Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

LAYOUT BASED MODEL GENERATION

Peter Lorenz
Thomas Schulze

Department of Simulation and Graphics
Otto-von-Guericke University Magdeburg
D-39016 Magdeburg, PSF 4120, GERMANY

ABSTRACT

Layouts of systems (e.g. barbershops, job shop, road traffic and copper smelter) contain much information important for modelling.
Layout Based Model Generation (LBMG) means:
- extract all model relevant data from layouts,
- complement the initial layout by simulation specific components,
- extract additional model relevant data from other data sources,
- generate a simulation model from these extraction's,
- ask the user for complementary input data,
- run the simulation model, and
- present the simulation results and animation on the layout, generated in the second step.

This paper describes an efficient approach to simulation modelling. This approach intends to avoid any doubled or repeated data acquisition and any avoidable mouse click or drop and drug. The modeler can make precise, realistic models with little cost of time or money. The models that are created are now free of bugs that are normally introduced by the manual model construction and data acquisition process. The *preconditions* for LBMG are the
- existence of a defined set of object classes in the domain of application and
- existence of layouts or other pictures which contain model relevant information and which are of interest for the presentation of results by animation.

The first ideas for LBMG are described by Lorenz, Schriber and Schulze (1994) and by Lorenz and Schulze (1995). The LBMG approach is described and verified by the implementation of a *prototype*. The prototype uses AutoCAD™ ,DXF and Proof™ Layout file types as the layout basis for model generation. The approach can be used for any simulator with a language oriented model description interface, e.g. GPSS/H, MODSIM, SIMAN and other simulation systems.

1 DOMAINS, INSTANCES AND TOOLS

First we will explain some frequently used terms.

Domains are special fields of simulation applications. Each domain has its own set of object classes. Object classes contain attributes, rules or methods for changing attributes and rules for interaction with other objects of the same or other classes. A domain may have its own rule definition language and interface to the simulator. The rules control the dynamic change of object attribute values over time. Examples for domains are barbershops, job shops, road traffic intersections, copper smelting plants.

The presented LBMG approach has both general and domain specific components. Examples for *general components* are:
- a LANALYS (Layout Analysis) component, which looks for all classes, objects, messages and paths in a Proof layout file and writes the names and the positions of all objects in a *structure file* and
- a NAMINP (Name Import) component, which looks for NAME attributes of blocks in a DXF file, extracts all values of these attributes in block entities and introduces these names into a Proof layout file.

Examples for *domain specific components* are:
- a GENMOD (Generate Model) component for barbershops, which reads data from the structure file, generates instance specific GPSS/H-statements and combines them with a sequence of domain specific, instance independent statements (this component is both domain and tool specific),
- a FINDCRSS (Find Crossing Point) component for traffic systems, which looks for crossing lanes and write the lane names and the crossing point co-ordinates in the structure file.

Instances are individual systems of a domain. Different

instances of a domain have different subsets of objects. All objects are representatives of the domain specific classes. Examples are:

- a barbershop with *n* CHAIRS, *m* BARBERS, an ENTRY and an EXIT door, a FIFO waiting order and defined places for all objects,
- a job shop with *n* different working STATIONS, each equipped with a limited waiting magazine for jobs, a SOURCE and a SINK point and defined places for each of these objects, and
- a traffic intersection with road traffic lanes, lanes for bikes and streetcars, traffic lights and signs, sources and sinks for different kinds of moving objects and a layout with detailed information about the geometry and topology of the whole system.

The prototype *avoids* instance oriented components. A model of an instance shall be generated or created completely by general, domain specific and tool specific components.

Tools are simulation systems like GPSS/H, MODSIM, SIMAN and others with a language oriented model description interface. The prototype is multi-tool oriented. Different tools are used and compared with respect to their usability for LBMG. The prototype contains tool oriented components, for example a GPSCONF-component. It checks the GPSS/H-conformity of all class, object, message and path names in the layout and in the structure file.

2 INTRODUCTORY EXAMPLE: JOE'S BARBERSHOP

Joe's barbershop is the traditional introductory example for queuing and discrete event simulation models (O'Donovan 1979). It is used here to explain the computer supported steps from an initial layout to the simulation model and to the animation on this layout.

