GRIMBI - A COMBINATION OF INTERACTIVE GRAPHICS METHODS AND CAD DATABASE TECHNIQUES FOR FUNCTIONAL MODELLING

K. Leinemann
Institut für Reaktorentwicklung
Kernforschungszentrum Karlsruhe

ABSTRACT

The CAD-system GRIMBI serves for synthesizing information structures under functional aspects. It supports manipulation of informations usually represented as blockdiagrams and tables. This CAD-system, therefore, is an aid in the early design phases. It is not simply a drawing system, but rather combines interactive graphics methods with database technology for building up information structures, which represent functional aspects of an object or a system. GRIMBI includes a specialized data definition facility to define logical model classes and to describe the external graphic data representation. Modelling according to a model class is done by standard operations, completed by working methods like stepwise refinement/abstraction and management of design alternatives. The operations are supported by an autonomous command/menue language and by a DML, which is an extension of PL/1. GRIMBI provides a separation of tasks between a mainframe (for data base management and analysis) and a satellite (for the interactive communication and the graphics subtasks). GRIMBI is implemented as a subsystem of REGENT, a CAD kernel system.

KEYWORDS: CAD, graphics, database, DDL, DML, functional modelling, satellite, task distribution

1. INTRODUCTION

Design of technical products and systems is concerned with many different design methods, resulting in a variety of CAD-systems with their own databases, tailored to the particular needs. Using such specialized CAD-systems for single tasks or groups of tasks, an over all CAD-system may be built up by an architecture shown in fig.1. The data bases of the subsystems are temporary, physical subdatabases of a central database containing all the data needed for design and manufacturing. These subdatabases, sometimes called operational databases, are linked to the central base by data transformers. This kind of loose linkage allows to use a higher data abstraction level in the central database, than it is required for the specialized processes. It unburdens the management of the main database, which discriminates only parametrized groups of data (information hiding). Such an architecture contrasts to the solution using logical subdatabases.

Subject of this paper is a special CAD-system for functional modelling planned as a building block of an architecture shown in fig.1. The CAD-system GRIMBI serves for synthesizing and management of information structures under functional aspects. It supports manipulation of information usually represented as block diagrams and tables.

Basic objectives of GRIMBI follow from its kind of usage. GRIMBI is particularly designed for an environment, which is characterized by rapid development and model changes. Hence, adaption of the modelling system to various model types should be easy, especially as it is done by the engineer himself, not by a system administrator. Particular emphasis was laid upon early detection of modelling errors. This is to prevent waste of resources and loss of time, due to working with an inconsistent model. Losses may be significant when an inconsistent model is used as input to expensive analysis computations. In many design areas several different functional models are used simultaniously: e.g., in plant design P&I-diagrams are used as information basis to derive fault tree models, functional models for reliability analysis /1/.

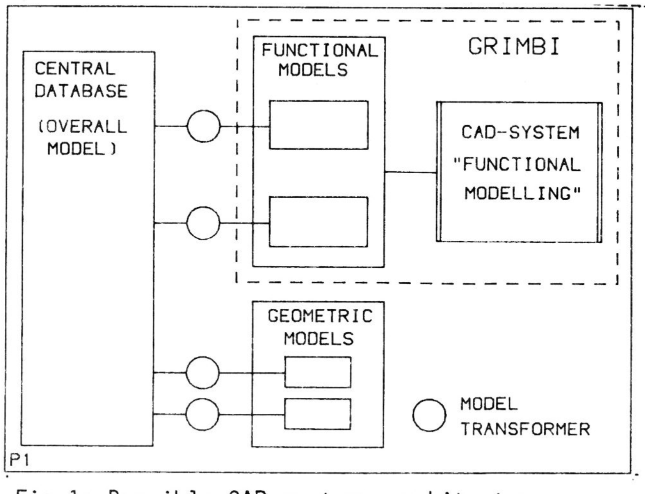
CENTRAL DATABASE (OVERALL MODEL)
FUNCTIONAL MODELS
GRIMBI
CAD-SYSTEM "FUNCTIONAL MODELLING"
GEOMETRIC MODELS
MODEL TRANSFORMER
P1

Fig.1: Possible CAD-systems architecture

The analysis algorithms typically are running on a large main frame in batch (e.g. fault tree analysis programs for reliability analysis). Integration of interactive synthesis and batch analysis therefore is required too.

