Formation: Knowledge-Based Layout of Classified Telephone Directories

Gail Anderson, Andrew Casson, Ann Macintosh, Robert Rae,
AIAI, The University of Edinburgh,
80 South Bridge, Edinburgh EH1 1HN

Barry Gleeson, Simon Carter,
Pindar Set Ltd,
Newlands Park Drive, Scarborough, North Yorkshire YO12 6DT

Abstract

Pindar Set Limited, the UK company that originates the Yellow Pages for British Telecommunications plc, has developed a business strategy based upon a generic, knowledge-based, batch layout software suite for use by publishers internationally. Pindar's clients, publishers such as BT, increasingly insist on faster and more flexible responses to requests for changes in the way their books are laid out, and hence the way they look to their customers, the companies advertising in the Yellow Pages. In order to maintain its competitive advantage, Pindar formed a strategic alliance with AIAI at the University of Edinburgh to produce its next generation directory layout solution—Formation.

Formation is a knowledge-based layout system for classified telephone directories. It comprises a graphical user interface to a batch layout system and a style definition system with a library of layout strategies and methods. The system is based on the specialised language LSSL, a fully featured programming language which has been developed at AIAI for use in knowledge-based layout systems.

Although Formation was required to produce classified telephone directories, it was intended to be a much more general system. AIAI has designed a general purpose framework which supports the expression of knowledge about layout requirements in a natural and modular manner. This allows detailed knowledge about the layout of the pages of a book to be considered at a higher level: the book's style. Through the Formation user interface, it is quick and easy to reconfigure a style.

The knowledge-based system has been in trial use with Pindar since January 1996. During this time, Pindar has been able to demonstrate the flexibility of the system to its clients. The approach taken, that is, the design and implementation of a suitable language in which expert knowledge about document layout can be expressed, has been well validated.

Formation currently runs under Windows on PCs, and will be in full production use for BT's Yellow Page directories from September 1996.

1 Problem Description

Founded in Scarborough in 1836, Pindar has been a pioneer in the development of automated systems for directory production through its early involvement in the production of British Telecommunications plc's Yellow Pages classified telephone directories. Pindar has originated the BT Yellow Pages since 1979, employing 150 people over 4 sites in the UK. Today it is involved in the production of classified telephone directories throughout the world.

Pindar's clients, the directory publishers, insist on increasingly faster and more flexible responses to requests for change in designs, usability, layout and aesthetics of pages demanded in turn by their customers, the businesses advertising in the Yellow Pages. Pindar, realising the need to maintain competitive advantage, formed a strategic alliance with the Artificial Intelligence Applications Institute, AIAI, at the University of Edinburgh to produce its next generation directory layout solution—Formation.

BT's Yellow Pages provide regionally based classified business directories. In the UK, a typical directory of 1,500 pages may contain 30,000 businesses classified alphabetically under 3,000 different headings, with a further 5,000 display advertisements. The print run for a directory will depend on the particular region: this would only be around 25,000 copies for the Isle of Man, but over half a million for Glasgow South. As there are about 75 different directories that need to be printed and distributed, it is important to lay out the directory efficiently and balance the publisher's layout requirements against production considerations such as keeping down the page count with its associated printing costs, as well as reducing the weight of the book with its associated transport costs.

Automated systems which address these issues are in regular use throughout the world [1]. However, Pindar required greater flexibility and faster throughput than existing systems offered. AIAI, having experience in developing a number of layout and constraint based applications, was a natural choice to partner Pindar. In addition to the provision of software, Pindar required a full understanding of the knowledge-based system, so that it could maintain the system in future, and develop it further without needing to call on external support. As a Technology Transfer organisation, AIAI understood this requirement and was able to provide visiting worker facilities through which Pindar staff could work with experienced members of the Formation project team.

2 Application Description

2.1 Formation

The Formation system has been developed to meet the requirements of Pindar's existing and future clients throughout the world. This paper focuses on the experience gained with Formation in producing classified telephone directories to meet the needs of two different markets: the UK through BT's Yellow Pages and the USA through the software publisher Yellow Magic Incorporated. Although both have similar basic requirements, BT is a single publisher with a consistent house style that is uniform across the UK, whereas Yellow Magic has to support a wide variety of regional publishers, each wishing to be differentiated from the others through its own individual house style. Though both publishers address the same basic problem, the USA demands greater flexibility and generality.