Our barbershop is somewhat different from its classical ancestor. It involves not only BARBERs and CHAIRs as abstract objects: Our objects have names and positions. The barbershop is an area with defined ENTRY and EXIT doors limiting walls and windows. The CUSTOMERs arrive at the ENTRY door and accept the following *behaviour rules*: Arriving CUSTOMERs

(1) go to a free BARBER or if there is no BARBER free,
(2) go to a free CHAIR to wait for a free BARBER or if there is no CHAIR free,
(3) go from the ENTRY to the EXIT door.

Served CUSTOMERs go from the BARBER to the EXIT door. All customers move with a constant speed between locations. Input data are the inter arrival time of CUSTOMERs and the serving time of BARBERs. They are asked for at the beginning of each run.

The modelling process should start with a *master layout* for barbershops. This can be an AutoCAD layout with predefined blocks for BARBERs, CHAIRs and DOORs. The blocks should have a NAME attribute. *Modelling an instance of a barbershop* means

- changing the positions of any passive primitive (wall and window),
- changing the positions of existing block entities (DOOR, BARBER, CHAIR), or rotating them, and
- inserting new items of blocks.

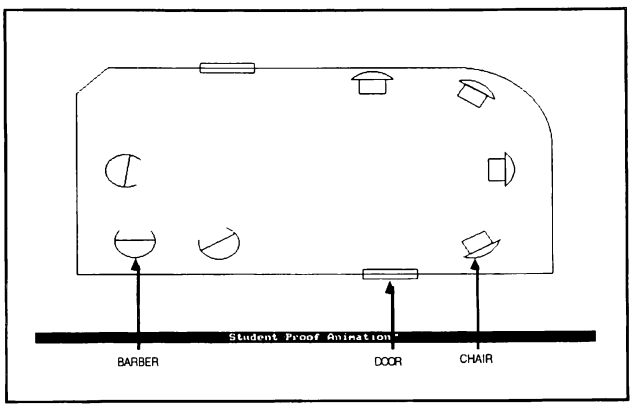

Figure 1: Native Layout of a Barbershop

The modified master layout maybe called the *CAD Layout* of the modelled barbershop. It is used as the input file for a filtering program. The filtering program

- eliminates unneeded layers of the original layout file (e.g. electrical circuits if a such layer exists) and
- transforms the AutoCAD DXF File into a Proof Layout file.

The result of this process is *a native or initial animation layout.* Figure 1 shows an example of such a native animation layout: It only contains walls, DOORs, BARBERs and CHAIRs.

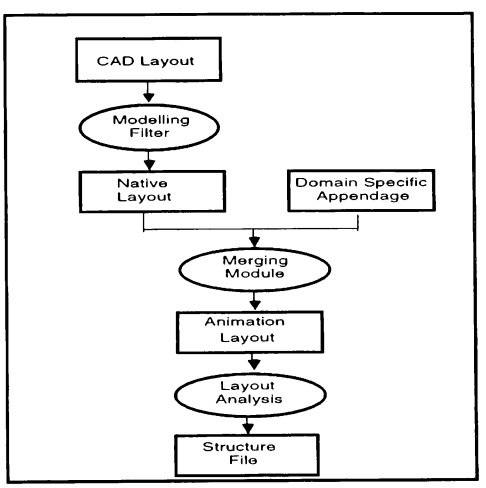

Figure 2: From the CAD Layout to the Structure File

This is not enough information for an acceptable animation. The native layout has to be appended by simulation and animation specific items: a clock, some counters as messages varying during the animation, and messages identifying the instance and showing the input data. This appendage is domain specific. Figure 2 shows the steps between the CAD layout and the animation layout.

The result of filtering the CAD layout and merging the result with a domain specific appendage is the *animation layout* shown in Figure 3.