GRIMBI was designed and implemented according to the following design objectives. GRIMBI should support

- construction of graph models, usual in engineering,
- early detection of problem dependent modelling errors, to provide consistent models,
- descriptive model class specification to facilitate adaption to special problem classes and their changements,
- modelling operations, usual in systems design like stepwise refinement/ abstraction, management of design alternatives, handling various model types simultanously,
- integration of interactive work and batch computations.

2. GRIMBI SYSTEM OVERVIEW

Like database systems GRIMBI distinguishes between /2/

- a data definition phase to describe a certain model class, that means to describe problem oriented modelling restrictions and
- a data manipulation phase to generate model class instances and to analyse them.

Fig.2 represents a basic architecture of GRIMBI, showing its parts (data and processes) and its kind of usage. For data definition, an activity, which demands for a careful planning, and hence is best done in batch mode, a DDL (data definition language) is available as extension of PL/1. Interactive modelling, batch modelling, and batch access for analysis are supported by a data manipulation module (DBMS).

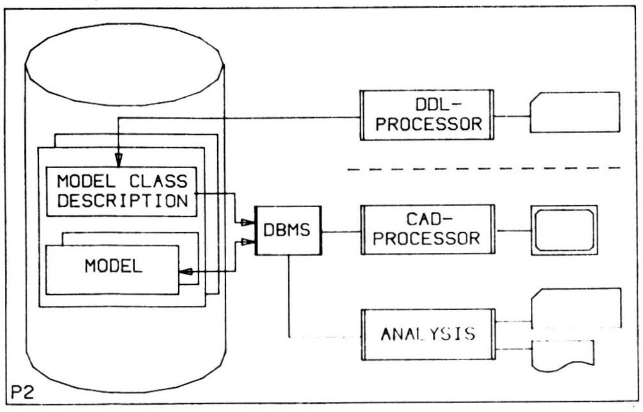

Fig.2: Basic architecture of GRIMBI

Its operations are bounded by the model class restrictions. They are available using a DML (data manipulation language), implemented as as PL/1-extension for batch processing, or using a self-contained command/menue language in interactive mode, allowing data identification by picking. The DML-commands for batch modelling do not include the possibilities of editing the graphical representation of the data like symbol positioning, or routing of connections. Such graphic information may be added later on in an interactive mode, before additional interactive modifications are performed with the model.

3. MODEL CLASS DESCRIPTION

Adaptation of GRIMBI to a certain problem area is done using the GRIMBI-DDL to describe a model class. Model class descriptions represent problem oriented modelling restrictions, they enable the system to recognize modelling errors and to guarantee model consistency to the model type used in analysis algorithms.

The GRIMBI-DDL is based on a problem oriented data model, suited for functional modelling. Fig.3 shows graphically the elements of the block diagram data model. Using such a specialized data model instead of a general purpose model (e.g. CODASYL-DBTG), greatly facilitates the schema definition. This is particularly relevant in a development environment, where new model types are created often and are subject to rapid improvement. The data definition task in this environment should be done by the engineer, not by a data base administrator, and therefore should be easy.

A model class is a collection of user (or problem) data types, characterized by data types of the DDL. The GRIMBI-DDL differentiates between three classes of data types.

(1) Types to describe the basic elements for model building: BASEOBJECT characterizes elements without an internal structure, but with ports. RELATION describes sets of object pairs, and COMPLEX specifies object types with a fixed internal structure, composed of BASEOBJECT and RELATION type objects.
(2) Types to describe model substructuring: SUBNET-DESCRIPTOR, SUBNETPORT, NET, SYSTEMPORT.
(3) Types to describe the graphical representation of the data: BASEGRAPHIC, ATTRIBUTEGRAPHIC, PORTGRAPHIC, PORTCOORDINATES, TEXTWINDOW, PICKAREA.

