a number of groups. However, the simplifications that groups admit can only be made if every entity is made a member of one group at most. Entities are assigned to the group to which they are most strongly attracted. This guarantees that the strongest FRs acting on each entity will be satisfied, at least partially. In practice, a number of entities might be members of one group but might also be potential members of others. Which entities should be allowed to attract other entities into the group cannot be known with certainty until the ultimate membership of the group has been decided. A "least commitment strategy" is well suited to solving problems such as this and heuristics have been developed during this work to determine when sufficient information is available to decide an assignment with reasonable confidence. PLS uses least committment reasoning and does not make decisions before the germane information is available and therefore, it is not forced to draw assumptions that it might have to withdraw subsequently.

The least commitment strategy is implemented in PLS by forming groups in two stages during each iteration. In the first stage, PLS assigns every entity to every "preliminary group" of which it might be a member. Many of these preliminary groups have members in common. In the second stage, PLS finds each entity which is a member of more than one preliminary group. These shall be referred to as "disputed entities" hereafter. Each disputed entity is assigned to the final group to whose other members it is most strongly attracted. The attraction of the entity to the groups of which it might be a member is evaluated by comparing link sets on the PO Table. The second stage is knowledge-based to implement the heuristics and to allow other heuristics to be added at a later date. This two-stage approach is more efficient than attempting to identify which group an entity should be made a member of while the group is being created in the first instance.

As a consequence of the definition of a group - a set of entities which will be close in the final layout - it is unacceptable to impose an arbitrary minimum or maximum number of members for a group. Thus, every entity that belongs amongst the members of a group for engineering reasons should be collected into it. Nonetheless, three processes have been studied, the test process and two others worked as paper exercises. It has been observed that groups have between four and seven members in these plants. This appears to be an intrinsic feature of the domain. Groups of these sizes are ideal to simplify the plan layout task. Their members are sufficiently numerous that relatively few groups are formed in each iteration. However, their members are sufficiently few to simplify their positioning relative to one another within the group. The exceptions to this are the groups which comprise an equipment item and its standby (or standbies), which typically have two members. To some extent, these groups are a special case in that they effectively represent an almost indivisible single entity.

The groups formed by PLS while laying out the test process are tabulated in Table 3 at the end of this Chapter. This Table is presented to illustrate the operation of the generic procedure employed to form groups. Frequent reference will be made to this Table throughout this Chapter to demonstrate how the procedure copes with the practical difficulties of a "real world" problem. Conversely, these references can also be read as explanations of how PLS arrived at these groups in this instance. The groups that PLS had formed after the second iteration are also displayed by the false colours in Figures 2 to 7. PLS actually creates groups in the same order as the FR instances that cause members to be collected are created. The entries in the Table have been rationalised into an order that is clearer to the reader. The identifiers of the groups have been changed to reflect this order. The Table shows the states of the groups after both stages of each iteration.

8.1 Representing Groups

Groups simplify the plan layout procedure because they are independent entities which stand *in lieu* of their members. They are not just aggregations of other entities but have their own existence and properties, such as size and position. This feature is realised in PLS by each group being represented explicitly by a separate frame in the database. The only distinctive feature of a frame for a group is its slot named "members". The frames that represent the entities collected into the group are written into this slot. This provides yet another example of the use of a slot with frames as its value to represent a relationship. The frames written into the "members" slot are restricted to those for the immediate members of the group. If a group is a member of another, the members of the former are not listed explicitly amongst the members of the latter. This is consistent with the role of groups as surrogates for their members. That is, the members of a group are restricted to the entities which PLS will act upon directly while it is positioning them relative to one another.

Except for the "members" slot, the frames for groups have an identical general structure to those for equipment items. Thus, PLS can act upon frames which represent groups and equipment items without distinction. The iterative procedure for group formation starts by collecting frames that represent equipment items. As groups are formed, PLS collects frames that represent groups. During the early iterations, PLS collects a mixture of frames that represent equipment items and others that represent groups. For example, a pump and its standby might be formed into a group in the first iteration. During the second iteration, this group may be collected into a group along with the feed vessel and its ancillaries. Because PLS represents equipment items and groups similarly, the same code can be applied to either. A similar benefit accrues to the code for plan positioning. PLS iterates to position relatively the most expansive groups, then increasingly local groups and finally individual equipment items. At times during the iteration, PLS must position combinations of groups and equipment items. Groups were originally represented by markedly dissimilar frames to equipment items. The intention was to stress the difference between the two types of entity. However, this was found to be very inconvenient and no practical use for the distinction was found. Accordingly, the frame structures were unified.

8.2 Forming Preliminary Groups

Different principles underlie preliminary physical, logical and segregation groups. This reflects the markedly different properties and topologies of the design constraints that cause members to be added to each type. The engineer's approach to forming physical groups clearly acknowledges constraints as their cause. PLS's procedure for forming physical groups is very similar to the engineer's. However, the principles which guide engineers to form logical and segregation groups are not obvious from the manual technique. The engineer does not perform these tasks explicitly. Indeed, these principles were unexplicated prior to this work.

8.2.1 Forming Preliminary Physical Groups

Physical groups are formed by collecting entities that share a similar need. Entities can be identified as sharing a need if they are related by physical FRs that arise from equivalent causal factors. Effectively, the causal factors are the needs that the related entities pose. The FRs are merely records of these needs. Thus, if the FRs are equivalent, the needs must also be equivalent. If an entity shares a number of needs with other entities, it is made a member of one group for each need. These groups do not merge even though they share a common member. For example, a vessel might be made a preliminary member of the group of equipment items that are local to minimise the length of cooling water pipework. The vessel might also be made a preliminary member of the group of items that are local to minimise the length of high voltage power cable required for its agitator. The two groups are not merged despite both containing the vessel. This obviates one of the problems with Shuquair's approach. In this, the two groups would be merged even though the majority of the members of one had no need to be close to the majority of the members of the other. Forming groups of entities related by equivalent constraints is unique to PLS. PLS can form groups which correspond to features which are important and widely seen in layout, such as heat exchanger banks or pump bays, only because it embodies this principle.

The Algorithm to Form Physical Groups

1. PLS's knowledge base contains rules which each identify sets of physical FR instances which arise from the same causal factors by inspecting one of the "type", "name" or "criterion" slots of all physical FR instances. PLS applies the first of these rules to the database.

2. If the rule finds an FR instance that it matches, PLS forms a frame to represent the physical group. Both entities related by the FR are recorded in the "members" slot on the group's frame.
3. The rule continues to traverse the database and inspects all physical FRs. Whenever the rule matches an FR instance frame, PLS adds the entities that the FR relates to the "members" slot of the group's frame.
4. PLS invokes the next rule and the procedure is repeated until all rules have been applied.

It is well known in the conceptual layout domain that some constraints which demand proximity are more easy to satisfy than others. It is important that PLS reflects this reality. In the test process, the weighing tanks T101, T102 and T103 and the condenser H125 are all elevated above the rest of the plant and therefore, could be supported by a localised structure. FRs are formed to represent this attraction. However, the weighing tanks are also attracted to the reactors R104, R108 and R112 and the condenser to the column C124. In the configuration that an engineer would strive for, the weighing tanks are positioned with the reactors and the condenser with the column. The reactors and weighing tanks are then positioned with the column unit. This is because the constraints between the weighing tanks and the reactors and between the condenser and the column are only satisfied if the distances between the related items are short. Conversely, the constraints between the elevated items are satisfied provided that the items are positioned in the same general area of the plant. This does not imply that the constraints between the tanks and the reactors and the condenser and column are either more or less important than those between the elevated items. There is no correlation between the ease with which an FR is satisfied and the importance of satisfying it.

The relative ease with which different types of FR can be satisfied is important domain knowledge. It is sensible to satisfy the difficult FRs first. Thus, FRs that are easily satisfied should be held in abeyance for a certain number of iterations. A group formed in an earlier iteration occupies a smaller area than a group into which it is collected later. To continue with the example above, the FRs attracting the elevated items should be held in abeyance until a later iteration. Thus, PLS collects the reactors and the weighing tanks into one group and the column and condenser into another during an early iteration. It then forms a more expansive group in a subsequent iteration, collecting the reactors and column groups.

The converse of the above also applies. Some FRs are difficult to satisfy. These FRs should be neglected after a certain number of iterations if they are not already satisfied. If an FR is sufficiently relatively important to warrant the entities that it relates being collected into a group, this will occur during an early iteration. If the entities are not collected at the time that is appropriate, this implies that the FR which relates them is considered to be unimportant.

The knowledge engineer reflects this important domain knowledge by recording it as an ordered pair of numeric values on the FR's frame. The first value records the number of iterations which must be complete before the FR should be considered. The second records the number of iterations after which the FR is no longer considered. In PLS, this ordered pair is written into a slot called "activity-level". If the maximum is not specified, the FR is considered in every iteration after the minimum has been reached until the group formation procedure halts. The minimum must always be specified because the values are an ordered pair, but a value of 0 effectively imposes no lower limit. In the test process, FRs that represent the attraction of elevated items were marked as becoming active in the third iteration. Referring to Table 3, a group of elevated items (Group G29) is first seen amongst those formed in the third iteration.

The control is most needed during the early iterations. Each of these correlates closely with a recognisable "scale" of group. Thus, the significance of early iterations is clear and the knowledge engineer can control the behaviour of the FRs accordingly. The procedure starts with the assignment of each equipment item to its own group. The early iterations form groups as follows:

1st Groups of equipment items and their standbies or all machines on the same shaft;

2nd Members of unit operations into groups and groups which correspond to widely observed idioms such as heat exchanger banks or pump bays;

3rd4th Serial or parallel unit operations and unit operations or physical groups with a strong attractive relationship to one another such as the column units discussed in the example above.

The significance of later iterations is less clear. However, this does not cause a problem because only FRs which act over long distances are considered in the later iterations in any case. Thus, it is meaningless to specify exact cut-off points at which any of these FRs become or cease to be significant. Thus, the knowledge engineer can safely assume that an FR that should be active during any of these later iterations can be specified as being active during them all or *vice versa*.

As a consequence of this control, all constraints acting on an entity might be inactive during a particular iteration. Thus, PLS does not impose the restriction that an entity must be collected into any group during a particular iteration. Rather, in PLS, entities may remain uncollected or "visible" after an iteration if no FR expresses a reason to collect them.

8.2.2 Forming Preliminary Logical Groups

Mainly, logical groups correspond to process units or sets of units. Each process unit includes one equipment item which actually performs the function of the unit. The unit also includes the ancillaries of this item. These ancillaries do not perform the function of the unit *per se*, but rather, support the main item in its function. Within the unit, it is likely that the items will be highly inter-connected, probably by streams that have higher flowrates and are at more extreme conditions than those flowing between the units. Accordingly, the members of the unit are related by both many and strong FRs which arise from the process connections between them. The members might also be strongly related to members of other units, either by other FRs which arise from the process connections or because of other factors such as common service requirements. However, these FRs will not be either as strong or as numerous as those which form a dense cluster of FRs amongst the members of the unit. It is for this reason that the members of a process unit gravitate towards one another, even though the layout engineer rarely explicitly forms groups which correspond to the units. It is also for this reason that it is important that logical groups are formed given the central role of groups in this work in satisfying the constraints that influence the plan layout. The logical groups ensure that the many and strong FRs are satisfied, at least partially. Without logical groups, no explicit attempt would be made to satisfy the constraints in these clusters and therefore, many important FRs would be likely to be left unsatisfied.

For example, consider the reactor units in the test process. Each unit comprises a reactor vessel, a condenser, a catalyst make-up tank and a catalyst pump. The members of each reactor unit are related by many physical FRs which express requirements that the members should be close. The reactor is attracted to the condenser to minimise the length of the pipework, both to reduce the cost of the large bore vapour pipes and to prevent transfer lag in the control of the reactor unit as a whole. The reactor is attracted to the catalyst feed pump to minimise the length of the catalyst feed line, again to improve control of the reaction. A number of FRs also relate each member of each reactor unit to the corresponding member of the other reactor units. For example, all of the condensers and reactors are related because they all use cooling water; the reactors are also related because their agitators are connected to high power cables; and the catalyst make-up tanks are related because they all receive frequent manual attention and should be close. However, the FRs which relate the members of each reactor unit to the corresponding items in the other units are either less strong than those that relate the entities to the other members of their unit or are not synergistically strengthened by being members of a concentration of FRs. The reactor units do not coalesce because the FRs that relate corresponding members of the groups are relatively weaker. The units retain their natural autonomy and can be considered to be single macroscopic entities. The strength of the FRs within them make them almost indivisible.

This could be considered commonsense to those skilled in the art. In practice, this behaviour is difficult to formalise sufficiently to implement in a computer system. It is also difficult for a layout engineer to elucidate the procedure underlying the behaviour during knowledge elicitation or perhaps when training a neophyte. It would appear that the layout engineer determines the members of a process unit by applying compiled knowledge. The engineer "knows" the make-up of any process unit. However, this approach was rejected for use in PLS. It is difficult to define whether a particular process unit is sufficiently close to the typical configuration of a unit of its generic type. For example, we cannot even be certain of the members of a distillation unit, a very common process unit. In general, a distillation unit will include one condenser. However, if the supply of cooling water is restricted on a particular site, an air cooled exchanger might be used as the main condenser and a water cooled exchanger used as a final condenser. Thus, a distillation unit designed for this site will deviate from the norm. The compiled knowledge will break down in this situation unless a very large range is included in the system knowledge base. The technique developed during this work can determine the members of a process unit which does conform to a norm entirely satisfactorily. However, the technique works equally as well to determine the members of an atypical unit.

Two other configurations are widely seen in layouts designed manually. When flow diverges, runs in parallel through a number of units and then reconverges, the parallel units are positioned close to one another together with the entities at which flow diverges and converges. However, the parallel units are usually observed to be autonomous within this larger set of entities. The entities at which flow diverges and converges are not incorporated into any one of the parallel units but rather, are close to them all. In this configuration, each of the parallel units embodies a very strong cluster of constraints. Another, slightly weaker, cluster inter-relates the parallel units and the entities at which flow diverges and converges. This cluster of constraints is still stronger, perhaps because of the density of constraints rather than their individual strength, than the constraints that relate the entities at which flow diverges and converges to the units that precede and follow them. Thus, the parallel units and the strongly related entities are attracted to one another to form a more expansive set of entities.