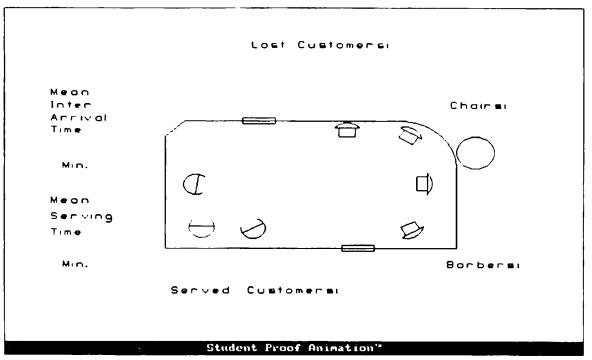

Figure 3: An Instance of Barbershop Animation Layout

Of course it is possible to start with a Proof based master layout already containing all animation relevant items.

The next step is a general, not domain specific *layout analysis*. Layout analysis uses the animation layout file as input. It detects all classes, all objects with names and positions, all messages, bars, plots and paths, and writes these items in the *structure file*. Table 1 shows an excerpt from the structure file of the above mentioned instance of the barbershop.

Table 1 : Excerpt from the Structure File of a Barbershop Instance

```
$OBJECTS
$CHAIR 4
CHAIR1 11.3763 9.3839 Color F7
CHAIR2 14.6864 3.1512 Color F7 Rotation 210
$DOOR 2
ENTRY 11.8956 2.001 Color F7
EXIT 6.7979 10 Color F7
$BARBER 3
JOE1 3.4321 6.0064 Color F7 Rotation 260
JOE2 6.5679 3.1164 Color F7 Rotation 30
$CUSTOM 0
$CLOCK 1
JCLOCK 18 8 Color F4 Rotation 150
$SMHAND 0
```

The structure file can be scanned manually or by programs. If discrepancies or inconsistencies are detected, the layout has to be edited. The proper structure file is the input for domain and tool specific model generators or domain specific models.

Figure 4 shows the approach used for generating the *simulation model* based on different tools and using the structure file from the introductory barbershop example.

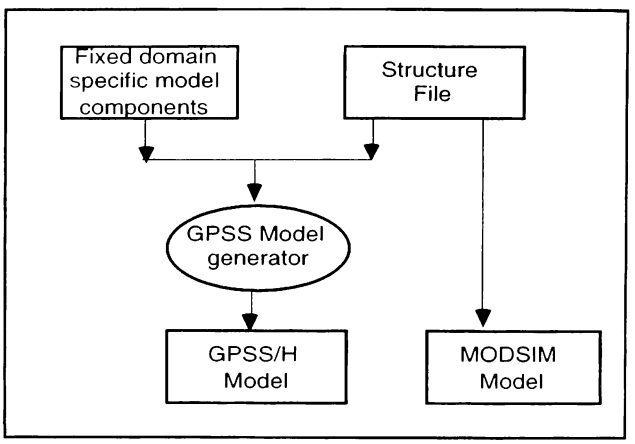

Figure 4: From the Object File to Tool Specific Models

The final animation can be created by different tools and runs with the same animation layout. Figure 5 shows a snapshot of the animation of a barbershop instance.

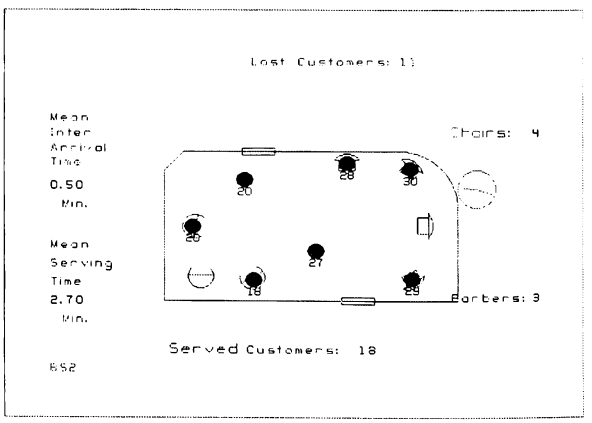

Figure 5: A Snapshot of an Animation Layout of a Barbershop Instance

The spectrum of model variation in the barbershop domain can be estimated by comparing Figure 5 and Figure 6. The new instance of a barbershop model shown in Figure 6 is derived from the preceding *by layout editing* only.