Basic user elements for modelling are of the type BASEOBJECT, a data element, characterized by attributes and ports, but without an internal structure. Attributes are described by a data

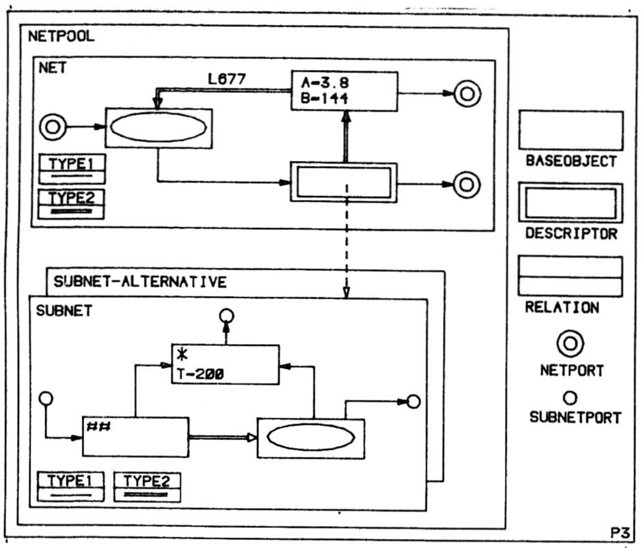

Fig.3: Problem oriented data model of GRIMBI

```
+----------------------------------------------+
| DESCRIBE                                     |
| 1 <usertype> BASEOBJECT [<attriblist>]       |
|                         [<portlist>]         |
|                                              |
| <attriblist>::= 2 <attrib> [,2 <attrib>](0:n)|
| <portlist>  ::= 2 <port>   [,2 <port>](0:n)  |
| <attrib>    ::= <attribname> ATTRIBUTE       |
|                        <datatype>            |
|                        [DIM(<dim>)]          |
|                        [VSET(<vset>)]        |
|                        [DEFAULT(<def>)]      |
|                        [CLASS(<class>)]      |
|                        [REQUIRED]            |
| <port>      ::= portname>  PORT <porttype>   |
|                 [DIM(<dim>)]                 |
|                 [FAN(<min>,<max>)]           |
|                 [<port-attrlist>]            |
| <datatype>  ::= BIN FIXED(15) | BIN FIXED(31)|
|                 BIN FLOAT(21) | BIN FLOAT(53)|
|                 CHAR(<n>)     | BIT(<n>)     |
| <vset> ::= (<from> TO <to>) | (<v_1>,.,<v_i>)|
| <porttype>   ::= IN | OUT | INOUT           |
| <port-attrlist> ::= 3 <attrib>               |
|                     [,3 <attrib>](0:n)       |
+----------------------------------------------+
```

Fig.4: GRIMBI-DDL syntax of BASEOBJECT

```
+----------------------------------------------+
| DESC 1 AND  BASEOBJECT STEXT('AND-GATE'),     |
|      2 TYPE ATTRIBUTE CHAR(3) DEFAULT('TTL') |
|                       VALUESET('TTL','ECL'), |
|      2 IN   PORT IN DIM(2) FAN(0,1),         |
|      3 NEG  ATTRIBUTE STEXT('Signal-negat.') |
|                       BIT(1) DEFAULT('1'B),  |
|      2 OUT  PORT OUT  FAN(0,5)               |
|      3 NEG ATTRIBUTE BIT(1) DEFAULT('1'B);   |
+----------------------------------------------+
```

Fig.5: GRIMBI-DDL example for BASEOBJECT

type, a valueset, and a default value. Arrays of attributes are allowed. Ports together with relations define how the objects may be emdedded in an environment, how objects may be connected to a problem structure. Ports are of type input, output or bidirectional. Port arrays are allowed too, the required and permitted number of connections may be given. FAN(1,1) for example means, that the port has to be connected to another port once and only once in a valid model. The syntax of the BASEOBJECT is shown in fig.4, an example in fig.5.