The other configuration is observed when process flow progresses through a number of units in sequence, in which the first unit radically changes the process conditions so that they impose stringent engineering requirements on the subsequent units and inter-connecting pipework then finally, the last unit in the series renders the conditions fairly innocuous once more. High pressure reaction loops in which gas is compressed in a compressor unit, reacted in a high pressure reactor then let down through a turbine exemplify this. It is likely that the units will be observed to be adjacent to one another in a spatial sequence. Other units to which they are connected by streams at less stringent conditions are likely to be excluded from this sequence. Again, the individual units are likely to be observed to be autonomous within this sequence of units, for the same reason as the parallel units remain autonomous. The constraints that arise from the process connections between the units are stronger than those which relate the first or the last in the sequence to the units that precede or follow the series. Thus, the units are again attracted to one another and act as an indivisible set of entities.

The situation in which flow diverges then converges is the most complex. The following procedure suffices. The members of each process unit are collected into separate groups to maintain the autonomy of the unit. The entities at which flow diverges and reconverges, hereafter referred to as "shared entities", are not collected into any of these groups so as to not compromise this autonomy. Each shared entity is collected into a separate, independent group. If a shared entity has ancillaries of its own, these are collected into the independent group. This independent group can then be positioned as a single entity optimally with respect to all of the parallel groups. The shared entities are made members of a logical group even if they have no ancillaries of their own. This important implementational device forces these shared entities to be considered as disputed entities if they might also be members of a physical group. Thereby, they will not be assigned to the physical groups as a "bye". In the subsequent iteration, the parallel groups and the groups including the shared entities are collected into the more expansive group. This must occur in a subsequent iteration to leave an opportunity for any of these newly formed groups to be considered as potential members of other groups, particularly physical groups. This opportunity would be lost if the more expansive groups were formed "on the fly" in the same iteration as the autonomous groups. Colloquially, the logical groups would gain an "unfair advantage".

The weighing tanks in the test process exemplify this procedure. These are not collected into any of the logical groups formed around the reactor units (groups G8, G9 and G10) because each weighing tank feeds each group. However, a group is formed for each of the weighing tanks (groups G11, G12 and G13). These "protect" the weighing tanks from being made members of the group of entities that require a support structure (group G29).

Logical FRs record the direction from any entity in which the gradient of the strength of the FRs is greatest. Each logical FR relates an entity to the other entity to which it is most strongly attracted. This is the central criterion to determine to which logical group an entity should be added. The directionality of the logical FRs is exploited to ensure that entities are added to the appropriate group and also to identify entities that should be made members of independent groups.

PLS employs two separate but complementary algorithms to form logical groups. Both correspond to the procedure described above. The first identifies the members of an autonomous unit and collects them into a preliminary logical group. The second identifies parallel units through which process flow diverges and converges and collects them into a more expansive group. The second algorithm does not consider autonomous logical groups that the first has formed during the current iteration. Rather, the second collects logical groups which were formed in the previous iteration. The first algorithm is invoked during every iteration to collect groups formed during the previous iteration. It cannot be held in abeyance during an iteration merely because the second algorithm has been invoked. If it were not invoked, groups formed during the previous iteration might be collected into physical groups during the subsequent iteration without PLS considering that they might also be members of a logical group. Parallel logical groups might be formed in any iteration. If they were, the second algorithm is invoked to collect them into the more expansive logical groups. In practice, the first algorithm sets a flag if it detects groups which might run in parallel. The second algorithm is invoked in any iteration in which this flag is set. The flag is cleared at the end of each iteration. In the following description of these algorithms, the entity recorded in a logical FR's branch frame in which the "logical-end" slot has the value "master" will be referred to as the "master". The other entity related by the logical FR will be referred to as the "ancillary".

The Algorithm to Form Autonomous Groups

1. PLS searches for entities that are the master of all other entities that they are related to by logical FRs. PLS forms a frame to represent the logical group corresponding to the process unit around each entity. The entity is recorded in the "members" slot on the group's frame.
2. PLS adds entities to the logical groups of which their masters are members. This acknowledges the significance of the direction of the logical FRs. However, PLS only adds an entity to a logical group if all of its masters are members of the same group. This ensures that PLS does not collect a shared entity into more than one logical group. The entities that PLS discounts might be added subsequently if all of their masters are made members of the same group. PLS continues this procedure until no more entities can be added.
3. PLS searches for entities whose masters are members of more than one logical group. PLS forms a frame to represent the logical group corresponding to the segment of the process unit around each entity. The

entity is recorded in the "members" slot on the group's frame. It is at this stage that PLS forms the independent groups with shared entities as their members.

4. PLS iterates around steps 2 and 3 until all entities related by logical FRs have been added to logical groups.

The Algorithm to Collect Parallel Groups

1. PLS searches for a logical group whose master is an entity at which flow converges or diverges. PLS inspects the logical FRs which relate the master of this group to its masters to identify the groups to which the latter belong. Note however, that PLS has not yet determined that these latter groups are definitely parallel.
2. If the latter groups have been collected into a logical group created in the current iteration, then PLS has decided that they are parallel when considering another group containing a common ancillary. PLS adds the group under consideration to the newly created group.
3. If the latter groups have not been collected into a logical group created in the current iteration, then PLS must determine whether they are in parallel. PLS compares the physical FRs that run parallel to each logical FR between the group under consideration and the potentially parallel groups. If identical physical FRs run parallel to each logical FR, then PLS concludes that the latter groups are parallel. PLS forms a new group and adds the group under consideration and the parallel groups.
4. PLS iterates until it has checked every logical group formed during the previous iteration.

No specific algorithm is required to collect serial unit operations into logical groups. Rather, new logical FRs are formed between groups at the end of each iteration. These new logical FRs reflect the strongest FRs acting on the units now that the previously strongest FRs acting on their members have been totally subsumed within the groups just formed. However, parallel groups, and the independent groups of which the shared entities are members, are neglected when these new FRs are formed. These groups are left to be collected into the more expansive groups during the next iteration.

8.2.3 Forming Preliminary Segregation Groups

Entities might pose a hazard either in their own right because a hazardous event can occur within them, or in combination with another entity, such as an entity that might act as an ignition source in combination with an entity from which a flammable vapour might be released. In either case, the hazard is eliminated if the entity is placed away from the main area of the plant. If the entity is not to be placed within the main area, it is entirely acceptable to position other entities that pose the same hazard with it. These other entities do not increase the risk of the hazard occurring or the scale of the likely consequences if it does. For example, if a furnace is positioned away from the main area of the plant because it is an ignition source, any other furnace can be positioned with it without increasing the likelihood that a flammable vapour which drifts to the one or many furnaces will be ignited. Thus, entities that pose a similar or compatible hazard are collected into a segregation group because they cannot be left in the main area of the plant. The segregation group is the best destination for them. Crudely, segregation groups are not formed because positive benefit accrues. Rather, they are formed because they offer the "least worst" solution.

Entities can be identified as posing the same hazard if they are related by segregation FRs that arise from equivalent causal factors. This is analogous to physical groups. Similarly, if an entity poses a number of hazards, it is made a member of one segregation group for each hazard. These groups do not merge even though they share a common member. In practice, this is a highly unlikely occurrence.

An entity is segregated from the main area of the plant to protect the majority of the process equipment from its potentially hazardous effects. In practice, the decision as to which entity to segregate is somewhat arbitrary, especially if the hazard only arises between a combination of entities. In many cases, a few entities will be repelled from many entities. In these cases, the few entities should be distinguished as the entities to be ejected from the main area of the plant. This is particularly important if the equipment within a plant poses a number of potential hazards. The set of entities that will occupy the main area of the plant must be clearly defined so that the entities to be ejected from the main area can be distinguished consistently. This distinction is important domain knowledge. If the innocuous entities could not be distinguished, there would be no way to identify which of the entities related by a segregation FR should be made the member of the segregation group. There would be a risk that both hazardous and innocuous entities would be made members of the group, thereby defeating its purpose. Thus, segregation FRs are notionally directed in PLS to record which entity should be ejected from the main area. Their branch frames include a slot called "segregation". This slot is given the value "source" on the branch frame that records the object that must be segregated. The slot is given the value "target" on the branch frame that records the object that must be protected.

PLS uses a two-phase procedure to form preliminary segregation groups. In the first phase, PLS forms a segregation group to receive all entities that pose the same potential hazard if they are not segregated from the rest of the plant. In the second phase, PLS adds their innocuous ancillaries to the group, using the core of the algorithm for forming autonomous logical groups.

The Algorithm to Collect Intrinsically Hazardous Entities

1. PLS's knowledge base contains rules which each identify sets of segregation FR instances which arise from the same causal factors by inspecting one of the "type", "name" or "criterion" slots of all segregation FR instances. PLS applies the first of these rules to the database.
2. If the rule finds an FR instance that it matches, PLS forms a frame to represent the segregation group. The entity that must be segregated is recorded in the "members" slot on the group's frame.

3. The rule continues to traverse the database and inspects all segregation FRs. Whenever the rule matches an FR instance frame, PLS also adds this entity to the "members" slot of the group's frame.
4. PLS invokes the next rule and the procedure is repeated until all rules have been applied.

Major divisions of a plant are likely to be sufficiently physically large for segregation to occur naturally. No explicit action need be taken to segregate the hazardous entities. This important domain knowledge is recorded in the "activity-level" slot of the segregation FR instance frames, exactly as for physical FRs.

8.3 Developing Final Groups

Although the membership of preliminary groups is often a reasonable approximation to that of the final groups, a procedure is required to develop the final groups. An iterative procedure applies heuristics which seek entities which can be assigned to their final group with certainty. This reduces the level of uncertainty for the next pass as the number of disputed entities which are members of a group decreases and the number of entities that are known to definitely contribute FRs to the link sets increases. Thus, the estimated and actual strengths of the link sets converge and more pairs of link sets can be compared unequivocally in subsequent passes through the loop. The loop is terminated when all disputed entities have been assigned to one group. The heuristics are implemented in rules to allow additional heuristics to be added if they are discovered. These rules form appropriate link sets and compare them on the PO Table. No back-tracking was observed when PLS was applied to the test process. This demonstrates that a least commitment strategy is highly efficacious for this purpose.

Once all entities have been assigned to their final groups, some of the preliminary groups will become redundant and can be deleted. Three rules were adopted in PLS to identify the groups that should be deleted. The first is that any group that finishes with no members should be deleted. Empty groups serve no purpose.

They do not stand *in lieu* of any entities so there is no value in assigning a position to them. Groups finish with no members when all their original members are disputed entities. This does occur in practice, mainly with physical groups. For example, the condensers and reactor vessels in the test process are all members of both a logical group and the physical group into which all equipment which require cooling water are collected. All members of this preliminary physical group are disputed entities. Alternatively, a group might be found to subsume completely another once all disputed entities have been assigned to their final group. This might occur if all entities that require a particular service happen to be members of the same process unit, for example. The second rule is that the subsumed group is deleted. The subsuming group equally guarantees that its members will be close in the final layout and thus, the subsumed group does not increase the information about the solution. The third rule is that either of the groups can be selected arbitrarily for deletion. An arbitrary selection is satisfactory because the properties of the group during subsequent group formation and plan positioning are defined by the set of FRs that relate its members to members of other groups. Clearly, this set has the same members whichever group is retained. Similarly, two groups might be found to have identical members once all disputed entities have been assigned to their final group. In practice, PLS's last operation in each overall iteration of the group formation procedure is to tidy up the groups formed during the iteration by applying these rules.

"Least commitment deadlock" was only observed in one particular case which arises from a feature of the domain. Consider the three identical reactor units in the test process. All are equally attracted to each weighing tank. Deadlock occurred when the reactor units and the weighing tanks were being laid out with respect to one another. PLS handles this unusual case by making an arbitrary choice of which to satisfy. Clearly, all configurations are equally as satisfactory in this case because equal benefit accrues from satisfying each of the identical link sets. The device employed in PLS is recommended as generally useful.

There are two complex cases in assigning disputed entities to their final groups. The procedures to manage these cases are correspondingly sophisticated. When a disputed entity might be a member of either a logical or a physical group, the disputed entity and its ancillaries should be considered to be a single composite entity. If the composite entity is found to be more strongly attracted to the physical group, its members should be removed from both of the groups and a new group should be formed with the members of the composite as its members. The ancillaries of the disputed entity should be included in the composite entity because they no longer have a reason to be members of the logical group once the disputed entity is removed. Rather, they should remain with the disputed entity so they can continue to serve it. A new group is formed because the ancillaries of the disputed entity should not occupy space in the physical group to spread the area over which the shared service or facility must be provided given that they do not require this service themselves. The new group is independent and can be positioned close to the physical group if appropriate. Strictly, the new group can be collected into a group with the physical group in a subsequent iteration. The outcome is the same. PLS considers the attraction of all members of the composite to the logical group when it decides whether the composite should be removed. The ancillaries of the disputed entity should not be removed from the logical group if they are also ancillaries of another member. Rather, they should remain in the logical group to serve their other master. Thus, the attraction of the disputed entity to all members of the logical group, including its own ancillaries, should be considered in this case.

This procedure is also applied when the disputed entity is a member of an independent logical group formed because its master is an ancillary of members of a number of other logical groups. The independent group might be collected into a more expansive group with the other logical groups in a subsequent iteration. Thus, all FRs that relate the disputed entity and its ancillaries to members of all of these other logical groups should be considered to attract the composite entity to its logical group.

The second case establishes the procedure used to decide whether an entity should remain a member of a segregation group. It is inevitable that all members of a preliminary segregation group will also be members of a logical group. The logical FRs that cause the ancillaries of the hazardous entity to be made members of the segregation group will also cause them to be made members of the logical group. Similarly, one or more members of the segregation group might also be members of physical groups. However, entities are not made members of a segregation group because they are particularly attracted to it. Rather, the entities are repelled from the bulk of the process and the segregation group is formed to act as a repository for its members. Accordingly, the effect of the assignment of the entities on the layout as a whole should be considered rather than only on the groups to which they might be assigned. That is, the strength with which the entities are repelled from all entities in the body of the process should be compared with the strength by which they are attracted. Furthermore, the attraction of the disputed entities to the other members of the segregation group should not be considered. Entities should not be "pulled in" to a segregation group. Rather, they must be "pushed out" from the rest of the process. This is because benefit accrues if the members of the segregation group are separate from the bulk of the process but a substantial cost is also always incurred. This procedure embodies a philosophy that entities should only be allowed to remain members of a segregation group if there is no alternative.