In the LBMG prototype (see Section 6) the user selects the step **Create or Modify a Layout/ Modify a**

Proof Layout, chooses the Draw Mode, changes the positions of objects, creates new objects and starts the steps **Analyse a Layout, Run the MODSIM Based Model** or **Generate and Run the GPSS Based Model** (Figure 12). Without any manual work and only by calling these steps, the new model is automatically generated and executed. After a few moments the animation for the new instance can be regarded.

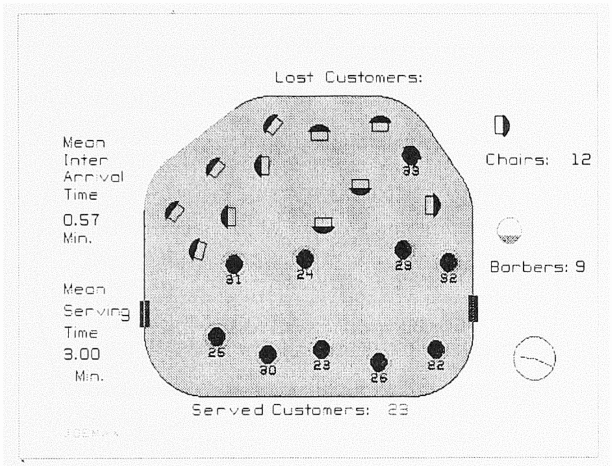

Figure 6: Another Instance of a Barbershop Model

The basic idea of LBMG presented above using the barbershop domain has to be modified and extended, if it is to apply to other domains. The following sections are related to general questions and problems arising from the transfer to other domains.

3 CAD AND ANIMATION LAYOUT

CAD layouts of manufacturing, traffic and other systems embody information not useful for simulation models, and they do not include any information needed for simulation and animation.

CAD layouts are suitable for model generation only if their construction follows domain specific *conventions*. The conventions can be an agreement between the CAD user and the simulation model constructor. They can also exist before anyone has thought about LBMG.

In some German towns, a convention ("Hamburg-Standard") on urban planning with AutoCAD is generally accepted. This convention defines the content of layers in a layout file and the symbols (AutoCAD blocks) which are to be used for real objects. Figures 7 and 8 show examples of layers in a traffic intersection layout, made using the "Hamburg Standard". The layout is imported from a AutoCAD DXF file into a Proof animation layout file. The filtering is easy to do with the layer structure of this standard.

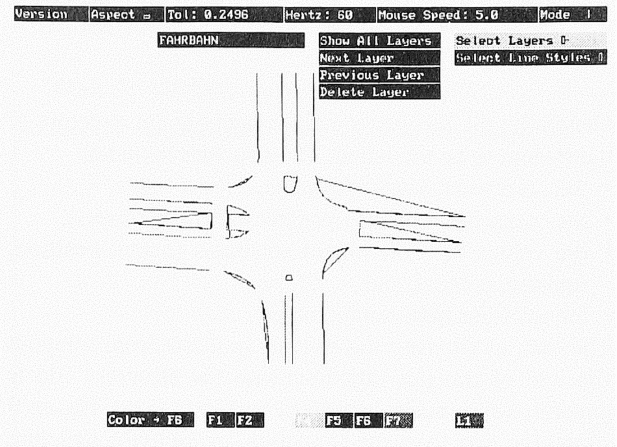

Figure 7: Layer "FAHRBAHN" of a Traffic Intersection Layout

Which simulation specific appendages to the animation layout are necessary in the domain of traffic intersection models?

(1) A clock, counters as messages varying during the animation, and messages identifying the instance and showing the input data analogue to the introductory example,
(2) bars and plots as graphical presentation tools for system variables to be presented during the animation,
(3) paths for the movement of driving objects,
(4) markers for traffic sources and sinks and regions for a comprehensive statistical observation,
(5) merging areas for characterising of two or more lanes, and
(6) messages for behaviour presentation of selected drivers.