Formation was designed to satisfy two important requirements: flexibility in describing or modifying the details of a particular style of layout, and high throughput.

The Formation user interface, under Windows95, is shown in figure 1, below. This

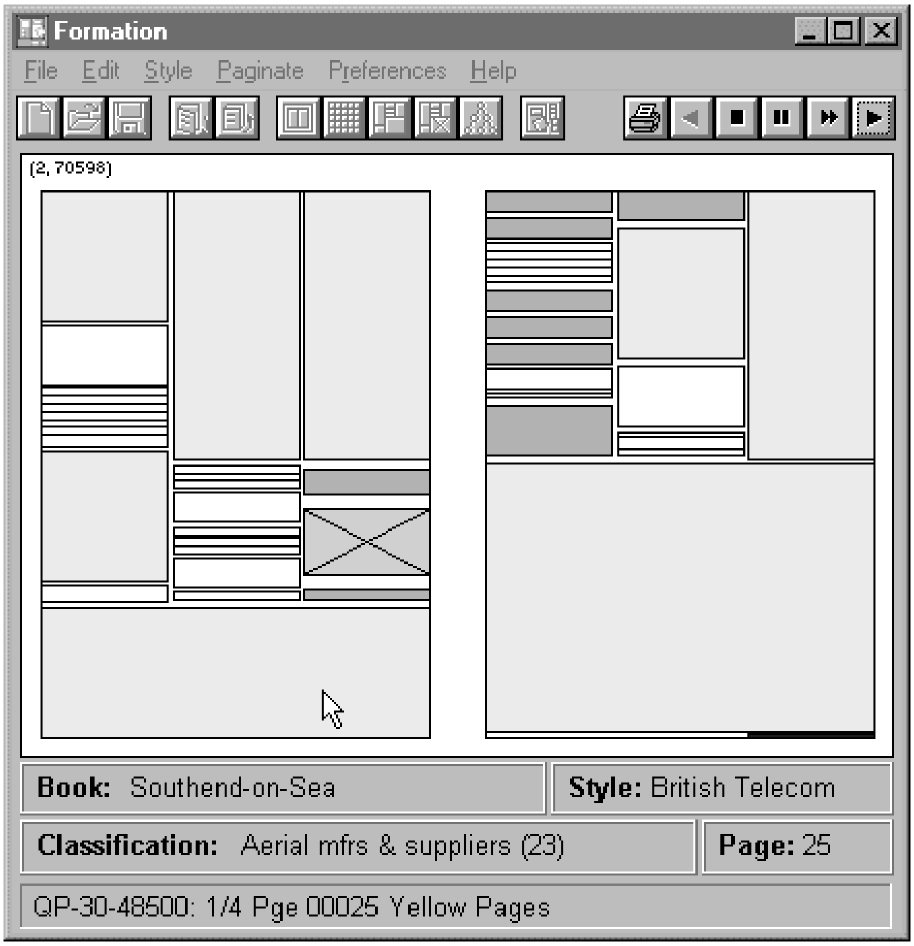

Figure 1: The Formation operator interface under Windows95

gives the operator a unified environment for performing the various tasks required for producing a book from its data. Formation provides interactive facilities for creating and modifying the way in which books are laid out, as well as for producing individual books. The Formation user interface can also display a representation of the pages of the book as it is being laid out, providing the operator with immediate feedback, as well as making demonstration of the system's capabilities both easy and attractive.

2.2 Approach

Although Formation was specifically required to produce classified telephone directories, it was always intended to be a much more general system. Instead of bringing specialist knowledge to bear directly on an efficient specialised representation of a classified directory, AIAI has designed a general purpose framework which allows knowledge about layout requirements to be expressed in a natural and modular manner.

Formation was designed particularly to support the expression of a required way of laying out the pages of a document. This means that the emphasis is on carrying out correctly what has been specified, rather than on achieving some notional optimal solution to the general 2-D packing problem. This distinguishes Formation from other current approaches to layout which are based on constraint satisfaction technologies, such as the DFKI LayLab system [3].