The procedure is complex and therefore worthy of discussion. PLS forms and compares up to six link sets to determine whether the hazardous entity, the hazardous entity and its ancillaries or neither should be assigned to the segregation group. In practice, PLS forms the minimum number of these link sets as necessary to make a decision. These link sets include FRs as follows:

1. All segregation FRs which relate the hazardous entity to all entities other than those which are currently members of any segregation group.
2. All physical FRs which relate the hazardous entity and its ancillaries to all entities other than those which are currently members of any segregation group.

3. All physical FRs which relate the hazardous entity to its ancillaries.
4. All segregation FRs that relate any hazardous entities in this segregation group to the ancillaries of the hazardous entity being considered.
5. All physical FRs that relate the ancillaries of the hazardous entity being considered to all entities other than those which are currently members of any segregation group.
6. All physical FRs which relate the hazardous entity to all entities other than those which are currently members of any segregation group.

PLS assigns the hazardous entity and its ancillaries to the segregation group if it is unacceptable to place the hazardous entity within the body of the process and the hazardous entity's ancillaries must be placed with it. This segregates the hazardous item and allows its ancillaries to serve it. PLS selects this combination if link set 1 is stronger than link set 2 and link set 3 is stronger than link set 4.

PLS assigns the hazardous entity to the segregation group but returns the innocuous entities to the body of the process if the ancillaries are particularly vulnerable to the hazard or if it is unacceptable to place the hazardous entity within the body of the process but desirable to place the hazardous entity's ancillaries there. PLS selects this combination if link set 1 is stronger than link set 2 and either link set 4 is stronger than link set 3 or link set 5 is stronger than link set 3.

PLS removes the hazardous item and its ancillaries from the segregation group either if the hazardous item is more strongly attracted to the remainder of the process than repelled from it or if its ancillaries are more strongly attracted to the remainder of the process and it is more strongly attracted to its ancillaries than the strength of its repulsion from the rest of the process. PLS selects this combination if link set 6 is stronger than link set 1 or link set 5 is stronger than link set 1 or link set 3 is stronger than link set 1.

Table 3: Groups Formed per Iteration.

	Identifier	Type	Purpose	Members	
				Preliminary	Final
1st	G1	Physical	Standby pair	P107A, P107B	P107A, P107B
	G2	Physical	Standby pair	P111A, P111B	P111A, P111B
	G3	Physical	Standby pair	P115A, P115B	P115A, P115B
	G4	Physical	Standby pair	P116A, P116B	P116A, P116B
	G5	Physical	Standby pair	P118A, P118B	P118A, P118B
	G6	Physical	Standby pair	P122A, P122B	P122A, P122B
	G7	Physical	Standby pair	P128A, P128B	P128A, P128B
2nd	G8	Logical	-	**R104**, H105, T106, G1	R104, H105, T106, G1
	G9	Logical	-	**R108**, H109, T110, G2	R108, H109, T110, G2
	G10	Logical	-	**R112**, H113, T114, G3	R112, H113, T114, G3
	G11	Logical	-	T101	T101
	G12	Logical	-	T102	T102
	G13	Logical	-	T103	T103

	Identifier	Type	Purpose	Members	
				Preliminary	Final
	G14	Logical	-	G4	G4
	G15	Logical	-	T117, **F119**, T120, P121, G5, G6	T117, F119, T120, P121, G5, G6
	G16	Logical	-	H123, **C124**, H125, T126, H127, G7	H123, C124, H125, T126, H127, G7
	G17	Logical	-	**D129**, V130, X131, X132, P133, H134	D129, V130, X131, X132, P133, H134
2nd	G18	Physical	Cooling Water Service	R104, H105, R108, H109, R112, H113, H125	None (deleted)
	G19	Physical	LP Steam Service	H123, H127, P121, H134	None (deleted)
	G20	Physical	Frequent manual attention	T106, T110, T114	None (deleted)
	G21	Physical	415 V, high current	R104, R108, R112, X131, P133	None (deleted)
	G22	Physical	4 kV, high current	F119, D129	None (deleted)
	G23	Segregation	Dust explosion	**D129**, V130, X131, X132, P133, H134	Subsumed by G17

	Identifier	Type	Purpose	Members	
				Preliminary	Final
3rd	G24	Logical	(Parallel flow)	G8, G9, G10, G11, G12, G13, G14	G8, G9, G10, G11, G12, G13, G14
	G25	Logical	-	G15, G16, G17	G15, G16, G17
	G26	Physical	Cooling Water Service	G8, G9, G10, G16	None (deleted)
	G27	Physical	LP Steam Service	G18, G19	None (deleted)
	G28	Physical	415 V, high current	G8, G9, G10, G17	None (deleted)
	G29	Physical	Elevated, support structure	G11, G12, G13, G16	None (deleted)
	G30	Physical	4 kV, high current	G15, G17	None (deleted)
	G31	Segregation	Dust explosion	G17	None (deleted)
4th	G32	Logical	-	G24, G25	G24, G25
	G33	Physical	Elevated, support structure	G24, G25	Subsumed by G32

Entities marked in bold are masters of logical groups

Chapter 9 : Plan Positioning

The most difficult and creative task in process plant layout is to determine the positions of the equipment in the plan, although it is interesting to observe that this task typically consumes less time than the more routine tasks of sizing the equipment and calculating elevations. This is because an engineer exploits innate human spatial reasoning skills, the ability to visualise or imagine the solution as it emerges and the extreme flexibility of thought. It is likely that these faculties would prove highly difficult to emulate in a computer program. Even with these faculties, the engineer must rely heavily on stereotypical solutions acquired over many projects. These stereotypes capture typical configurations of sections of plant. PLS uses an approach to plan positioning that has many similarities with the engineer's. Like the engineer, PLS decomposes the plant into groups before attempting to position the equipment items. However, PLS does this using a highly formalised approach as the first of two complementary but separate tasks which comprise PLS's overall technique for plan layout. The second task is to position the members of each group relative to each other and relative to members of the other groups until the plan layout emerges. This Chapter describes the techniques used in PLS to conduct this second task.

Substantial information about the plan layout has been derived once groups have been formed. Many of the more difficult layout decisions have been made. Once PLS has formed the groups, it has determined:

- Which items will be close to one another, however PLS positions the members of each group in detail.
- Which attractive FRs can be considered when PLS selects those which are to be satisfied fully.

- Which attractive FRs will be partially satisfied even if PLS does not consider them explicitly when positioning entities.
- Whether it is possible to separate items which, for preference, would be distant from one another.
- Which repelling FRs will be fully satisfied and which unsatisfied, so that PLS can neglect the latter when positioning entities.
- That all FRs within a group demand a similar degree of proximity to be fully satisfied, so PLS can treat all FRs within a group identically.

This simplifies PLS's plan positioning task to selecting the FRs which it will guarantee to satisfy fully, positioning the members of each group relative to one another to satisfy the selected FRs and any others which can be satisfied serendipitously, then mapping the relative positions into absolute positions. This applies equally to groups of any expansiveness. Thus, PLS employs a near identical approach to laying out the members of any group. It is important to note that this approach was developed during this work specifically to lay out chemical process plant and specifically to be used in conjunction with the preceding group formation phase.

PLS determines the relative positions of the members of a group to satisfy the FRs which impose the stronger demands for proximity or separation, in preference to the other FRs if necessary, to develop an optimal plan layout. If physical FRs demand different degrees of proximity, they influence the membership of more or less expansive groups. Within a group, all physical FRs demand a similar degree of proximity. There is no scope to satisfy fully relatively unimportant FRs, while holding in abeyance other FRs that are more important but also more easily satisfied. It follows from this that, for any physical FR individually, the best relative position of the entities that it relates is adjacent. Thus, in an optimal plant layout, the entities that are related by the most important physical FRs should be adjacent. Similarly, any segregation FRs between members of a group demand a degree of separation that is comparable to the degree of proximity that the physical FRs between the members demand. Thus, segregation FRs can be introduced directly into the analysis. The difference between a segregation FR and a physical FR is only manifested when PLS specifies distance between the related members. If members of a group are related by a segregation FR which is more important than the physical FRs which relate them, these members are positioned far from one another in an optimal solution.

However, some factors do not constrain the relative position of items directly. Rather, the relative positions must be set to ensure that looser design objectives are met. One of the most important of these design objectives is the need to provide routes by which personnel can reach the spaces where operation and maintenance tasks are carried out and through which equipment items can be removed for repair. These routes are not the same as the spaces for operation or maintenance themselves. For example, space must be left around both ends of a shell and tube heat exchanger so that the head and channel can be removed for tube cleaning. There is no intrinsic need for space along the majority of the length of the shell. However, if the layout is designed so that the routes to both the head and channel access spaces approach the exchanger from the same end, space must be left alongside the exchanger through which one of the routes can pass. The minor routes to the equipment access spaces lead off from a major access way. The minor routes provide short and direct paths from the access way to the points where personnel will work, to facilitate escape in the event of an emergency. The major access way runs either through or alongside the plant and is typically 3 to 4 m wide. It is imperative that the major access way is straight and entirely clear of process equipment and pipework. This allows maintenance machinery such as cranes and fork lift trucks to be driven through the plant, and provides the primary escape route from the plant. Another important design objective is the need to provide routes for piping in, through and out of the group. These routes are often provided by pipe racks. Pipes will enter the group along the rack, leave the rack to connect to individual items of equipment, return to the rack and leave the group. Pipework connecting members of the group will also run via the pipe rack unless the items are adjacent. Pipes are routed via the rack because the layout would quickly become cluttered by pipes run direct from nozzle to nozzle. Effectively, access to equipment would be prevented.

In any localised section of a plant, the configuration of access way, process equipment and pipe rack will match one of very few alternatives. These configurations can be assembled into an unlimited number of layouts, of course. They have arisen over many years as engineers have considered the fundamental constraints on layout and have responded to feedback from plant operation and maintenance personnel. These configurations form the basic building blocks of a plan layout. An engineer selects a configuration to suit a section of plant then lays out the equipment to achieve the configuration. It would be somewhat artificial for an engineer to neglect this accumulated knowledge. If the engineer were to do so, he would recreate one of the configurations in most cases, because the known configurations cover all acceptable layouts for a typical plant. Thus, it is extremely rare that a layout engineer devises a plan layout from first principles. Once the engineer knows the configuration of the access ways for a section of the plant, they impose a localised skeletal structure into which the equipment is arranged. In a few cases, plan layout is reduced to a one-dimensional problem of ordering the equipment items along the access way.

PLS echoes the engineer's underlying philosophy when deriving the plan layout of a group but implements the philosophy in a very abstract approach. PLS decides directions, and later distances, between the members of the group by applying rules which inspect the topology of the FRs between them. These rules consider to which other members any member is most strongly attracted and consider which members have the strongest claim to any particular space. They postulate possible directions or distances and record them on SRs as their conclusions. Each of these rules is predicated on the access configuration for the group, although many of the rules apply to more than one configuration. PLS selects the access configuration for the group before starting to position its members. It considers factors such as whether the plant is enclosed in a structure or a building; the number of group members which require access; how much of this access must be provided from the main access way; whether the major flows into the group enter at the same end as the major flows out leave and the relative importance of flows through the group, the recycle flows and forward going flows between the group members.

PLS retains total flexibility in how it achieves the selected access configuration because it follows a procedure and selects the positions of each member to best satisfy the FRs that act upon it. It does not lay out the group by applying a stereotypical solution, unlike an engineer. That is, PLS relies on meta-knowledge to lay out the groups. The engineer would appear to use "scripts" which correspond to the concept of Schank and Abelson [1977] of prototypical sequences of events which can be stepped through to produce a solution.

This approach was selected for PLS partly because it corresponds reasonably closely to manual practice. Therefore, it is comprehensible to the user. It facilitates knowledge elicitation, increases user confidence in PLS's results and will allow audit information to be expressed in terms that relate to the domain. It was also clear from the literature that layouts generated by computer systems, whether for process plant or other facilities, only include adequate provision for access if the approach that the system embodies explicitly recognises this factor. A recurring theme throughout the review of computer systems for automated layout in Chapter 2 is that many produce layouts in which access ways are at best tortuous and are frequently fragmented and disconnected. This is explained by the following example. In one typical configuration for process plant layout, the process equipment is aligned, one or two rows deep, alongside the access way. This configuration increases the distances between the equipment items, whereas the constraints between the items tend to attract them towards one another. The constraints between the equipment items alone force a dense layout through which passage is difficult or impossible. A layout which satisfies the design objectives of straight and clear access ways will not emerge unless this objective is imposed as a constraint. It appears that the access ways must be deliberately designed into the layout, and the pattern of the access ways must be decided before layout commences. PLS's approach does this. Furthermore, many entities require access at a number of positions. A previous example has shown that a shell and tube heat exchanger requires access at both ends. A computer system would need to try a huge number of orientations and positions of the entities before satisfactory access routes would emerge from the individual access spaces around the entities aligning and coinciding. This approach could be combinatorially explosive.

A total of 8 configurations for group layout were elicited during this work. These cover the layout of all sections of the test process and two other plants. PLS selected two of these to lay out the test process. It used the same configuration for all of the groups which comprised equipment items as their members. This configuration is perhaps the most common in process plant in general. A major access way runs across the front of the group and a pipe rack with a minor secondary access way beneath it runs along the back of the group. Members of the group are placed alongside the major access way if they require attention frequently, if a lot of space is required around them for operation or maintenance or if physically large components might be removed from them during maintenance. Small items which can be removed complete for maintenance are positioned at the back of the group where they can be reached from the secondary access way. Items which need attention very infrequently are also placed towards the back of the group. A second configuration was used to lay the groups out relative to one another. In this, a major access way runs through the centre of the group on its major axis. Members of the group are oriented so that their sides at which substantial access is required face the major access way. Two minor access ways run along the sides of the group, parallel to the major access way, with pipe racks above. This configuration is appropriate if the pipe racks contain relatively few, relatively small bore, pipes. A central pipe rack, which must be constructed before the process equipment is installed, is more appropriate otherwise.