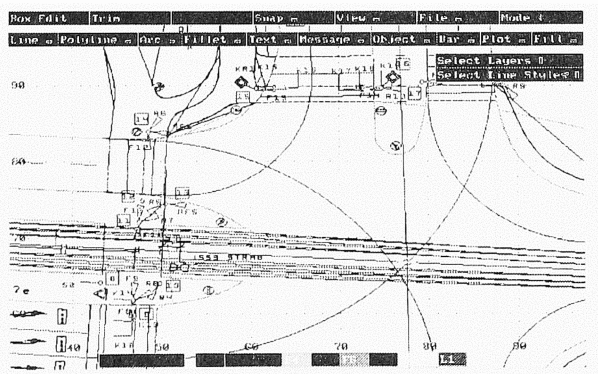

Figure 8: A Zoomed Layout with Some Layers

The appendages are stored in a library. They can be chosen manually inside the PROOF system or a program selects automatically the needed items and the user has to place the items on right places.

4 LAYOUT ANALYSIS AND STRUCTURE FILES

Generally, the layouts contain data about the structure of the system to be modelled. The structure includes:

- the involved classes of objects,
- the comprised objects (instances of these classes),
- the values of some object attributes (name, position, rotation, colour), and
- the relations between the objects.

The task of filtering the first three items from a layout file and of transferring them into the structure file is general, not domain dependent and easy to do. Figure 9 shows the way from the animation layout to the structure file. Table 1 is an example of a result for this general, not domain specific solution.

The fourth task is domain dependent and can require much effort. This can be exemplified by using traffic systems: In this case for each path or lane, all crossing paths, all joining and all branching paths, all detectors, traffic lights and signs bounded to it, must be detected.

A domain specific tool for traffic systems was developed to support the detection of relations between path objects. This tool analyses all paths in a PROOF layout and generates a description of the path network which is imbedded in the Structure File. The use of this description reduces the human amount of network implementation up to 50 %.

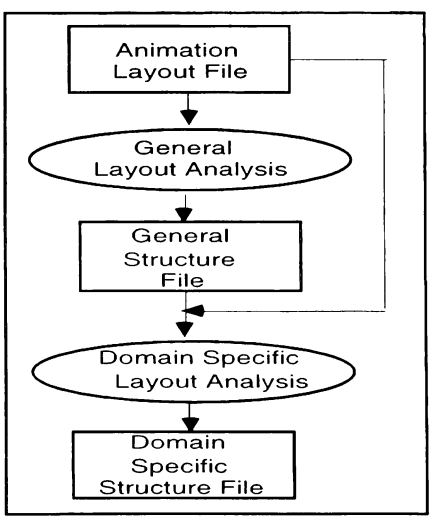

Figure 9: General and Domain Dependent Layout Analysis

5 MODEL GENERATION

Models of the real world usually contain a great amount of *non layout based (NLB) information.* Some examples are timetables for the work in a factory, and time varying traffic stream intensity data. Figure 10 shows the three components as needed sources for model generation. They have to be merged to get the final simulation model.

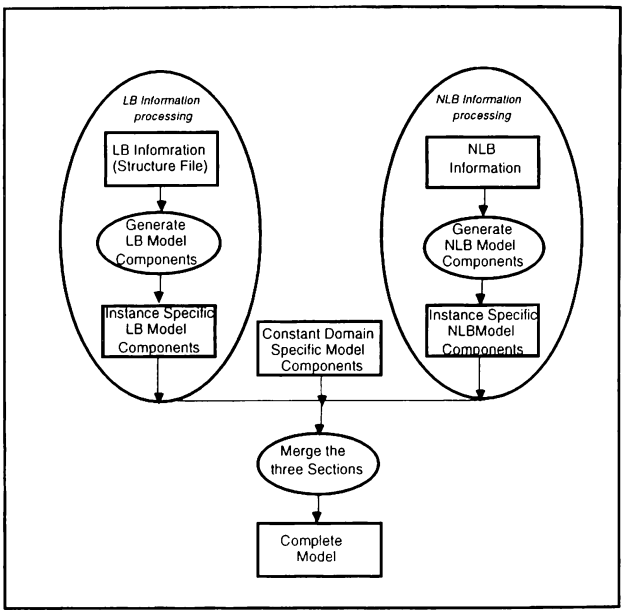

Figure 10: LB and NLB Trees in Model Generation

Walper (1994) contains a domain specific example for NLB Information processing. For the domain "Road Traffic Intersections", a special language for the description of load dependent traffic control systems has been proposed. Control algorithms coded by this language can be translated into GPSSH code and can be used for traffic light control program generation.