A RELATION describes the possibilities, respectively the restrictions, of relations between pairs of objects. Fig.6 shows the syntax for a RELATION, fig.7 an example of a relation to construct electrical supply nets. The instance of a user relation type is graphically represented by a legend symbol (E_NET1). A relation may be characterized by attributes similar to the baseobjects, for example by the VOLT attribute in fig.7. Additionally, there is a special attribute for relations, the TUPELATTRIBUTE, related to an element of a relation, i.e. a connection. The key issue of a RELATION- definition in GRIMBI is the formulation of modelling restrictions. The FROMDOMAINE/TODOMAINE-clauses contain all object types, allowed to be start points or end points of a connection. For example: a connection TRANSF1_OUT - PUMP1_IN in fig.7 is allowed as a connection of type E_NET. The TYPE-clause restricts the structure of a relation to a certain type, for example the type TREE, which prohibits loops. Restrictions, related to attribute values of object pairs or relations, are described by the RESTRICTION -clause. In the example, this clause reclaims the equality of the voltage values of the connection and the two connected ports. To be able to use more complex restrictions, there exists the possibility to insert special user algorithms to check the consistency. R_ENTRY_CO-algorithms are called after each CONNECT, R_ENTRY_CL are called while closing a subnet.

The type COMPLEX has been provided as a means to define macros, composed by baseobject and relation types. Attributes of a COMPLEX object are related to attributes of its elements.

The DESCRIPTOR-SUBNET data type characterizes a subnet type relative to two levels of abstraction. The DESCRIPTOR part represents the subnet at the higher level like an object of type BASEOBJECT, but with an internal structure and a variable number of ports. Theses ports correspond to the subnet ports, belonging to the lower level of abstraction. Synthesis of a subnet is restricted to usage of a specified set of BASEOBJECT, RELATION and SUBNETPORT types. Subnets are the regions of interest and therefore may be handled as a whole.

```
DESCRIBE 1 <relname> RELATION
                    [<attriblist>]
                    [<tupelattrib>]
                    [FROMDOMAINE(<fromlist>)]
                    [TODOMAINE(<tolist>)]
                    [TYPE(<strutype>)]
                    [RESTRICTION(<restr-list>)]
                    [R_ENTRY_CO(<entylist>)]
                    [R_ENTRY_CL(<entrylist>)]
<restr-list>::=<operand><operation><operand>
        [,<operand><operation><operand>](o:n)
<operand>    ::=  FROM.<attrname> [(<ind>)]
               |  FROM..<attrname> [(<ind>)]
               |  TO.<attrname> [(<ind>)]
               |  TO..<attrname> [(<ind>)]
               |  REL.<attrname> [(<ind>)]
               |  <const>
<operation>  ::=  = | ¬= | < | > | <= | >=
<tupelattrib>::= 2 <attrname> TUPELATTRIBUTE
                              <like-attrib>
<strutype>   ::= TREE | ...
```

Fig.6: GRIMBI-DDL syntax for RELATION

```
DESC 1 E_NET RELATION STEXT('SUPPLY NET')
             FROMDOMAINE(TRANSF.OUT,...)
             TODOMAINE(PUMP_IN,LAMP_IN,...)
             TYPE(TREE)
             RESTRICTION(TO..VOLT=REL.VOLT,
                         FROM..VOLT=REL.VOLT)
             R_ENTRY_CO(POWERSUM)
             R_ENTRY_CL(FREE_PORTS),
      2 VOLT  ATTRIBUTE INTEGER DEFAULT(380)
                        VALUESET(220,380),
      2 DIAMETER TUPLEATTRIBUTE REAL
             VALUESET(1.TO 10.) DEFAULT(1.5);
```

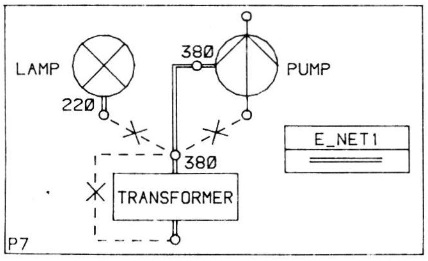

Fig.7: GRIMBI-DDL example for RELATION

DDL-GRAPHICS

A special feature of GRIMBI-DDL is the capability to describe graphic symbols for the external representation of the data. This feature corresponds to a FORMAT statement of programming languages which also defines the external representation of some information. The terms to describe the symbols reflect the logic structure of the objects. The BASEGRAPHIC expresses the basic meaning of the object, ports are described by the PORTGRAPHIC. Attributes may be related to TEXTWINDOWS or to sets of symbols (ATTRIBUTE-GRAPHIC). This facility allows to visualize an attribute value alternatively by a text string in a text window or by a symbol, choosen automatically out of the set in dependency of the actual value. These parametrized graphic symbols are useful in visualizing database contents (e.g. variation of a fluid surface in a vessel, shown in fig.8).