Both of these configurations can be used to lay out groups whose members are either predominantly equipment items or predominantly groups. This is true of the majority of the configurations. Clearly, PLS must select a configuration for the more expansive group that is compatible with the configuration of the less expansive groups being positioned within it. The configurations used in the test process fit together. The major access ways of each less expansive group would merge along the central access way running through the plant. The secondary access ways would merge to form passages between the process equipment and walls of the building. The pipe racks designed into each less expansive group would also join together along these spaces to form an integrated piping system for the plant as a whole.

In PLS, the FRs and SRs manipulated during plan layout run between features of the entities, much like elevation constraints. Thereby, the orientation of an entity can be selected to best satisfy the constraints. The constraints on the components of an equipment item impose "moments" on the item to rotate it to fit the layout of its group, the constraints on the individual members of a group impose moments on the group as a whole. Furthermore, if an entity is physically large, it is important that other entities are positioned relative to the appropriate point on the entity. For example, the air heater H134 in the test process is positioned relative to the air inlet nozzle of the dryer D129.

9.1 The Plan Tree

Generally, the FRs between the members of a group form a network of adjacency requirements. PLS imposes an arrangement of the equipment which is effectively linear and thus, does not correspond to the topology of the network. PLS determines the relative positions of the equipment in the linear arrangement by identifying the member of the group which has the strongest claim to any physical position in the line. Notionally, PLS compares the strength of attraction of the item being considered to sets of items either side of each of its potential positions. For example, to determine whether item A should be North or South of item B, PLS would compare the attraction of item A to all other items North of B to its attraction to all other items South of B. There is an obvious contradiction in this. In principle, until PLS has positioned all other items, it cannot know which items should be allowed to attract the item being positioned in each direction.

PLS circumvents this difficulty by forming an approximate structure of the layout of the group called the "Plan Tree". The Plan Tree is a tree of the strongest link sets between the group members, extracted from the network of FRs. In physical terms, the Plan Tree contains the most important proximity and separation requirements amongst the group members. One SR is created to correspond to each of these important positional requirements and one node in the Plan Tree corresponds to each member of the group. PLS forms link sets of all FRs relating each pair of entities in the group and orders these link sets by decreasing importance via the PO table. The maximal spanning tree is extracted from this network automatically by PLS, using Prim's algorithm [Aho *et al* 1983]. The group is easiest to integrate into the overall layout if the member to which the most important positional relationships enter the group is at one end of the group. PLS identifies this member and uses it as the root of the Plan Tree when it is being formed.

The number of branches in the Plan Tree and the relative length of these branches depends on the topology of the constraint network within the group. This topology defines the optimal layout and appropriate access configuration for the group. The structure of the Plan Tree and the criteria by which the access configuration is chosen arise from the same fundamental factors. Thus, the Plan Tree maps to a physical layout easily. PLS uses the structure of the Plan Tree as an indicator of which access configuration should be used for a group, although some topologies suggest a number of configurations and other factors must be considered. For example, a Plan Tree might split into two branches. The group would be laid out with both branches progressing in the same direction if the members at the end of both are attracted to the same entity. Thereby, both branches would terminate close to that entity. A configuration in which the group is laid out down both sides of a pipe rack would be appropriate. If however, the members at the end of the branches are attracted to different entities, it would be appropriate to adopt a Tee configuration. The two branches would be laid out along pipe racks which diverge from the branching node. The criterion by which PLS selects which member to position at an end of a group will depend on the access configuration it has selected for the group. Consider, for example, the configuration in which a group is laid out in a single linear arrangement along one side of a major access way. PLS chooses this configuration when process media flow through the group, in at one end from one group and out at the other end to another. PLS selects the member from which the most important outgoing flows leave the group as the most "Northerly". It is important to note that PLS can apply these criteria before the relative positions of the groups are known.

PLS treats the members of the group in a main branch like beads on an abacus wire when it lays them out. This leads to a layout which is most efficient and requires least doubling back. This corresponds to the axiom, widely held amongst layout engineers, that a plant should be laid out to follow process flow. Flow is one of the major factors that constrains the detailed juxtaposition of the process equipment.

Typically, most members of the group are members of a main branch. Consider the Plan Tree for the group which contains the column C124 in the test process, shown in Figure 8. This group has one main branch which progresses through H123, C124, H125, T126 and P128. The other member, H127, is in a separate side branch. This Plan Tree branches to a degree that is typical of those in the test process and in two other processes. This example shows how closely the Plan Tree corresponds to the satisfactory final layout of the group. The assumption that the Plan Tree is a sound basis for a good layout is validated.

The Plan Tree for the group which contains the filter F119 in the test process is unusual. It is shown in Figure 9. The side branches include more members of the group than the main branch. However, this group exhibits another important property of the Plan Tree which makes it useful even in this unusual case. The

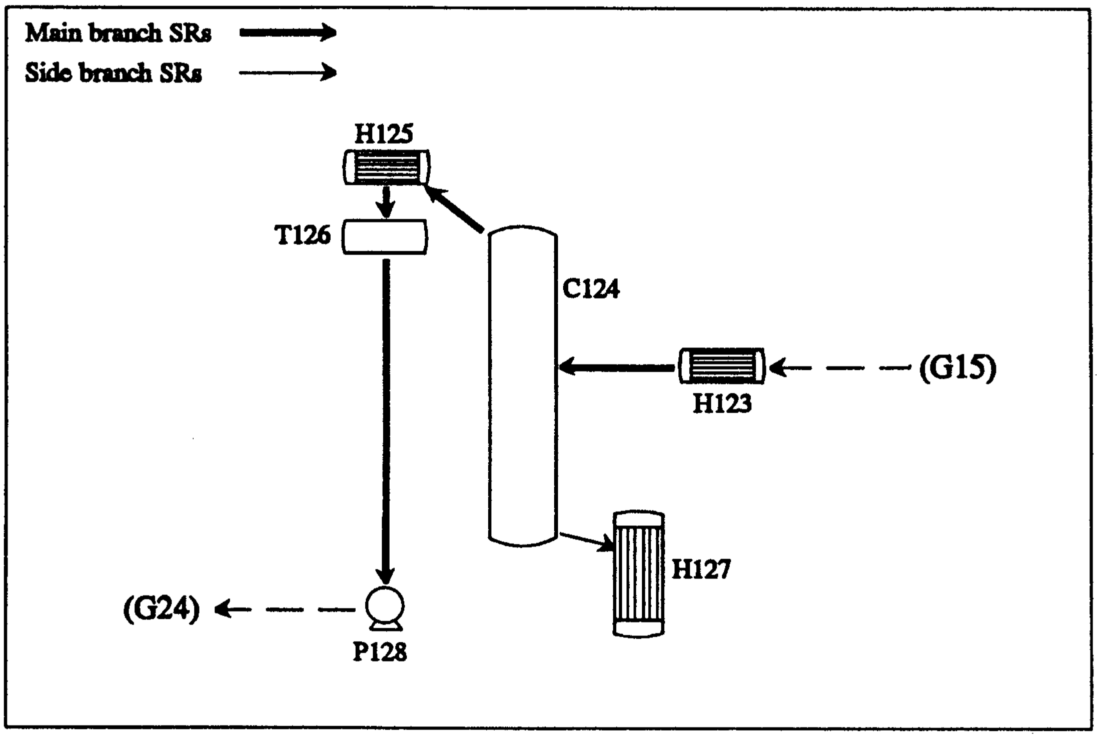


Figure 8: Plan Tree for Column Group

members of the side branches are physically smaller than the members of the main branch in this group because the flows through the members of the side branches are relatively low. The main flow through a group progresses through the members of the main branch. This is often observed. The smaller items are relatively easy to position - they can be tucked into gaps amongst the larger members of the group to make good use of available space. Furthermore, the members of a side branch often connect to members of other groups or even to service streams - they are often supporting ancillaries of the main items in the group. These items can be positioned more flexibly because they are only subject to one major constraint within the group. Thus, they can be positioned to whichever side of the major items that they serve to make best use of space. It is important to note, however, that the members of a side branch cannot be assumed to be less strongly constrained to the item they serve than members of the main branch are constrained to one another. In some cases, the strongest constraint on the major item will run between it and an ancillary on a side branch. The thermosyphon reboiler, H127, is a good example.

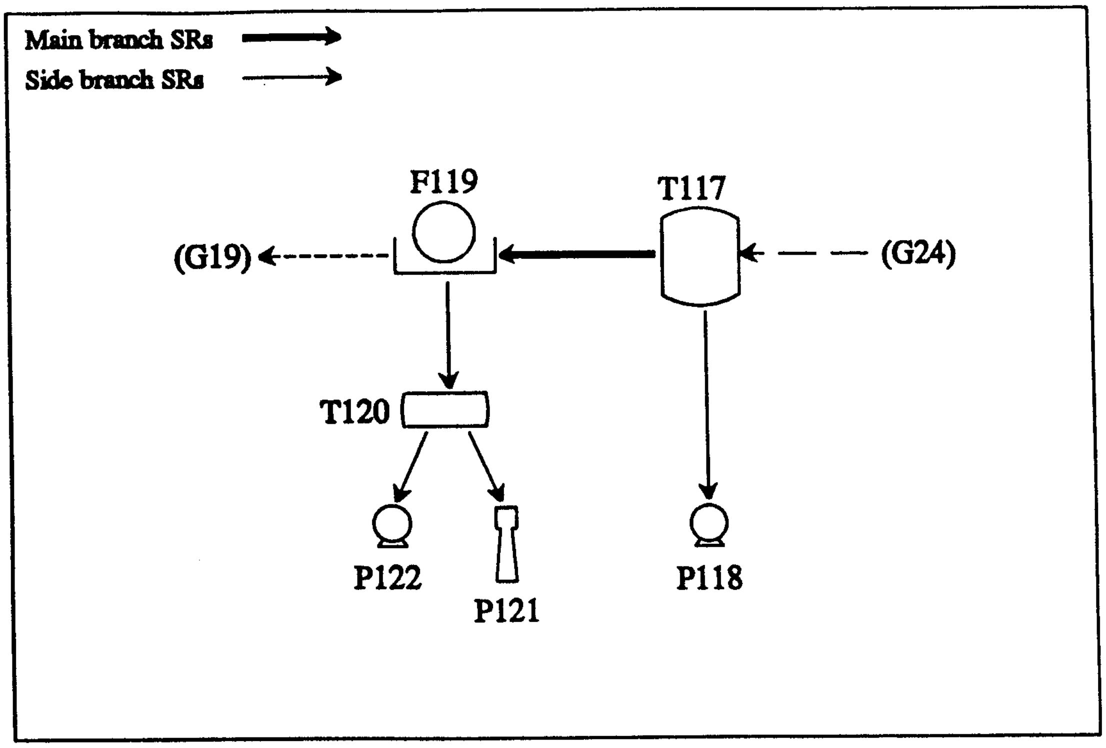


Figure 9: Plan Tree for Filter Group

Sometimes, the position of a group member is set by criteria other than its position in the Tree. For example, in some configurations, some members will be positioned to one side of the majority of the other members, away from the access way and close to the pipe rack. Nonetheless, the basic structure of the group's layout, expressed in the Plan Tree, is important when deciding the position of every member, even if their positions are ultimately modified by other criteria.

9.2 Positioning The Group Members

The process equipment occupies a very small proportion of the total volume of a typical process plant. The surprising figure of 5% is widely stated. The remainder of the volume is occupied by access ways, structural members, instrumentation and control equipment and pipework. The layout engineer does not usually lay out the plant with the objective of packing the process equipment as closely as possible. He recognises that "empty" space between the equipment items will be put to good use by the engineers performing the detailed design of the plant. Rather, the layout engineer positions the equipment items to best satisfy the constraints on their relative positions. This time-honoured practice has been found to produce layouts that are economical and practical. The engineer positions two items adjacent to one another if they are strongly attracted to one another. He will introduce an item between two others if the former is more strongly attracted to the space between the latter items than the latter items are attracted to each other. The item might be attracted to the space because it is attracted to the items which it is forcing apart on either side. Alternatively, the items which are being forced apart might be the least strongly attracted of all those alongside an access way or pipe rack. If the item being pushed in is strongly attracted to this feature, this is the least disadvantageous position for it.

Although the engineer does not attempt to pack the members of the group as closely as possible, he still attempts to use space economically and minimise the overall size of the group. He will position items above others if there is sufficient clearance, particularly to reduce the horizontal distance between strongly attracted items. He will position items in otherwise empty space if this does not compromise the constraints on their positions. Finally, a high proportion of the members of the group must be positioned alongside the access way. The space alongside the access way is at a premium. Every item positioned there increases the length of the group. Thus, if the engineer decides that it is essential that an item is alongside the access way, he will position it there. Otherwise, he will exclude it, even though it might have a limited need for access. For example, a pump in a large bore line carrying cooling water will be generally reliable but also heavy. The ideal position for the pump would be alongside the major access way, to facilitate its removal if it were to fail. However, because the pump is generally reliable, it would be acceptable to design the layout so that the pump would have to be lifted out on slings occasionally. PLS shares the engineer's philosophy. It places items in the "best" position if they have a strong claim to that position, as defined by the FRs acting upon them. However, PLS will also place an item in the "least worst" position if there is no obvious best position for it.

In configurations with either one main branch or two which run in parallel, PLS develops the layout "Northwards" away from the root member. If the group has two parallel main branches, PLS synchronises the distance from the root to which it extends each branch to allow for cross links between the two branches that the configuration requires. PLS deals with these links as it encounters the linked entities and adjusts the length of the shorter branch to suit. In a group in the form of a Tee, PLS adopts the convention that one main branch runs "North" from the root and one "South". When PLS has laid out each group individually, it assembles them to lay out their surrounding group. It reflects and rotates the groups to fit the more expansive group. In the test process, PLS reflected the groups containing the filter and the dryer along their major axes.