The Model Generation, then, is a *domain and tool dependent* step with a great variety of possible solutions. In the case of our introductory example, we can compare two possible solutions:

- generating a GPSS/H Model, and
- using a general MODSIM II Model.

5.1 Generating a GPSS/H Model

In the implemented prototype, the generation of GPSS code is made by a GPSS/H program. GPSS/H contains efficient features for file and string processing for executing the simple task of translating data from the structure file into GPSS/H statements. Table 2 shows

some lines of the generated code.

Table 2: Fragments of the Generated Code

```
      VCHAR*8  &INSTANCE
      LET      &INSTANCE='XXX'
*   CHAIRS
      INTEGER  &NCHAIR      ANZAHL
      LET      &NCHAIR=4
  CHAIRS FUNCTION  PH(CHAIR),S4,F
  1,CHAIR1/2,CHAIR3/3,CHAIR4/4,STNEU/
  XCHAIR FUNCTION  PH(CHAIR),D4
  1,14.5648/2,15.3484/3,14.6864/4,11.3763/
  YCHAIR FUNCTION  PH(CHAIR),D4
  1,9.0275/2,6.076/3,3.1512/4,9.3839/
*   DOORS
      INTEGER  &NDOOR      ANZAHL
      LET      &NDOOR=2
  DOORS FUNCTION  PH(DOOR),S2,F
  1,ENTRY/2,EXIT/
  XDOOR FUNCTION  PH(DOOR),D2
  1,6.7979/2,11.8956/
  YDOOR FUNCTION  PH(DOOR),D2
  1,10/2,2.001/
```

The GPSS/H Model generation uses FUNCTIONS of the S-type for an automated mapping of object names to numbers. In this way, the generated models are flexible and can handle any number of included objects.

5.2 Using a domain specific MODSIM Model

MODSIM is a general-purpose, modular, block structured language which provides support for object-oriented programming and discrete event simulation. The example MODSIM model is instance independent. All needed object instances are allocated dynamically. The allocating process is controlled by the Structure File. The needed object declarations are divided into domain specific objects and domain independent objects.

The set of domain specific objects includes, for example, ChairObj, BarberObj and CustomerObj. Table 3 shows the declaration for the CustomerObj.

Domain independent objects are, for example, QueueObj, GeneratorObj and ControlSimObj. These object declarations are used in other domains. The MODSIM model is structured into 4 modules:

- allocating independent objects,
- reading experimental parameters,
- reading the Structure File and allocating domain specific objects, and
- simulation.

Table 3: MODSIM declaration for a CustomerObj

```
CustomerObj = OBJECT
 ident : STRING;
 speed : REAL ;
 ASK METHOD setIdent
      (IN number : INTEGER );
 ASK METHOD setSpeed
      (IN aSpeed : REAL );
 ASK METHOD getIdent () : STRING ;
 ASK METHOD getWalkTime
      (IN source : GeneralObj ;
       IN target: GeneralObj ):REAL;
 TELL METHOD EntryShop;
 TELL METHOD TakeBarber
      (IN source : GeneralObj ;
       IN barber : BarberObj );
 TELL METHOD TakeChair
      (IN source : GeneralObj ;
       IN chair : ChairObj );
 TELL METHOD LeaveShop
      (IN source : GeneralObj );
END OBJECT;
```

Table 4 shows the relationships between the objects of the Structure File and the declared MODSIM objects.

Table 4: Relationship Between Structure File Objects and Declared MODSIM Objects

Detected Structure File objects	Declared MODSIM objects
$CHAIR	ChairObj
$DOOR	EntryDoorObj
	ExitDoorObj
$BARBER	BarberObj
$CUSTOM	CustomerObj
$CLOCK	AnimatorObj
$SMHAND	AnimatorObj
$BGHAND	AnimatorObj
MESSAGES	AnimatorObj

6 FILE MANAGEMENT AND PRESENTATION

The prototype of the LBMG has a file structure and a presentation management which are constructed according to the problem structure described above. Figure 11 shows the file structure of the prototype.