As a high speed of layout was a major system requirement, support for human operator interaction was designed in as an integral part of the production process. This has resulted in a relatively interactive manner of use for what is still, basically, a batch system. Manual intervention is needed for many real-life reasons. As well as the many practical difficulties there are in fully and unambiguously describing how every interaction that can result from every possible occurrence of every possible combination of input data should be handled, advertisers can require changes to be made to a particular page even after the book has been output to film for printing.

To support the operator, individual styles can collect statistics on the current run—the number of entries placed so far, the percentage of space within the book which has not been filled by entries, and so on—as it lays out a book. This information is collected on a page by page basis, and any significant deviation by one of these "quality measures" from a pre-set target can trigger a message to the operator who can then examine the particular page in question, either there and then or once the run is complete.

At Pindar, these completed pages may be edited using a page editor which has been developed specially in-house by Pindar's own technical staff. Yellow Magic's customers can modify individual pages through the *Sleight-of-Hand* tool which is available as part of the current Yellow Magic interactive layout software package.

In order to understand the kinds of knowledge that are involved in the layout task, AIAI carried out structured interviews with staff at Pindar who had experience in laying out Yellow Page directories by hand. Their experience reflected both work in the days before layout software was used extensively, and the continuing need to edit pages manually which have already been laid out automatically. This approach resulted in an ontology of concepts both specific to classified directory layout and of a more general nature. Following the knowledge structuring process, AIAI developed LSSL, the Layout Style Specification Language, which allows these ideas to be expressed in a natural manner.

LSSL is an object-oriented language which supports the direct expression of concepts relating to layout, such as "above", "follows", "paste", "justify". Using LSSL, layout *styles* can be implemented, each of which describes the way in which a particular document, or family of documents, should be laid out.

Formation is made up of three major components, as shown in figure 2:

- the operator interface to the system;
- the general layout engine which is based on the LSSL interpreter;
- style specification libraries which contain detailed methods, written in LSSL, for dealing with particular aspects of document layout.

Each individual style must address the following issues:

- the definition of the geometry and properties of the published document: this is based on the *spread*, which is conventionally (but not necessarily) two facing pages, as shown in figure 3;

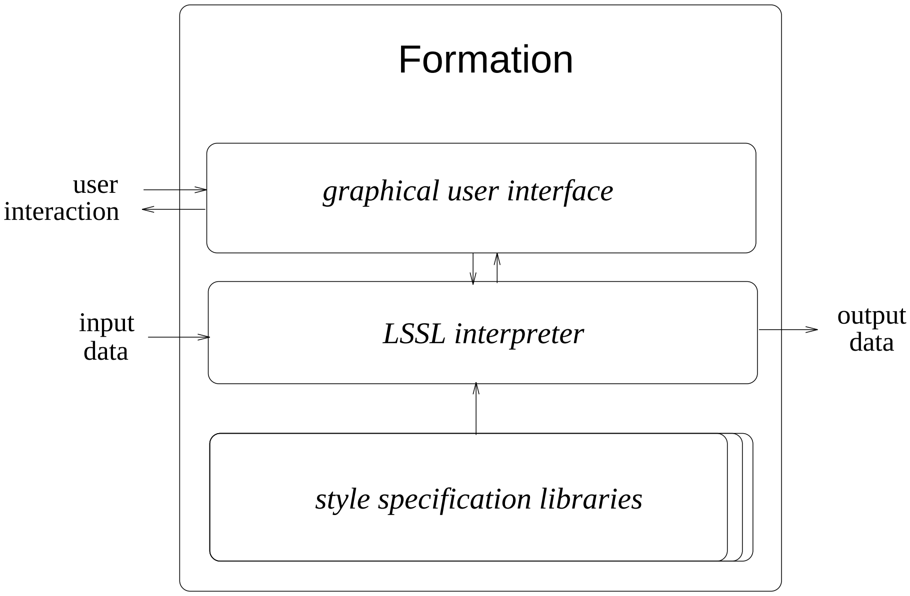

Figure 2: The architecture of Formation

- the definition of the types of entry which can appear in the document, such as: half page advertisement, classification header;
- the definition of the overall layout *strategy* and the individual *methods* which will be applied to produce the desired book from its entry data, such as: all line entries must be kept in sequence.

Pindar produces Yellow Page directories for BT by using a specific style which provides the precise look and feel that BT requires for its books.

In the remainder of this section, we provide overviews of LSSL and layout styles, and end by describing the implementation of the system.