9.3 Deriving Absolute Positions

PLS produces an approximate layout for each group from which it estimates the space each group occupies. Then, starting with the most expansive group, PLS lays out its members and determines accurately their relative positions. PLS continues to lay out progressively less expansive groups, considering this time the effect of FRs to other parts of the plant.

PLS then translates the relative positions of the groups within the most expansive group into absolute positions and recursively descends the heirarchy of groups, translating the relative positions of increasingly less expansive groups into the same coordinate system. Once this has been done for all groups, PLS knows the absolute positions of every equipment item in the plant. PLS has completed the layout.

9.4 Detailed Treatment of Plan Positioning

The preceding Sections have developed the principles of PLS's procedure for plan positioning. This Section presents the minutiae.

PLS selects one of the eight configurations to use for the group and identifies the one or two members that will be positioned at the end of each main branch. PLS traverses the Plan Tree, marking the frames of all of the SRs in each main branch. Thereby, PLS can identify readily group members that are on the main branch. This is important because PLS has less flexibility to position these entities. The same token is used to mark SRs in Plan Trees with one or two main branches so that positioning rules can be used for either without distinction.

PLS then traverses the Plan Tree again, this time to determine the orientation of each member and the directions and separations between the group members. It creates SRs to record this information. When PLS encounters a localised branch in the Plan Tree, it follows the SR on the main branch, if possible. This exploits the property of the Plan Tree of highlighting the group members which are likely to be more awkward to position. This reduces the probability of PLS having to adjust the layout to accomodate the side branches, but can't eliminate reworking entirely because side branches can contain large or otherwise difficult entities.

As PLS encounters each entity, it determines the provisional orientation of the entity before its position. The orientations are checked and adjusted once the group has been laid out. PLS contains a suite of rules which encode domain knowledge about the appropriate orientation of an entity. These rules are invoked by control rules, or "meta-rules" which are predicated on the group's configuration. Examples of this knowledge and where it was applied in the test process follow:

- If a feature on one side of an entity is attracted to a feature on another member of the group, the entity is oriented with the feature towards the other member. Typically, the feature will be a nozzle and the attraction will arise from process flow. The filter F119 was oriented with its discharge chute towards the dryer for this reason.
- Streams that flow between members of two groups or carry a service fluid will almost invariably be routed via the pipe rack. If the flow is large, then the nozzle through which the stream flows is oriented towards the pipe rack. This simplifies the pipework. For this reason, PLS arranged the condensers with their cooling water nozzles towards the pipe rack.

- If frequent or unencumbered access is required to any feature of the member, then the member is oriented with this feature towards the major access way. This is illustrated by the reactor vessels being oriented with their manholes towards the major access way.
- Long items are oriented across the group if no other factors suggest a different orientation. This uses space economically because the equipment is not strung out unnecessarily. The vacuum separator T120 was oriented across the group for this reason.

PLS then determines the position of the entity. The position is expressed in SRs which record the direction and distance of this entity from other members of the group which have been positioned already. PLS uses rules which identify possible positions for an entity, determine whether these positions are still available, and select the preferred position from amongst them, given the constraints acting upon the entity. They are predicated on the access configuration selected for the group and therefore, only make recommendations that can be implemented within that configuration. The rules are widely applicable, covering many situations in any plant and suitable for use from plant to plant. They do not express knowledge such as "a pump should be positioned under a pipe rack". The rules are invoked explicitly from other rules in what amounts to tightly controlled backward chaining. Some rules establish generalities, such as that the entity should be positioned alongside an access way because it requires access. Others consider detailed juxtapositions. Each of the general rules invokes more specific rules, selected to be compatible with the general rule's conclusions. A small set of these rules is presented below to illustrate the philosophy of the approach. The rules are simplified slightly for clarity:

1. If frequent access to an entity is required, then the entity must be positioned alongside the access way. Create an SR from the feature that requires access to the access way, with direction "East" and distance 0, and invoke rules 2 and 4.

Interpretation: This rule determines that the entity should be positioned alongside the access way. This is recorded by an SR from a feature of the entity to the access way of 0 length, that is, the feature is aligned with the access way. Thereafter, the rule invokes other rules to determine where the entity should be positioned amongst the group members.

2. If an entity is in a main branch of the group and if it fits above or below the most "Northerly" entity alongside the access way, then create an SR with direction "North" and distance 0 from the previous entity to this entity else invoke rule 3.

Interpretation: If the entity is in a main branch of the group, it must be positioned at the "Northernmost" end so that the group continues to grow "Northwards". If the entity will fit above or below the entity which is currently at the end, this is the most economical position. Otherwise, the entity must be positioned "North" of the last entity. Note the control of subsequent actions that this rule exercises.

3. Retrieve the distance between the origins of two entities in the "North" to "South" direction. Create an SR between the first and second entities with direction "North" and with this distance recorded on it.

Interpretation: This rule retrieves the minimum spacing of entities which are next to one another, in this case, "North" and "South", and creates an SR to record the spacing and direction.

4. If an entity is not on the main branch of the group and if it fits above or below the entity "South" of the entity from which the side branch grows, then create an SR with direction "South" and distance 0 from the previous entity to this entity, else invoke the rule 5.

Interpretation: This rule is similar to rule 2. However, the preferred position for an entity on a side branch is "South" of the entity to which it is most strongly attracted. PLS reserves the space to the "North" of the entity for members of side branches which it has yet to treat - these subsequent side branches will be more "Northerly" than the current side branch because they grow from more "Northerly" entities in the main branch. This rule expresses heuristic knowledge which makes good use of the assumptions that the Plan Tree supports. The position of the entity is chosen from a number of otherwise equally good positions to minimise interference with the rest of the layout. The ejector in the test process was positioned by a rule such as this.

5. If an entity fits above or below the entity "North" of the entity from which the side branch grows, then create an SR with direction "South" and distance 0 from the next entity to this entity else invoke rule 6.

Interpretation: If the entity does not fit to the "South" of the entity from which the side branch grows, then try to the "North".

6. Select the entity to be positioned then find the pair of entities alongside the access way that are least strongly attracted to one another. Delete the SR between these entities. Invoke rule 3 on the more "Southerly" of the pair and the entity being positioned. Invoke rule 3 on the entity being positioned and the more "Northerly" of the pair.

Interpretation: In this situation, PLS cannot find a suitable position for the entity either immediately to the "South" or "North" of the entity from which the side branch grows. The entity must be positioned alongside the access way - PLS adopts the engineer's philosophy and only specifies that access is required if it is essential. PLS invokes this fall-back rule which searches for the pair of entities alongside the access way which are least strongly attracted. These can be forced apart to admit the new entity with least disadvantage to the layout as a whole. Note once again how PLS benefits from recording relative positions on SRs rather than recording absolute positions. Usually, the SRs either side of the entities being forced apart remain valid and no recomputation of position is required.

In process plant layout, different minimum distances between two entities will be required depending on the types and function of each entity. For example, generally, it should be possible to disassemble an item of equipment without removing pipework from any other, so that the plant may continue to be operated. However, if a reactor vessel's head needs to be removed for maintenance, the condenser for the reactor is temporarily redundant. There is no benefit in leaving sufficient clearance between the vessel and the condenser so that the pipe work can be left *in situ*. The space required between two entities also depends on their relative orientation. For example, more space is required between a vessel with a bayonet coil and another equipment item if the other item is positioned in the direction in which the coil will be withdrawn. Space must be left above a vertical tubular exchanger for the tubes to be withdrawn, whereas it must be left in front of a horizontal exchanger. In either case, the space can be used for maintenance of other equipment, but equipment cannot be positioned within it. Clearly, there is adequate clearance between a horizontal exchanger above a vessel with a top mounted agitator if the space needed to withdraw the tubes and agitator shaft coincide. There is inadequate clearance if the exchanger is immediately above the agitator. Thus, PLS infers the distance required between two entities specifically for that pair. It does so using Procedural Attachments which derive the distance from the origin of one entity to the origin of another. Procedural Attachments are used so that the distances will be re-calculated as PLS adjusts the layout and queries different space needs. Each Procedural Attachment calculates and returns the distance in one cardinal direction. These Procedural Attachments can refer to tables of typical separation distances, elicited from published tables [*eg* Mecklenburgh 1985, pp 555-577] and those in engineering standards.

Chapter 10 : Discussion

During this work, an approach to automating process plant layout by Expert System has been developed. This approach has been implemented successfully in PLS which can generate process plant layouts from chemical process data, automating each aspect of the task. Thus, the first and third objectives of the work have been achieved. PLS has been applied to a test process which is representative of typical process plant layout problems. It generated a layout that is entirely satisfactory and conventional from an engineering viewpoint. The second objective has also been achieved.

10.1 The Value of PLS

PLS has the potential to improve the quality of process plant design generally. It should produce better layouts more cheaply and timely than is currently practical. This should result in technically better plant by improving the intrinsic quality of a critical design element and by indirect benefits on other design disciplines. PLS also offers organisational benefits including Quality Assurance, increased confidence in design proposals and better deployment of skilled and expensive designers.

Some of the key benefits that PLS offers are described below. These benefits are closely linked in practice and could be argued to be a number of manifestations of the same basic effect. They are presented separately for clarity nonetheless.

Layout Quality Improved

There is significant informal evidence that layouts designed manually are likely to be sub-optimal. The positions of the 42 items of equipment in the test process were governed by 1200 constraints. This number would prevent a layout engineer considering all the constraints in a practical time. The engineer is limited to identifying the most significant constraints and manipulating these. This behaviour is actually common in design activities [Simon 1969]. This can lead to important constraints being left unsatisfied, not because they are inconsistent with others but because they have been overlooked. For example, a senior engineer in a major process industry contracting company once confided to the author that the piping engineers in his company often encountered great difficulties routing pipes through supposedly satisfactory layouts [Thompson 1989]. PLS's knowledge base can identify many more constraints than an engineer would notice. PLS then has the capacity to manipulate all of these constraints. It will not be overwhelmed by their large number. PLS will leave constraints unsatisfied if and only if it explicitly determines that it must do so.

In manual layout, the problem is exacerbated because the layout engineer is likely to show bias in which constraints do catch his attention. It is probably inevitable that the constraints that arise from issues germane to the engineer's discipline, usually piping design, are more likely to be uppermost in the designer's thinking. To ameliorate this, it is common practice for a number of layouts to be designed then reviewed and compared by a multi-disciplinary panel. This increases the likelihood that a design will arise that is either satisfactory to all disciplines or can at least form the basis of an acceptable solution. In an extreme case known to the author, albeit a particularly difficult design, engineers from a range of specialisations produced thirty two layouts before they achieved a generally acceptable solution [Thompson 1989]. PLS offers the opportunity to avoid this bias. Knowledge can be elicited from a range of experts and pooled in the knowledge base. During elicitation, contention can be identified. PLS will then consider all this knowledge equally, irrespective of its source. Nonetheless, PLS can be deliberately configured to devise a solution which emphasises a specific issue.

Quality Assurance Implemented

Quality Assurance schemes require an approved procedure for a task to be established. The procedures as laid down must then be applied rigorously and exhaustively whenever the task is carried out. PLS offers a formal means to implement Quality Assurance in layout design. PLS follows a structured and formalised procedure which it can apply to plant after plant. PLS also applies all knowledge in its knowledge base that is salient to a particular layout design correctly and exhaustively. All design requirements that should be met will be identified in the form of constraints and given due consideration. Furthermore, as it works, PLS records sufficient information in its database to demonstrate that this has been done. This audit of the layout is an equally important aspect of Quality Assurance. Clearly, PLS cannot apply knowledge that is absent from its knowledge base. However, the knowledge base can be inspected, verified and amended if necessary. Conversely, it is not possible to verify the knowledge and procedures that a layout engineer uses. It is unrealistic to assume that an engineer will apply all his voluminous knowledge. It is not practical to expect the engineer to provide an audit of the layout. The book-keeping overhead would stifle his progress.

Layout Integrated With Computerised Design

Conceptual layout has been described as an "island of man-power in a sea of automation" [Madden 1990]. It remains essentially a manual task due to the current lack of appropriate computer systems. As such, it integrates poorly with the computerised design and data recording systems widely adopted by the other design disciplines. The layout designer must extract his precursor process data from this integrated environment, manipulate these data manually, and then re-enter his results into the software environment. Conceptual layout links the process and plant design, but lies outside the data management mechanisms of the integrated software environment used by the preceding and subsequent disciplines. This compromises the Quality Assurance of the design as a whole. The data transfers are also error-prone and time-consuming.

Any computer system used in conceptual layout clearly has the potential to bridge this gap. The interface from PLS into one three-dimensional modeller and piping design system, CADCentre's "PDMS", demonstrates that PLS has the potential to meet this need. Interfaces to a number of other similar modellers would be needed if PLS were to be used in commercial design offices. Similarly, interfaces from process design databases and other data sources would also be beneficial, although it is much less common for process designers to use these databases than for plant engineers to use modellers.

Substantial Time Savings

PLS took approximately 8 hours to generate a layout for the test process from the initial data. Typically, an experienced layout engineer would require at least 3 days to attempt a layout of similar complexity [Woodland 1989]. Clearly, PLS has the potential to reduce the time that highly skilled and expensive specialist engineers must spend developing a layout. This is intrinsically beneficial, especially because PLS does not compromise the quality of the layout to achieve these savings. However, computer systems which automate other process design tasks can also produce their results significantly faster than an unaided engineer. It is widespread, if not universal, practice to use these systems to explore more options and develop a design further in the time available rather than use them to save manhours. Process simulators are a notable example of this. These are used to explore perhaps ten process alternatives in the time that a process engineer would need to design the first. It is thought likely that PLS would be used similarly to enhance the quality of layouts.

Two factors must be considered when comparing the speed of PLS and an unaided layout engineer. Firstly, the times quoted for both PLS and the engineer do not include the time taken to collate the initial process data. The time quoted for PLS does not include the time taken to enter the data either. If PLS were interfaced to a process design database, no collation would be required and minimal time would be required to transfer the data into its database. Thus, PLS's speed advantage would be increased. Secondly, the MicroVAX hardware and VAXLisp software were chosen to develop PLS because they could be made available to the author at no cost. Neither the hardware nor software could be considered to be appropriate to this work. Current workstations are at least one order of magnitude faster than a MicroVAX. The latter was not a recommended platform for LISP even when it was competitive. VAXLisp exacerbates this difference because it is very inefficient in comparison to other vendor's LISP products. Indeed, VAXLisp has now been withdrawn. Thus, it is reasonable to assume that PLS would lay out the test process in under an hour if running on appropriate hardware and supporting software.