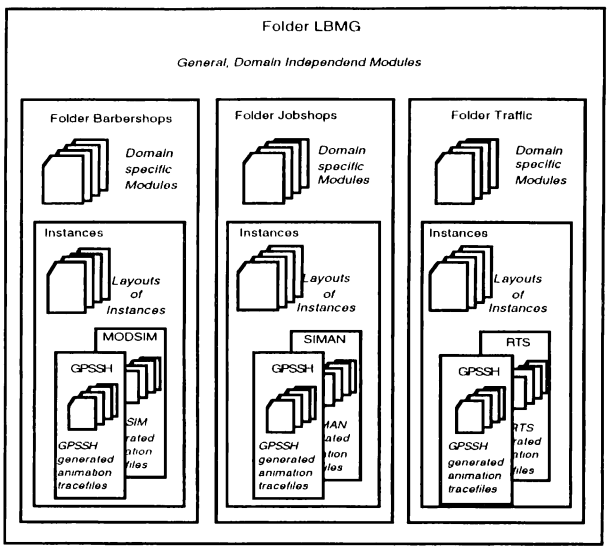

Figure 11: File and Problem Frame Structure for LBMG

The prototype contains a Proof presentation system with an equivalent structure. Figure 12 shows the initial presentation menu and the main menu for presenting Joe's barbershop. Here the basic functions of the system are visualised.

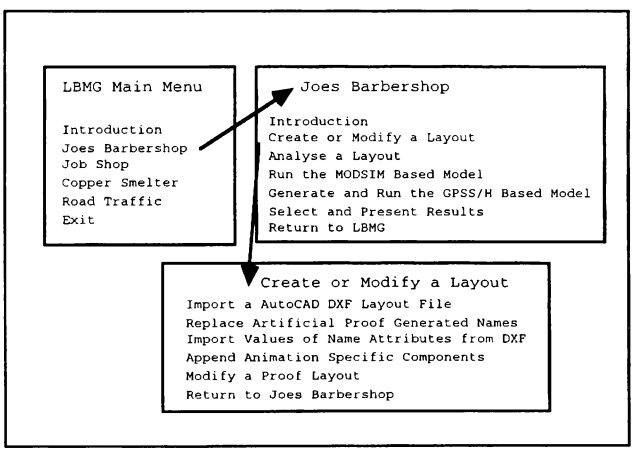

Figure 12: Presentation Menus

7 COMPETITIVE SOLUTIONS

During recent years, some competitive solutions to our approach have been published and used. The following section describes the main differences between these approaches.

The LBMG approach presented here is a special kind of model generation. Layouts are inserted into the bundle of basic information for model generation.

Language Based Simulation Model Generators

To generate instances of models in a defined domain is an old and well published approach. The classical approach uses special languages for model description and special translators to translate these description into a simulation language. Examples of this solution are given in Ludwig (1983) and Walper (1994). Kimbler and Watford (1988) contains an overview. From the viewpoint of LBMG, the language based model generation is one component of the LBMG approach.

GUI Based Modelling

GUI (Graphical User Interface) based modelling is a widely used approach for model generation. Models are constructed based on the usual GUI functionality. Icons are used as object symbols and connected by edges and arrows. Requesters are used for model description and data acquisition. This is a common solution for some simulators. SLAMSYSTEM, Extend, ARENA and WITNESS are some examples for such simulators. In Germany some systems, which are oriented on GUI based modelling like DOSIMIS, SIMPLE++, Create! (Rüger, Hoppe and Kirchner 1990) and some others, have been developed and used.

The main difference to the LBMG approach is the fact, that in the case of GUI based modelling, all *layout specific information* (classes, objects and their relations) have to be imported by man or hand into the model. There is no computer supported way for introducing layout modifications into the simulation model. The manual import or data transfer from the layout to the model leads to additional errors and expenses.

Integration of CAD Layout Backgrounds

Some papers describe an approach using CAD layouts as background information for model construction and for animation (*CLB approach*). In Thim and Abels (1991), Tumray (1992), Klußmann and Krauth (1995), and Mertins, Rabe and Könner (1995) layouts are used as support for model construction. One proposal consists of the use of a special AutoCAD layer for simulation model components and the transfer of these components into a model description. The main differences between CLB and LBMG are:

- CLB uses the normal CAD layout as a basis for the introduction of new, simulation specific layers which are separated from the original layers; LBMG uses the normal layout as a source for data acquisition, and
- CLB has no connection or "background-connection" only between the CAD layout and the animation layout for presenting results; LBMG uses this appended layout with some animation specific elements for animation.