2.3 LSSL

LSSL is a special purpose language for describing how data should be laid out within a document. In contrast to PostScript which supports describing the shapes and colours on a page, drawing lines, creating fonts, and so on, LSSL supports documents that have a well-defined syntax. In this, LSSL can be viewed as implementing a particular family of documents in the same way as does a particular SGML Document Type Description [4, 2]. However, as well as specifying a grammar for the content of the document, LSSL supports the expression of detailed knowledge about how the content should be laid out on the pages of the document, for example, specifying what widow and orphan control is required. LSSL adds semantics to the input syntax definition, thus also addressing the concerns of DSSSL [5].

LSSL is an object-oriented language which supports an extensible hierarchy of object types. As well as general-purpose facilities such as symbols, numbers, lists and functions, LSSL allows the programmer to express and manipulate concepts that are specific to document layout. LSSL provides facilities which address the following key concepts:

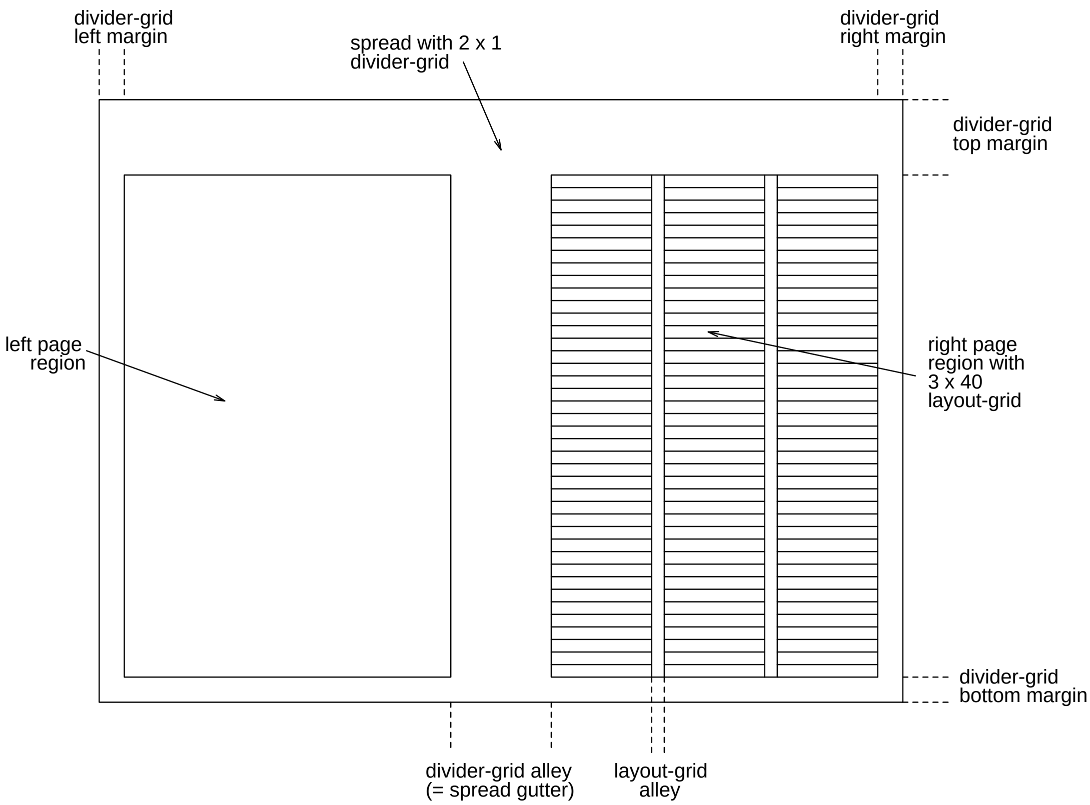

Figure 3: Spread structure with regions and grids

block A *block* is a rectangle; blocks are sub-typed into *items* and *regions*. Items represent the data to be laid out in a document, and regions represent rectangles within the document which can be filled by the layout process.

entry An *entry* is an input data item which should be positioned by the layout process.

grid A *grid* is used for logical two-dimensional division of a region: the definition of a grid includes its horizontal and vertical scale, any padding between the rows or columns of the grid, and any margins at the top, bottom, left or right of the grid. Grids are sub-typed into *divider-grids*, which divide regions into smaller regions, and *layout-grids*, upon which items can be placed. See figure 3 for an example.