Engineer's Time Used Better

Much of the layout engineer's time is spent conducting tasks which are routine and highly structured. Examples include the initial physical sizing of equipment and the calculation of sufficient elevation to provide a required NPSH for a pump. PLS encodes these routine tasks and applies the knowledge automatically. The engineer is freed to concentrate on the creative and more ambitious components of layout design. This makes better use of the valuable and limited time of highly-skilled specialists.

Rational Review and Amendment Supported

A completed layout design is subject to discussion and critical assessment from within the design team as part of its optimisation. The plant design engineers whose work is dependent on the layout also review and criticise it. Typically, a plot plan or crude three-dimensional model is the only product of the layout engineer's work. In particular, there is no record of which layout constraints were considered by the engineer, let alone which constraints conflicted with one another and what compromises had to be made. This would not be feasible in the technique currently used by the layout engineer.

Generally, in design, a structured history is regarded as important to show how changes at one point might impact on the design as a whole [Kelly 1984, Singh 1983]. Similarly, any design decision points provide scope for exploring alternative solutions. Records of the decisions made will suggest how potentially fruitful alternatives might be generated [Balzer 1984]. Changes in the design and the generation of alternatives forms the basis of design optimisation. The highly informal technique for conceptual layout cannot provide the records to support this. This hampers the review and amendment of proposals.

Furthermore, a design history will serve to verify a solution. It will demonstrate that all significant constraints have been considered and will expose value judgements made while comparing their importance [Lam 1983, Barrow 1984]. A completed layout is likely to be inspected and analyzed by regulatory bodies concerned with safety and environmental aspects of the design. The lack of a design history for a layout precludes formal verification. In the extreme, this might conceal serious flaws in the design. It will certainly slow the acceptance of the proposal.

PLS records all constraints that it identifies and retains them in its database for subsequent inspection. It also records inconsistencies amongst constraints and which of the constraints PLS chooses to satisfy. Together, the records of the constraints and the inconsistencies effectively constitute a design history for the layout, something which the manual technique cannot provide.

Data Provided To Process Designers

Approximately 60% of the cost of a general process plant is directly dependent on the layout (*eg* piping, structural and building costs). The proportion is higher in certain branches of the industry. At present, factorial techniques are used early in the project to approximate the effects of the layout on cost estimates. Factorial techniques are generally accepted to produce estimates with a precision of ±25% and only represent typical or average circumstances. For example, process equipment containing highly flammable reagents or solvents is often more widely spaced than otherwise. This allows releases to disperse to prevent ignition or to minimize the impact of an ignition should one occur. Selecting such highly flammable process media may improve process performance. Factorial estimates will not reflect the impact of the increased spacing requirement on piping and land costs. This limits the resolution with which alternative processes can be compared and introduces economic risk into the project.

Process alternatives are also evaluated against other criteria such as macroscopic safety issues, the plant's space requirement if space is at a premium as in offshore plant, and so on. The choice of a flammable solvent may be precluded by the likelihood of a release drifting to a potential source of ignition before it has dispersed sufficiently to be non-flammable. Unit operations requiring physically large, but relatively cheap items of equipment are likely to be specified in preference to the converse if space allows. For example, a gravity settler is likely to be orders of magnitude larger than a centrifuge performing the same duty, but markedly cheaper. Similarly, process conditions may force the use of larger equipment types. Tubular heat exchangers are generally larger than compact plate heat exchangers for the same heat load, but plate exchangers are limited to low pressure operation. Limitations on the space available might enforce the selection of one process. Accurate technical assessments of process alternatives are impeded by the lack of a layout. This introduces technical risk into the design.

The process engineer is forced to make imprecise evaluations because it is not economically practical to generate a layout manually early in the project. If a process alternative being studied is rejected, the layout will be discarded and the effort invested in designing it will be lost. Process engineers could base economic and technical evaluations on a layout generated by PLS earlier in the project than currently feasible. Approximate layouts are entirely satisfactory for this, commensurate with the quality of process data on which they are based and the purpose for which they are intended. The time invested in a layout generated by PLS is minimal so it is inexpensive to discard the layout after the evaluation. PLS's knowledge base also makes the skills of layout engineers available to process engineers to remove the need for specialist support for these tentative designs.

Once a process alternative has been selected and design has progressed, the process engineers commence detailed design of the equipment items. This requires three-dimensional data in some cases. For example, a pump is sized to deliver sufficient head into its discharge vessel. The elevation of the vessel and length of discharge pipework are required for the calculation of the required head. It is common practice to commence the detailed design of the equipment before a layout has been designed. The process engineer must assume values for the three-dimensional data. To minimise rework of the design, factors of safety are incorporated into these assumed values. This inevitably leads to over-design and a plant that is less economical than the optimum. The process engineer might use PLS to generate provisional layouts quickly and read the three-dimensional data that he needs from these. Although these data are approximate, they should be more precise than assumed values. The process engineer can design the equipment with more precision and thereby reduce the factors of safety.

Widespread Use of PLS

It is now important to consider whether the philosophy embodied in PLS will allow it to have widespread use.

10.2 PLS as a General Tool

Great care was taken to avoid building features into PLS that were specific to the test process. The benefits listed above are only worthwhile if PLS can be used to design layouts for a wide range of processes, even from different branches of the process industry with their own requirements and practices. PLS was deliberately designed to do this. However, the test process includes a diverse range of layout issues and problems which are representative of the range that would be expected across processes and industry sectors. This provides powerful pragmatic evidence of the generality that was designed into PLS.

10.3 Problem Size

The test process is a reasonably complex process that typically would occupy one plot. Larger than this, a process would be broken into plots which would be laid out effectively separately. Thus, the test process showed PLS undertaking a layout that was as large as or larger than the majority of practical layout problems.

PLS would still reach a solution if it were required to lay out a process that is larger than the typical. The increased problem size would not cause it to break down. This is because PLS reaches its solution by solving a series of localised and almost independent sub-problems. PLS considers a small number of entities concurrently. The size of each sub-problem is independent of the overall size of the process. Thus, PLS has to solve more problems of the same size rather than larger problems when laying out a larger process. The total time taken by PLS varies a little more than linearly with the number of independent problems. Note that the "nuisance effect" of an increased number of entities occupying more of the computer's memory even when they are not being reasoned about is actually insignificant. The procedure to form groups will divide a process into plots automatically.

Performance might degrade as the complexity of a problem increases. The test process was deliberately analogous in complexity to the problems that PLS might solve in a practical working environment so no extrapolation is needed. The complexity depends on the average number of constraints that relate any pair of entities. The knowledge base used during the trials identified all the frequently encountered constraints. The knowledge base would have to be extended to a level of completeness where it would also identify those constraints that arise infrequently if PLS were to be used commercially. However, this would not introduce significantly more constraints than PLS reasoned about in the test runs and would not significantly increase the complexity of the problem.

10.4 Sufficiency of Knowledge

It would be unrealistic to expect PLS's knowledge base to exhaustively cover any arbitrarily chosen process given the widely disparate practices, issues and even idioms observed across the domain as a whole. Each sector of the process industry has its own distinctive layout constraints. Indeed, companies adopt styles of layout that emphasise the practices and factors that they consider to be important. Thus, although PLS does not currently contain sufficient knowledge to allow it to lay out a wide range of plants, this does not indicate a lack of generality of the approach.

To extend the knowledge base, a user must be able to add knowledge efficiently and easily. PLS provides appropriate representations and reasoning strategies for any knowledge that a user might wish to add. PLS provides sufficient constructs to represent layout knowledge and reasoning techniques to apply it. These were certainly sufficient to represent all of the diverse knowledge elicited for the test process in forms that were appropriate and elegant.

10.5 Generality of Approach

PLS uses rules to identify FRs and create frames to record them in its database. The FRs re-express the constraints implicit in a process in terms of the fundamental epistemology of plant layout, such as proximity, adjacency or alignment. Thereafter, PLS reasons almost exclusively about the FRs to generate its solution. Effectively, the FRs present the spatial synthesis algorithms with a totally generic restatement of the layout problem. The algorithms are independent of any specific problem that might be set to it.

This is crucially important. The FRs effectively comprise an interface between the fairly specific rules that identify them and the spatial synthesis algorithms of PLS. Users can extend the knowledge base readily to configure PLS to the style of layout appropriate to their needs. They need only add knowledge about items of equipment and constraints that are peculiar to their needs. The spatial synthesis algorithms of PLS provide a constant and sufficient underlying environment to support these extensions.

10.6 Further Work

The author concentrated on the research required to develop a generic problem solving structure for PLS. Effectively, this was the first phase in developing an Expert System to be used commercially to generate layouts from process data. This research phase was the author's original work and is reported in this thesis. The specific objectives for this work, stated in Section 1.1, were achieved in full and the research phase of this work is now complete.

Further work is needed before PLS can be made available for use widely in practice. This work would reflect the differences between basic research and commercial software implementation. The principles established in the research phase would be incorporated in a formally engineered version of PLS. No additional concepts need be developed. If sufficient professional software engineering skills were brought to bear, PLS would develop into a commercial system. This Section highlights the major tasks that must be carried out.

10.6.1 Quality Assured Implementation

The prototype of PLS is principled, rational and self-consistent. In particular, PLS has the potential to generate Quality Assured layouts but it can only realise this potential if it is re-implemented subject to Quality Assurance itself. Quality Assurance must be adopted from the outset of the implementation of any software. It cannot be grafted on to a completed program.

10.6.2 The User Interface

The user interacts with the prototype of PLS while it is running by setting and inspecting LISP forms via the LISP listener and debugger. The initial process data are specified in a text file that is loaded into PLS. The knowledge base and PO Tables are defined elsewhere then compiled and loaded. The results of a run are assessed by inspecting printed listings or by viewing three dimensional PDMS models constructed from PDMS input files written by PLS.

Clearly, a user requires intimate knowledge of the detailed implementation of PLS to interact with it. This is entirely satisfactory in a prototype. However, PLS requires a properly engineered user interface, probably predominantly graphical, before it can be used practically.

10.6.3 Expanding the Knowledge Base

The knowledge base constructed during the experimental work comprises a representative sample of the knowledge employed by layout engineers. It is intended specifically to support testing of PLS. This is consistent with the orientation of the work to date towards researching the basic principles of the system. However, PLS requires a much broader knowledge base before it can be released commercially.

The majority of layout issues and constraints are common across all sectors of the process industry. A liquid will always flow downward under gravity whether it is a sterile medium in a pharmaceutical plant or the heavy ends of a crude oil distillation. Each user should then be able to invest their effort in true customisation and would not have to replicate the efforts of other users capturing the same common issues.

The side effects of a knowledge elicitation exercise such as this could also be highly beneficial. The last major review of layout practice was carried out by the Institution of Chemical Engineers Working Party between 1978 and 1983 [Mecklenburgh 1985]. The process of building a knowledge base for PLS would expose current practices to review and allow best practices to be identified. The knowledge base itself would be a highly apposite medium to record and communicate these practices.

It is highly recommended that formal methods, or preferably a computerised knowledge elicitation tool [*eg* Shadbolt 1993], should be used to support a major knowledge elicitation exercise such as this if it were carried out.

10.6.4 Refined Handling of Site Topography

The topography of the imaginary site for the test plant imposed two constraints on the layout which PLS correctly reflected in its layout design. The initial data included the positions of a bulk tank farm from where the acetone and monomer would be taken (North of the plot) and the product warehouse (South of the plot). PLS then oriented the layout to minimise the distances that the raw materials and finished products would need to be transported.

PLS must be able to generate a layout that responds to the topography of the site and the environs of a real plant. The examples included in the test process only impose fairly loose constraints on the layout. However, more restrictive constraints could also be imposed. For example, if PLS were used to generate a layout for a plant revamp, it is likely that the positions of certain items of equipment would be fixed. If PLS were laying out an extension to an existing plant, the positions of the existing equipment would constrain the positions of the new items. Similarly, PLS might be forced to develop a layout that is internally sub-optimal but that fits into a restricted space, perhaps if it were laying out a plant to be built within an existing facility.

At present, PLS cannot respond to more specific constraints imposed by site topography such as those exemplified. Minor refinements are required in the plan layout algorithm itself and the plan layout scripts.

The plan layout algorithm must be refined to allow PLS to reflect the impact of external positional constraints. PLS must identify the entities that are constrained in their position by site topography. It should then consider whether it is appropriate to make these entities roots of plan trees. Thereby, PLS will be able to develop the layout around these fixed points.

Both the plan layout algorithm and the plan meta-knowledge must be refined to allow PLS to design layouts that fit into restricted space. When PLS identifies that an overall space constraint has been violated, it must identify the groups that can be shortened along the critical axis, apportioning space saving amongst more than one group if necessary. The plan layout scripts must then position the members of these groups sub-optimally from a local perspective to reduce the length of the group. This might be at the expense of increased length on the other axis or by elevating entities above the minimum set by process needs.

It is important to note that the basic principles of PLS's approach to plan layout would be retained. The modifications described above are no more than refinements to the existing approach.

10.6.5 Refined Back-Tracking and Multiple Solutions

When PLS cannot decide which of a number of contradicting constraints should be satisfied, it makes an arbitrary selection. In principle, PLS could generate solutions in which each of the constraints are satisfied in turn and generate alternative designs. At present, only the first solution is developed to completion. This simplification was made purely to speed development of the PLS prototype.

It is widely held that there is rarely a single "best" solution to a layout problem. A number of solutions with particular strengths but also weaknesses are likely to be approximately as good. Each solution will satisfy a different sub-set of the constraints but all sub-sets will be comparable in overall satisfaction. However, one solution will satisfy a sub-set that best fits the emphasis specific to the job and this solution will be considered the "best" in this case. PLS must be able to generate alternative designs to emulate this behaviour if it is to be used practically.