8 SUMMARY

LBMG is a new approach for simulation model construction. It is characterised:

- by an access to all information in a CAD layout of the system under study,
- by a high degree of realism in modelling without expensive data acquisition,
- by a continuous path from the CAD Layout to the presentation of simulation results,
- as an instrument for warranting the consistency of the layout and the model during all layout variations and modifications.

REFERENCES

Kimbler, D.L., and B. A. Watford. 1988. Simulation Program Generators: A Functional Perspective. In *Proceedings of SCS Artificial Intelligence Conference,* 234 -240. SCS , San Diego

Klußmann, J., J. Krauth, and R. Meyer-Splanemann. 1995. Bildgenerierte Simulationsmodelle auf der Grundlage von CAD-Zeichnungen. In *Integration von Bild, Modell und Text '95*, eds. E. Blümel, P. Lorenz, T. Strothotte, D. Ziems, 15-26. ASIM-Mitteilungen, Wien.

Lorenz, P., T. J. Schriber, and T. Schulze. 1994. The Design, Implementation, Application and Comparison of two highly Automated Traffic Simulators. In *Proceedings of 1994 Winter Simulation Conference*, eds. J. D. Tew, S. Manivannan, D. A. Sadowski, A. F. Seila, 1084-1092. Society for Computer Simulation, La Jolla, California.

Lorenz, P., and T. Schulze. 1995. Bildgenerierte Simulationsmodelle. In *Integration von Bild, Modell und Text '95* , eds. E. Blümel, P. Lorenz, T. Strothotte and D. Ziems, 27-42. ASIM-Mitteilungen, Wien.

Ludwig, W. 1983. Simulationsmodell und Fachsprache für Kreisförderer - ein Beitrag zur modellmäßigen Funktionserprobung geführter Transportsysteme. Dissertation, Hochschule für Verkehrswesen Dresden, Germany.

Mertins, K., M. Rabe, and S. Könner. 1995. Integration von Fabriksimulation und CAD. In *Integration von Bild, Modell und Text '95*, eds. E. Blümel, P. Lorenz, T. Strothotte, D. Ziems, 163-174. ASIM-Mitteilungen, Wien.

O'Donovan, T. M. 1979. *GPSS Simulation Made Simple*. ed. D. W. Barron. Wiley Seriers in Computing. John Wiley & Sons, Chichester, 1979.

Rüger, M., U. Hoppe, and H. Kirchner. 1990. Objektorientierte Modellierung von Bausteinen innerhalb der Simulatorentwicklungsumgebung Create!. In *6. Symposium Simulationstechnik*, ed. F. Breitenecker, 140-144. Vieweg Verlag.

Thim, C., and S. Abels. 1991. CAD-unterstützte Generierung von Simulationsmodellen. In *7. Symposium Simulationstechnik*, ed. D. Tavangarians, 584-588. Vieweg Verlag.

Tumary, K. 1992. Integrating Simulations with CAD Tools for Effective Facility Layout Evaluation, In *Proceedings of 1992 Winter Simulation Conference*, ed. J. S. Swain, D. Goldsman, R. C. Crain, J. R. Wilson, 995-999. Society for Computer Simulation, La Jolla, California.

Walper, R. 1994. Generierung von Modellen der Lichtsignalsteuerung für den Straßenverkehr. Diplomthesis. Universität Magdeburg, Germany.

AUTHOR BIOGRAPHIES

PETER LORENZ is a Professor at the University of Magdeburg, Department for Simulation and Graphics. He teaches discrete simulation, computer animation and graphics. His research interests include advanced applications of simulation and animation in manufacturing, logistics and traffic control.

THOMAS SCHULZE is an Associate Professor at the University of Magdeburg in the Department for Simulation and Graphics. His research interests include modeling methodology, public systems modeling and simulation output analysis. He is an active member of ASIM, the German simulation society.