spread A *spread* region, usually but not necessarily two facing pages, is the unit in which Formation lays out a document.

strategy A *strategy* encodes the high-level procedural knowledge which controls document layout; any given style will select a single layout strategy. Strategies can be parameterised, allowing them to be configured and providing control over their operation.

method *Methods* encode the detailed knowledge which specifies the appearance of a finished document; in general, each different method controls a different aspect of the document layout. For example, there might be a method to control the placement of display advertisements, and another method to control the splitting of listings across columns or spreads. Like strategies, methods can be parameterised. Defining a particular style will usually involve establishing a 'look and feel' for the document by configuring the parameters of the selected strategy and methods appropriately.

rule Each layout *rule* expresses a piece of knowledge, and each method will usually consist of several different layout rules. See 2.3.1, below, for an example.

A LSSL rule is an instance of a subtype of *function*. It has extra information which specifies when it should be called and with what arguments. These are not production rules in the traditional sense, although they resemble them in that they are invoked by the occurrence of particular *events*, rather than by being called directly from user code. When each rule is defined, it is associated with a particular object type or instance (which must be an `item` or a `grid`), and with a particular event in the layout process. As each event happens (such as the alignment of an item on a page), all the rules associated with that event are fired. LSSL programming, at the level of controlling the placement of entries within a spread, is therefore essentially event-driven.

style A layout *style* is the collection of all the information that is needed to turn an ordered sequence of entries into a correctly and pleasantly laid out document.

Style objects specify:

- the type of grid which defines the page structure;
- the types of the items that can be placed on the page;
- the strategy for placing items on the page;
- the methods which define how the finished page should look.

LSSL provides an inheritance mechanism, so that an instance of a LSSL type can inherit attributes ('slots') from its parent. LSSL syntax is similar to that of Lisp S-expressions but with some important differences, such as the addition of slot identifiers (for example `grid.hscale`, the `hscale` slot of a `grid` object), and the tokenisation of numbers (1.5 is read as the ratio 3/2 as LSSL does not provide floating-point numbers). LSSL looks superficially like Lisp: however, its execution model is, in fact, completely different.

2.3.1 An example LSSL method

```
(method eis "Entries in sequence"
   ("Sequenced entry type"  entry  nil))  ; slot holds entry type

;;; Record last sequenced entry pasted on this page
(rule eis.record-last (for eis.entry) (at paste) :(it gd _ _) ->
   (set gd.last-entry it))

;;; If last sequenced entry on this page gets unpasted, forget it
(rule eis.forget-last (for eis.entry) (at unpaste) :(it gd _ _) ->
   (if (eq it gd.last-entry)
       (cancel gd.last-entry)))
```

```
;;; Align a sequenced entry to follow the last one on this page
(rule eis.check (for eis.entry) (at align) :(it gd _ _) ->
   (or (not gd.last-entry)
       (> it.x gd.last-entry.x)
       (and (= it.x gd.last-entry.x)
            (> it.y gd.last-entry.y))))
```

The following general points are worth noting:

- A method can be configured by specifying values for its slots. In this case, the method eis has a slot eis.entry which determines what type of entry the method is to keep in sequence. A style which includes this method must configure it by filling in the slot, for example, by calling `(set eis.entry listing)`.
- The method eis has three rules associated with it. These are stored in slots of the method whose behaviour they define, in this case, the slots eis.record, eis.forget, and eis.check.
- We can associate a descriptive string with a method. This is a general feature of LSSL objects.
- The for slot specifies to what type of item the rule is applied.
- The at slot specifies the event at which the rule is called. For example, calling the built-in `paste` function will invoke all appropriate "paste" rules after pasting an item, the `align` function invokes all "align" rules, and so on.
- Calling a rule will produce a result, normally a boolean, which can be used by the caller. For example, if an "align" rule returns `nil` for a particular item, `align` will reject its current position.

2.4 Layout styles

A layout style controls the layout of a given document (or family of documents). A style will usually contain the following elements:

- the measurements and properties of the pages of the document, defined in terms of regions and grids;
- definitions of the appropriate item types and of their associated measurements and properties;
- definitions of the layout strategy to be used, and of the particular methods which control the appearance of the finished document.