Assumption Based Truth Maintenance Systems or ATMSs [de Kleer 1986a, 1986b, 1986c] allow Expert Systems to hold simultaneously a number of data sets which are individually consistent but mutually contradictory. ATMSs are frequently used to manage concurrent development of multiple solutions. For example, "KBDS" [Banares-Alcantara 1994] develops process flowsheets using an ATMS to hold a number of process options in its database. An ATMS could be incorporated into PLS to allow it to generate all of the best designs. An ATMS is not the only mechanism that can achieve this. Indeed, an ATMS is large and complex in its own right and generates voluminous "house keeping" data. Nonetheless, an ATMS is a good choice in this case because PLS also requires some form of Truth Maintenance System to increase its efficiency. An ATMS would clearly perform well as PLS's truth maintenance system because this is its intended purpose!

Chapter 11 : Summary and Conclusions

An approach to automatic conceptual process plant layout by Expert System has been developed. This approach is generic and supports plant layout in breadth. The approach is applicable to a wide range of processes from a wide range of process industry sectors. The approach also covers all aspects of conceptual layout, from estimating the size of the process equipment from minimal process data through to calculating their elevations and deriving their plan positions.

The approach has been implemented successfully as "Plant Layout System" or "PLS". It includes a supporting knowledge base which was elicited and encoded during this work. The generic approach and the implementation are both designed so that knowledge that is specific to a sector of the process industry, or even a company, can be readily added.

PLS has been applied to a test process of typical size and complexity, and which encompasses a representative range of layout issues and problems. It generated a layout that is entirely satisfactory and conventional from an engineering viewpoint. The approach to automated plant layout and its implementation in PLS is considered to be generally applicable.

All three objectives of this work, stated in Chapter 1, have been achieved. The work shows that it is practical to use an Expert System to generate conceptual process plant layouts in a commercial design environment. The work also indicates an approach by which such an Expert System can be implemented. Such an Expert System would bring substantial benefits to its users which include:

- Improved quality of layouts,
- Quality Assured layout,

- Layout integrated with computerised design systems,
- Layout data provided to process engineers,
- Potential for substantial time savings,
- Rational review and amendment of layout proposals
- Better use of layout engineers' time.

The approach developed during this work adopts the same philosophy and assumptions as a layout engineer. The approach places the same importance as does the engineer on the various design objectives for the layout. Both attempt to design the same features into the layout. This is why the approach succeeds and a highly important aspect of the research work was to elicit and understand these engineering principles. However, the approach to layout by Expert System and the engineer's approach differ markedly in all but the overall structure, even though both achieve the same results. It was found during this work that the engineer uses an approach that would be difficult to emulate in a computer program, even an Expert System. The engineer's approach is highly fluid, almost unstructured, and relies heavily on human spatial reasoning and visualisation faculties. An important opportunity to enhance plant layout practice would have been missed had the engineer's approach been emulated in any case. The approach to layout by Expert System imposes a formal procedure on a currently *ad hoc* task and allows the layout to be audited and inspected. This introduces Quality Assurance to plant layout, never feasible before.

The approach to layout by Expert System is perhaps the most important result of this work. It is appropriate, therefore, to summarise the approach, as follows.

- Design constraints in plant layout dictate relative positions of entities. The standard AI technique of constraint propagation represents the design constraints very naturally. This technique is highly efficient and appropriate for laying out process plant. The design constraints are inferred by generic knowledge, elicited from layout engineers. This ensures that this key step is rigorous and exhaustive, critical for layout quality and Quality Assurance.

- Constraints are divided into two categories, each with markedly different properties. These are called "Functional Relationships" or "FRs" and "Spatial Relationships" or "SRs". FRs record design constraints and drive the spatial synthesis. They form an interface between the layout knowledge and the generic spatial synthesis algorithms. The spatial synthesis algorithms propagate the FRs to form SRs, which record relative positions of the entities within the layout. The constraints are propagated by algorithms which are specific to their task for efficiency.

- All FRs are recorded in the database persistently, even those which cannot be satisfied. They can be inspected to ensure that all important constraints imposed by a process can be satisfied in a layout. All contradictions between constraints are also recorded explicitly. These records provide a design history and highlight possible design alternatives. Together, the records of the FRs and contradictions provide an audit for the layout solution, essential for Quality Assurance. An Expert System has the capacity to consider all germane constraints whereas an engineer is restricted to those which are key.

- Process plant layout is invariably over-constrained. Constraints are selected to be relaxed using a qualitative approach which is unique. This technique expresses the highly non-linear relationship between the number of constraints contradicted and the importance of satisfying them.

- The layout is synthesised in a series of separate but complementary phases into which the task is decomposed. The first phase is a pre-processing phase in which the input data are checked, the physical sizes of the equipment are estimated and the FRs are inferred and recorded. The elevations of the equipment items are calculated in the second phase, equipment items are collected into groups in the third and the plan positions are derived in the fourth. This decomposition extends the range of applicability by allowing PLS to lay out larger or more complex plants and does not compromise the quality of the layout.

- Process equipment items which have an affinity with one another are collected into groups. These groups are then collected to form increasingly more expansive groups using three separate algorithms. The first collects entities which can share a service or facility. The second uses a powerful abstract relationship which collects the members of a process unit or process stage. The third collects entities which demand segregation and which are mutually compatible. The techniques developed to form groups during this work are unique in their efficacy and sophistication.

- The groups greatly simplify plan positioning by reducing the number of constraints and entities which must be manipulated simultaneously. They allow some constraints to be neglected during plan positioning because group membership ensures that they are already fully satisfied. They provide an effective treatment of coarse spatial requirements and partial satisfaction of constraints and they divide the plant into sections in which powerful meta-knowledge can be applied to assist reasoning during plan positioning.

- A very typical frame based representation is used for all entities, including those derived during problem solving, such as constraints and groups. Effectively all process plant layout knowledge is procedural, so the frame language dispenses with inheritance of default values. Domain knowledge is expressed in data driven rules, which are invoked explicitly from meta-rules for efficiency. The rule language is also used to implement the constraint propagation algorithms, for which the explicit control is essential. Procedural Attachments are also provided because some layout knowledge is better applied in goal directed reasoning.

PLS is the prototype of an Expert System which has the potential to yield substantial benefits to the process industry. If a formally engineered version of PLS were constructed incorporating the principles established in this work, it would develop into a commercially usable system. This only requires sufficient professional software engineering skills to be brought to bear - no additional concepts or principles need be developed.

References

Aho, AV, Hopcroft, JE and Ullman, JD (1983) *Data Structures and Algorithms*, Addison-Wesley, Massachussetts.

Akin, O (1978) How Do Architects Design? In *Artificial Intelligence And Pattern Recognition In Computer Aided Design*, Latombe, JC (ed), North-Holland Publishing Company.

Al-Asadi, HD (1980) *Computer Aided Layout Of Chemical Plant*, PhD Thesis, University of Wales.

Alexander, C (1968) Major Changes In Environmental Form Required By Social And Psychological Demands. In *Proceedings Of The Second International Symposium on Regional Development.*

Allan, J (1984) Towards a General Theory of Action and Time. *Artificial Intelligence*, **23**, no 2, 123-154.

ANSI/ASME (1984) *Specification For Horizontal End Suction Centrifugal Pumps For Chemical Process*, B73.1M-1984.

ANSI/ASME (1984) *Specification For Vertical In-line Centrifugal Pumps For Chemical Process*, B73.2M-1984.

API (1986) *Centrifugal Pumps For General Refinery Service*, API Standard 610.

Archer, LB (1968) *The Structure Of The Design Process*, Royal College Of Art, London.

Asimow, W (1962) *Introduction To Design*, Prentice-Hall, New Jersey.

Atkinson, N (1987) Taking 3D Modelling Into Conceptual Design. *Process Engineering*, November, 53-55.

Balzer, R (1984) Capturing The Design Process In The Machine. In Proceedings of Rutgers Workshop on Knowledge-Based Design Aids: Models Of The Design Process.

Banares-Alcantara, R (1994) Applications of Knowledge Based Agents in the Support of Process Design. In *Proceedings of Applications of Knowledge Based Systems in the Process Industry*, Institution of Chemical Engineers, Manchester.

Barrow, HG (1984) Proving The Correctness Of Digital Hardware Designs. In *Proceedings of AAAI-83.*

Bausbacher E and Hunt R (1993) *Process Plant Layout and Piping Design*, Prentice Hall, New Jersey.

Baykan, CA and Fox, MS (1986) An Investigation of Opportunistic Constraint Satisfaction In Space Planning. In *Proceedings of AAAI-86.*

Bobrow, D and Winograd, T (1977) An Overview of KRL: A Knowledge Representation Language. *Cognitive Science*, 1, 3-46.

Brachman, R (1978) *A Structural Paradigm For Representing Knowledge*, Report Number 3605, Bolt Beranek and Newman, Cambridge, Massachussetts.

Brachman, R (1979) On The Epistemolgical Status Of Semantic Networks. In *Readings In Knowledge Representation*, Brachman, R and Levesque, H (eds), Morgan Kaufmann, California.

Brachman, RJ, Amarel, S, Engelman, C, Engelmore, RS, Feigenbaum, EA and Wilkins, DE (1983) What Are Expert Systems?. In *Building Expert Systems*, Hayes-Roth, F, Waterman, DA and Lenat, DB (eds), Addison-Wesley, Massachusetts.

Bredbury, J (1986) *Personal Communication.*

Briggs R (1987) Approaching The Paperless Office For Project Engineering. *Process Engineering*, July, 44-46.

Bryant, DA and Dains, RB (1977) Models Of Buildings In Computers: Three Useful Abstractions. *Industrialization Forum*, **8**, Number 2.

Bush, MJ and Wells, GL (1972) The Computer Generation Of Unit Plot Plans For Chemical Plant. In *Institution of Chemical Engineers Symposium Series Number 35.*

Charniak, E (1978) With Spoon in Hand This Must be the Eating Frame. In *Proceedings of TINLAP-2.*

Chieng, W-H and Hoeltzel, DA (1987) A Generic Planning Model For Large Scale Integrated Engineering Design. In *Knowledge Based Expert Systems In Engineering: Planning And Design*, Sriram, D and Adey, RA (eds), Computational Mechanics Publications, Southampton.

Cinar, U (1968) Facilities Planning: A Systems Analysis And Space Allocation Approach. In *Spatial Synthesis In Computer Aided Building Design*, Wiley, New York.

Coyne, RD and Gero, JS (1985) Knowledge And Sequential Plans. *Environment And Planning B*, No 12, 401-418.

Coyne, RD, Rosenman, MA, Radford, AD, Balachandran, M and Gero, JS (1991) *Knowledge-based Design Systems*, Addison Wesley, Massachussetts.

Craft, J (1985) Designing a Database for the Process Engineer. *Process Engineering*, May, 47-51.

Davis, E (1987) Constraint Propagation With Interval Labels. *Artificial Intelligence*, 32, 281-331.

DIN (1979) *Chemische Apparate: Volumen und Gewichte, Begriffe, Nannvolumenstuffen*, DIB 28100.

Earl, CF (1980) Rectangular Shapes. *Environment And Planning B*, No 7, 311-342.

Eastman, CM (1970) The Analysis Of Intuitive Design Processes. In *Emerging Methods In Environmental Design And Planning*, Moore, GT (ed), MIT Press, Cambridge, Massachusetts.

Eastman, CM (1972) Heuristic Algoritms For Automated Space Planning. In *Proceedings of The Second International Conference On Artificial Intelligence.*

Eastman, CM (1973) Automated Space Planning. *Artificial Intelligence*, No 4, 41-46.

Evans, FL (1979) *Equipment Design Handbook For Refineries and Chemical Plant*, 2nd edn, Volumes 1 and 2, Gulf Publishing Company, Houston, Texas.

Fahlman, SE (1975) A PLanning System For Robot Construction Tasks. *Artificial Intelligence*, 5, 1-49.

Fine, B (1965) Piping Design In Chemical Plant. In *Institution of Chemical Engineers Symposium Series, No 4.*

Flemming U (1978) Wall Representations Of Rectangular Dissections And Their Use In Automated Space Planning. *Environment And Planning B*, No 5, 215-232.

Forgy CL (1982) Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, 19, 17-37.

Foulds LR (1983) Techniques For Facilities Layout: Deciding Which Pairs Of Activities Should Be Adjacent. Management Science, 29, No 12, 1414-1426.

Foz, A (1973) Observations On Design Behaviour In The Parti. In *Proceedings of The Design Activity Conference.*

Francis RL and White JA (1974) *Facility Location And Layout: An Analytical Approach*, Prentice-Hall, New Jersey.

Frayman, F and Mittal, S (1987) COSSACK: A Constraint-Based Expert System for Configuration Tasks. In *Knowledge Based Expert Systems In Engineering: Planning And Design*, Sriram, D and Adey, RA (eds), Computational Mechanics Publications, Southampton.

Freeman, P and Newell, A (1971) A Model For Functional Reasoning In Design. In *Proceedings of The International Joint Conference on Artificial Intelligence.*

Fox, MS (1983) *Constraint Directed Search: A Case Study Of Job Shop Scheduling*, Pitman, London and Morgan Kaufman, California.

Fox, MS (1986) Observations On The Role Of Constraints In Problem Solving. In *Proceedings of Sixth Canadian Conference on Artificial Intelligence.*

Gavett, JW and Plyter, NV (1966) The Optimal Assignment Of Facilities To Locations By Branch And Bound. *Operations Research*, 14, 210-232.

Gero JS and Coyne RD (1987) Knowledge Based Planning As A Design Paradigm. In *Proceedings of IFIP Working Group Working Conference On Design Theory For CAD.*

Gilleard J (1978) LAYOUT - A Heirarchical Computer Model For The Production Of Architectural Floor Plans. *Environment And Planning B*, No 5, 233-241.

Grason J (1968) A Dual Linear Graph Representation For Space Filling Location Problems Of The Floor Plan Type. In *Emerging Methods In Environmental Design And Planning*, Moore, GT (ed), MIT Press, Massachusetts.