Strategies and methods are held in a *style library*, and new strategies and methods are added to the library as new styles are developed. By supporting code re-use in this way, many layout requirements can be met by configuring a style from existing library elements rather than being obliged to start from scratch. From the point of view of the system user or document publisher, therefore, the process of defining a style can turn into selecting and configuring a strategy and appropriate methods from a library of existing, preprogrammed components.

For example, different strategies have been required for the UK and the USA markets. For BT's Yellow Pages, all entries are read by classification in sequence as a single stream and placed in the order in which they are encountered. For Yellow Magic, however, it is more natural to establish two separate input streams—one for listings, the other for display advertisements–and place the display advertisements on the page first. These strategies have then been fleshed out by various methods which deal with specific layout topics, for example: placing listings, placing display advertisements, and dealing with breaks between columns and pages.

If another publisher wants to establish a style that looks different, he might be quite content to use existing components from the library, but to select a different combination of methods and configure these in a distinctive manner. For instance, different combinations of "Class continues in next column/page" and "Class continued from previous column/page" labels can alter the finished appearance considerably. Individual control over these labels is available by specifying particular values to the parameters of the general column break method.

2.5 System implementation

The heart of Formation is the LSSL language. As well as functions for the direct expression of concepts relating to layout (such as `get-space`, `paste`, `unpaste`, `align`), LSSL provides a wide selection of standard procedures for carrying out basic arithmetic, performing file-based input and output, and so on. The LSSL interpreter is implemented in Allegro Common Lisp, and runs on PC, Sun, and Apple Macintosh workstations. Common Lisp was chosen as the implementation language because of its suitability for writing interpreters, and Allegro Common Lisp from Franz Inc was used because of the quality of its development environment.

The PC was selected as the principal platform for the full Formation system for business reasons. However, the main development of the LSSL interpreter was carried out on Sun workstations.

The user of Formation can interact directly with LSSL through a special development tool, or indirectly through Formation's menus and dialogue boxes. The user interface to Formation is implemented using the Allegro Common Lisp GUI Builder. The Page Editor, however, is implemented in C++ using Microsoft Foundation Classes. This gives portability across the PC, Apple and Sun platforms, as well as allowing easy access to PC dependent facilities such as OLE.

3 Application building

Formation was developed at AIAI by a project team of five. The skills of the individuals in the team complemented each other, and covered requirements capture, knowledge elicitation and modelling, AI software design and implementation, and project management. The initial development of the LSSL interpreter and the BT Yellow Pages style took approximately two hundred man days, and was carried out over a period of 10 months, during which regular review meetings were held with Pindar staff. The system was delivered on time and to specification.

Throughout the development of Formation, a high priority has been to ensure that Pindar is able to maintain and modify the system in the future. As a technology transfer organisation, AIAI aimed not to deliver a 'black box' product, but to provide its client with a long-term solution to its business problem. From Pindar's point of view, in order to minimise any risks taken, it was an important requirement that it should be possible to maintain and upgrade the system in-house.

In order to meet this requirement, specific technology transfer work packages were built into the initial project and into the follow-on work for the Yellow Magic style. Pindar's technical staff visited AIAI regularly to work with the project team, and they were involved in discussion and comment at all stages of design and implementation. Following project delivery, members of Pindar's support staff spent further time at AIAI learning about the system.

As well as Pindar technical staff's involvement throughout the project to ensure that the system met technical requirements, it was necessary to ensure that the system met operational requirements. To achieve this, Pindar's Production Operations Manager was also involved at key stages. Having overall responsibility for the typesetting and formatting process by which the BT Yellow Page directories are produced, his input has been essential, from specification of the requirements through to acceptance of the delivered system.

Since Formation is used within Pindar's BT Yellow Pages production process as a batch layout system, little direct training on Formation has been necessary for the system's operators. Pindar has therefore concentrated the training of the operations staff on the use of the cut down interface for production. Training of the technical and support staff has been achieved through the technology transfer work described above, and Pindar staff have successfully created new styles using Formation and demonstrated them to prospective clients.

A key stage in the development of Formation was the compilation of the ontology of layout terms which resulted from our knowledge acquisition exercises. At this stage, and later on when developing the BT Yellow Pages and the Yellow Magic styles, we relied heavily on our experience and expertise in knowledge engineering. By contrast, during the development of the LSSL interpreter itself, more traditional software engineering skills were required.

4 Application benefits

Formation is a powerful but easy to use knowledge-based layout system. It is easy enough for a beginner to learn quickly how to lay out books with a given page structure according to one of the layout styles provided by the system. It is powerful enough that developers can use it to design and lay out new books. They can specify new page structures and either edit existing layout styles or create new ones. This can all be done within the system with little or no programming. However, because Formation is built on top of LSSL, it can also be used to develop more advanced layout control and style features.

Formation offers two important features: a high throughput, and great flexibility in describing the details of the layout required. Any increase in the speed of laying out a book will immediately give a shorter lead time for production, and hence give sales staff a longer period for taking in advertisements from customers. A less obvious benefit is the ability for other staff to alter layout details and get almost immediate feedback on the impact of the changes on the look and feel of the entire book: production staff can investigate ways of further reducing the page count; marketing staff can examine ways of improving visual impact.

As the detailed knowledge which controls the 'look and feel' of the finished document is represented in a way that is readily understood by non-computing staff, it can also help to inform individual advertisers about the rules that are applied to determine the eventual position of their advertisement on the finished page and the consequences of interactions with other advertisers.

Many different features of the layout process and the finished directory can be measured and made available to the operator as part of a quality report produced at the end of the layout session. The quality report can, for example, contain information on:

- the number of pages laid out in the book,
- the number of fillers used in the book or on a page,
- the percentage of filler space in the book or on a page, by area,
- the time taken to lay out the book, or a selected range of pages,
- the distance between a display advertisement and its related line entry.

As Formation is a batch system, this quality report gives the operator confidence that the directory has been laid out correctly according to the client's specific style. For clients such as BT, who have print-runs of over half a million copies of a single directory, Formation ensures that the book is laid out efficiently, keeping down the page count with its associated printing costs, as well as reducing the weight of the book with its associated transport costs.

Speed of layout is an important aspect of the system due to both the number of directories that have to be processed and the relatively short timescales involved in their production and printing. Formation can lay out a typical directory of 1500 pages with 30,000 businesses arranged under 3,000 classifications and a further 5,000 display advertisements at a speed of well over 1,000 pages per hour on a 100 MHz Pentium based PC.

A key business requirement for Pindar was to have, and to be able to maintain, a generic, knowledge-based system to support laying out classified telephone directories. Formation provides this, allowing Pindar to lay out classified Yellow Pages directories in a flexible way in order to service new and existing customers. Pindar also has an in-house team who understand the design of the system and how it was implemented, and who are capable of updating it to meet new clients' requirements.

Although Formation was intended specifically to produce classified telephone directories, it is a much more general system and can be used for the two-dimensional layout of general shapes based on rectangles. Because of this general framework, Pindar can see enormous potential for marketing Formation in Europe, the USA and the Far East.

The knowledge-based system, Formation, has been in trial use with Pindar since January 1996. During this time it has already brought Pindar some of the expected benefits. Because the detailed knowledge about layout is defined at the style level, and because it is quick and easy to reconfigure a style using the Formation user interface, Pindar has been able to demonstrate the flexibility of the system to its clients. The approach taken, that is, the design and implementation of a suitable language in which to express knowledge about document layout, has been well validated.

Formation will be in full production use for BT's Yellow Page directories from September 1996.

References

[1] Hong-Gian Chew and Moung Liang. ALEXIS: An Intelligent Layout Tool for Publishing. In *Proceedings of the 6th Innovative Applications of Artificial Intelligence Conference*, Seattle, Washington, 1994. AAAI.

[2] C. F. Goldfarb. *The SGML Handbook*. Clarendon Press, Oxford, 1990.

[3] W. H. Graf. The Constraint-Based Layout Framework LayLab and its Applications. In *Proceedings of the Workshop on Effective Abstractions in Multimedia Layout, Presentations, and Interaction*, San Francisco, 1995. ACM.

[4] ISO 8879: 1986 Information processing — Text and office systems — Standard Generalized Markup Language (SGML), 1986.

[5] ISO DIS 10179, Information technology — Text and office systems — Document Style Semantics and Specification Language (DSSSL), 1991.