Grason J (1971) Approach To Computerized Space Planning Using Graph Theory. In *Proceedings Of The Eight Design Automation Conference.*

Gunn, DJ (1970) The Optimised Layout Of A Chemical Plant By Digital Computer. *Computer Aided Design*, 11, Spring.

Gunn, DJ and Al-Asadi, HD (1980) Computer Aided Layout Of Chemical Plant: A Computational Method And Case Study. *Computer Aided Design*, 19, No 3, April.

Hayes-Roth, F and Lesser, VR (1977) Focus Of Attention In The HEARSAY-II Speech Understanding System. In *Proceedings of Fifth International Joint Conference on Artificial Intelligence.*

Hillier, FS and Connors, MM (1966) Quadratic Assignment Problem Algorithm And The Location Of Indivisible Facilities. *Management Science*, **13**, 42-57.

Hink, RF and Woods, DL (1987) How Humans Process Uncertain Knowledge. **AI Magazine**, Fall, 41-53.

Jain, D, Chatterjee, M, Unemori, A and Thangam, N (1992) A Knowledge-Based Automatic Pipe Routing System. In *Proceedings of 1992 ASME International Computers in Engineering Conference and Exposition.*

Johnson, TE and Weinzapfel, G (1970) *IMAGE: An Interactive Graphics-Based Computer System For Multi-Constrained Synthesis*, Report of Department Of Architecture, MIT, Massachussets.

Kant, E and Newell, A (1982) Naive Algorithm Design Techniques: A Case Study. In *Proceedings of The European Conference on Artificial Intelligence.*

Kelly, Van E (1984) The CRITTER System - Automating Critiquing Of Digital Circuit Designs. In *Proceedings of 21st Design Automation Conference.*

Kern, R (1977) *Plant Layout*, McGraw-Hill, New York.

de Kleer, J (1986a) An Assumption-based TMS. *Artificial Intelligence*, **28**, 127-162.

de Kleer, J (1986b) Extending the ATMS. *Artificial Intelligence*, **28**, 163-196.

de Kleer, J (1986c) Problem Solving with the ATMS. *Artificial Intelligence*, **28**, 197-224.

Kletz, TA (1976) *The Application Of Hazard Analysis To Risks To The Public At Large*, Elsevier, Amsterdam.

Krejcirik, M (1969) Computer-Aided Plant Layout. *Computer Aided Design*, **2**, No 1, Autumn, 7-19.

Kuipers, B (1975) A Frame For Frames: Representing Knowledge for Recognition. In *Representations and Understanding: Studies in Cognitive Science*, Bobrow, D and Collins, A (eds), Academic Press, New York.

Kuipers, B (1984) Commonsense Reasoning About Causality: Deriving Behaviour From Structure. *Artificial Intelligence*, **24**, 169-203.

Lam, M and Mostow, J (1983) A Transformational Model Of VLSI Systolic Design. In *Proceeedings of IFIP Working Group 10.2 Sixth International Conference on Computer Hardware Description Languages And Their Applications.*

Lee, RC and Moore, JM (1967) CORELAP - Computerised Relationship Layout Planning. In *Journal of Industrial Engineering*, XVIII, 3, 195-200.

Leesley, ME and Newell, RG (1972) The Determination Of Plant Layout By Interactive Computer Methods. In *Institution of Chemical Engineers Symposium Series, No 35.*

Levin, PH (1964) Use Of Graphs To Decide The Optimum Layout Of Buildings. *The Architect's Journal*, **7**, No 10.

Lew, IP and Brown, PH (1968) Evaluation And Modification Of CRAFT For An Architectural Methodology. In *Emerging Methods In Environmental Design And Planning*, Moore GT (ed), MIT Press, Massachusetts.

Lichtenstein, S and Fischoff, B (1977) Do Those Who Know Also Know More About How Much They Know?. *Organizational Behaviour And Performance*, **20**, 159-183.

Lichtenstein, S, Fischoff, B and Phillips, LD (1982) Calibration Of Probabilities: The State Of The Art To 1980. In *Judgement Under Uncertainty: Heuristics And Biases*, Kahneman, D, Slovic, P and Tversky, A (eds), Cambridge University Press, New York.

Liggett, RS (1972) Floor Plan Layout By Implicit Enumeration. In *Proceedings of Third EDRA Conference.*

Luckman, J (1967) An Approach To The Management Of Design. *Operational Research Quaterly*, 18, No 4, 345-358.

Mackworth, AK (1977) Consistency in Networks of Relations. *Artificial Intelligence*, 8, 99-118.

Madden, J, Pulford, CI and Shadbolt, NR (1990) Plant Layout - Untouched By Human Hand? *The Chemical Engineer*, 24th May, 32-36.

Maher, ML and Fenves SJ (1985) Hi-Rise: An Expert System For The Preliminary Structural Design of High-Rise Buildings. In *Knowledge Engineering in Computer-Aided Design*, North-Holland Publishing Company.

Laboratoires de Marcoussis (1986) *CADOO: A Knowledge Based System For Space Allocation*, Laboratoires de Marcoussis, Marcoussis, France.

Marcus, RI (1986) Multi-level Constraint Based Configuration. *Hewlett Packard Journal*, November, 54-56.

Markus, TA, Whyman, P, Morgan, J, Whitton, D, Mavert, J, Carter, D and Fleming, J (1972) *Building Performance*, Applied Science, London.

Masanori, N (1976) *Development Of A Computer System For 3-D Space Allocation*, Masters Thesis, Department Of Architecture, MIT, Massachusetts.

Maver, TW (1971) PACE1: Computer Aided Building Appraisal. *Architect's Journal*, July 28.

McBrien, A (1989a) Layout Execution. Presented at *Process Plant Layout*, University of Nottingham, 20th to 25th September.

McBrien, A, Madden, J and Shadbolt, NR (1989b) Artificial Intelligence Methods in Process Plant Layout. In *Proceedings of Second International Conference, IEA/AIE-89*, Tullahoma, USA.

McCarthy, J and Hayes, PJ (1969) Some Philosophical Problems From the Standpoint of Artificial Intelligence. In *Machine Intelligence 4*, Michie, D and Meltzer, B (eds), Edinburgh University Press, Edinburgh.

Mecklenburgh, JC (1985) *Process Plant Layout* (ed), 2nd edn, George Godwin, Harlow and Halstead Press, New York.

Metropolis, N, Rosenbluth, AW, Rosenbluth, MN, Teller, AH and Teller, E (1953) Equation Of State Calculations By Fast Computing Machines. *Journal Of Chemical Physics*, **21**, 1087-1092.

Minksy, M (1975) A Framework For Representing Knowledge. In *The Psychology Of Computer Vision*, Winston, P (ed), McGraw-Hill, New York.

Mistree, F, Hughes, OF and Phuoc, HB (1981) An Optimization Algorithm For The Design Of Large, Highly Constrained Complex Systems. *Engineering Optimization*, **5**, 179-197.

Mitchell, WJ, Steadman, JP and Liggett, RS (1976) Synthesis And Optimisation Of Small Rectangular Floor Plans. *Environment And Planning B*, No 1, 37-70.

Montanari, U (1974) Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences*, 7, 95-132.

Moore, JM (1974) Computer Aided Facilities Design: An International Survey. *International Journal Of Production Research*, No 12, 21-44.

Mostow, J (1985) Toward Better Models Of The Design Process. *The AI Magazine*, Spring, 44-56.

Mustacchi, C (1974) Optimal Process Layout By A Branch And Bound Technique. *Ing. Chim. Ital.*, 10, No 12, December.

Muther, R (1961) *Systematic Layout Planning*, Industrial Education Institute, Boston, Mass.

Muther, R (1962) Systematic Layout Planning. *Factory*, August, September and October.

Nadel, BA (1989) Constraint Satisfaction Algorithms. *Computational Intelligence*, 5, 188-224.

Negroponte, N and Grossier, L (1968) URBAN5: A Machine That Discusses Urban Design. In *Emerging Methods In Environmental Design And Planning*, Moore, GT (ed), MIT Press, Massachusetts.

Newman, WM (1966) An Experimental Program For Architectural Design. *Computer Journal*, 9, June, 21-26.

Pereira, LM (1978) Artificial Intelligence Techniques In Automatic Layout Design. In *Artificial Intelligence And Pattern Recognition In Computer Aided Design*, Latombe, JC (ed), IFIP North-Holland Publishing Co.

Perry, RH and Green, DW (1984) *Perry's Chemical Engineers' Handbook*, Sixth Edition, McGraw-Hill, New York.

Pfefferkorn, CE (1975) A Heuristic Problem solving Design System For Equipment Or Furniture Layouts. *Communications Of The Association Of Computing Machinery*, **18**, No 5, 286-297.

Rose, JC, Wells, GL and Yeats, BH (1978) *A Guide To Project Procedure*, Institution of Chemical Engineers.

Rumbaugh, J, Blaha, M, Premerlani, W, Eddy, F and Lorensen, W (1991) *Object-Oriented Modeling And Design*, Prentice-Hall, New Jersey.

Russo, TJ and Tortorella, AJ (1992) Plant Layout - The Contribution Of CAD. *Chemical Engineering*, April.

Ryan, K and Harty N (1990) Supporting Choice and Evaluation in Preliminary Design. In *Proceedings of the Third International Conference AIE/AEI.*

Sahni, S and Gonzalez, T (1976) P-Complete Approximation Problems. *Journal of Association Of Computing Machinery*, **23**, 478-485.

Simon, HA (1969) *The Sciences of the Artificial*, MIT Press, Massachussets.

Seehof, JM and Evans, WO (1967) Automated Layout Design Program. *Journal Of Industrial Engineering*, XVIII, No 12, 690-695.

Schank, RC and Abelson, RP (1977) *Scripts, Plans, Goals and Understanding*, Lawrence Erlbaum, New Jersey.

Shadbolt, NR, Motta, E and Rouge, A (1993) Constructing Knowledge-Based Systems. *IEEE Software*, **10**, No 6, 34-38.

Sharpe, R and Marksjo, BS (1985) Facility Layout Optimisation Using The Metropolis Algorithm. *Environment And Planning B*, No 12, 443-453.

Shuquair, MM (1978) *Studies on Plant Layout*, PhD Thesis, University of Sheffield.

Sileti, CJ (1993) Design of Protein Purification Processes By Heuristic Search. In *Artificial Intelligence in Process Engineering*, Mavrovounotis, ML (ed), Academic Press, San Diego, California.

Simon, HA (1973) Structure Of Ill-Structured Problems. *Artificial Intelligence*, 4, 181-201.

Silverstein, EE and Sun, P (1990) A Constraint Management Tool For Concurrent Engineering. In *Proceedings of the Third International Conference AIE/AEI.*

Simmons, R (1986) Commonsense Arithmetic Reasoning. In *Proceedings of AAAI-86.*

Singh, N (1983) *MARS: A Multiple Abstraction Rule-Based Simulator* Stanford Heuristic Programming Project Report HPP-83-85.

Stallman, RM and Sussman, GJ (1977) Forward Reasoning And Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence*, 9, 135-196.

Standard (1989a) Personal Communication.

Standard (1989b) Personal Communication.

Standard (1989c) Personal Communication.

Steadman, JP (1970) The Automatic Generation Of Minimum-Standard House Plans. In *Proceedings of Second Annual Conference of Environmental Design Research Association.*

Steele, GL, Jr (1985) *Common LISP: The Language*, Digital Press, Massachusetts.

Stefik, M (1981a) Planning With Constraints (MOLGEN: Part 1). *Artificial Intelligence*, **16**, 111-140.

Stefik, M (1981) Planning And Meta-Planning (MOLGEN: Part 2). *Artificial Intelligence*, **16**, 141-170.

Stefik, M and Bobrow, D (1986) Object Oriented Programming: Themes and Variations. *AI Magazine*, **6**, No 4, 40-62.

Syska, I, Cunis, R, Gunter, A, Peters, H and Bode H (1988) Solving Construction Tasks With A Cooperating Constraint System. In *Research and Development in Expert Systems V*, Kelly, B (ed), Cambridge University Press.

Taffe, P (1990) The Inside View To Offshore Design. *Processing*, March, 22-25.

Tubular Exchangers Manufacturers Association (1978) *Standards of the Tubular Exchangers Manufacturers Association*, Sixth Edition.

Thompson, I (1989) Personal Communication.

Waltz D (1975) Understanding Line Drawings of Scenes With Shadows. In *The Psychology of Computer Vision*, Winston PH, (ed), McGraw-Hill, New York.

Whitehead, B and Eldars, MZ (1964) An Approach To The Optimum Layout Of Single-Story Buildings. *The Architects Journal*, **139**, 1373-1380.

Wild, B (1972) *Mass Production Management*, John Wiley, London.

Willoughby, TM (1975) Building Forms And Circulation. *Environment And Planning B*, **2**, No 1, June.

Winograd, T (1975) Frame Representations and the Declarative/Procedural Controversy. In *Representations and Understanding: Studies in Cognitive Science*, Bobrow, D and Collins, A (eds), Academic Press, New York.

Yessios, CI (1972) FOSPLAN: A Formal Space Planning Language. In *Proceedings Of The Fourth Environmental Design Research Conference*, Mitchell, W (ed), UCLA, California.

Armour, GC and Buffa ES (1964) A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities. *Management Science*, January, 294-309.

Woodland J (1989) Personal Communication.

Appendix A : Notation for Example Frames

The following notational conventions were adopted for the example frames presented throughout this thesis.

A frame is delimited by braces "{...}". The first element within the braces, for example S-1, represents the unique number that PLS assigned to each frame in the database. PLS uses this number internally to index and cross-reference frames. For these examples, the numbers have been rationalised to be sequential for clarity. Thus,

```
{S-1
        :
        :
        }
```

represents a frame with the unique number S-1.

A slot of a frame is delimited by square brackets "[...]". The slot name appears first then the slot value second. Thus,

```
[name R104]
```

represents a slot with name "name" and value "R104".

Where the value of a slot is a frame, the frame is written in the same position as an atomic numeric or symbolic value. The slots of the frame are displayed inside its delimiting braces. Thus,

```
[drive
        {S-3  [required-power 100]
              [voltage 440]}
        ]
```

represents a slot with name "drive" and the frame S-3 as its value. The frame S-3 has two slots with names "required-power" and "voltage" respectively.