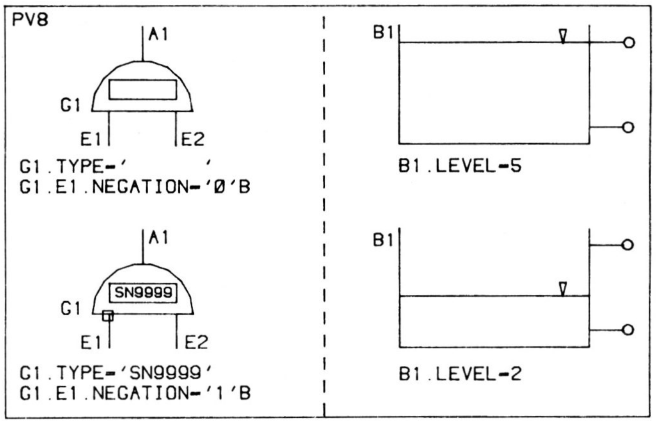

Fig.8: Parametrized symbols

Additionally PORTCOORDINATES (the start points of connections) and PICKAREAS (regions to identify objects, attributes or ports) are described with the DDL. Similarly the representations of relations are defined. A relation instance in a block diagram is the usual legend, expressing expressing the meaning of a set of lines (fig.7).

The DESCRIPTOR symbol has to be surrounded by a rectangular "port position line" on which the ports are shifted if the corresponding ports of the related subnet are shifted. Subnetports in the related subnet may be allocated and shifted only on a "position line" too (fig.9). Subnetports and ports of the related descriptor, though representing logically the same, look different, but they have the same relative position to the descriptor respectivly the subnet. This position naturally is a run-time value. If for example an instance of subnetport is created (INSERT), the related symbol of a descriptor port is automatically added at the related position. Subnets are visualized by block diagrams completed by particular symbol, the drawings legend, which represent attribute values of the subnet.

To describe the symbols the GRIMBI-DDL uses the graphic primitives POINT, POLYGON, CIRCLE, TEXT, characterized by usual attributes.

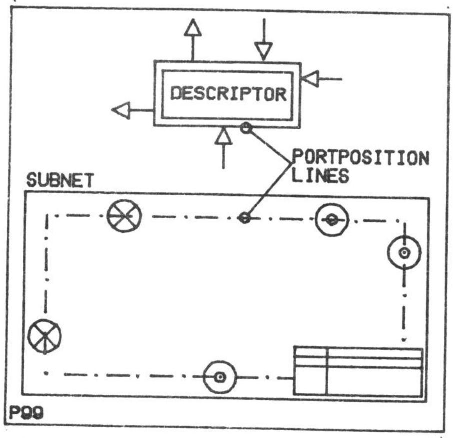

Fig.9: DESCRIPTOR-SUBNET graphics

3. DATA MANIPULATION

After initialization of a GRIMBI-BANK for a special model class (schema), modelling is supported by a many data manipulation operations, bounded by the model class restrictions. As the schema is interpreted at run time, the GRIMBI user may switch easyly between different data bases and to handle two bases at the same time (split screen), one for modelling, the other for reading or graphical editing, that means for example shifting of symbols or routing of connections. A model is represented by a NETPOOL, containing one NET and several SUBNETs (fig.3). According to the management of technical drawings, a NETPOOL is related to a complete set of drawings, a NET to the general sketch and the SUBNETs to the detail drawings. To each SUBNET the system may hold design alternatives, having the same interface description (DESCRIPTOR).

Regions of interest (NETs, SUBNETs) are accessed by the following operations: OPEN - to get the region of interest by name; CLOSE - to leave it; JUMP - to switch from a subnet port in the actual subnet to the related subnet; ALTERNATIVE - to go from the actual subnet to a design alternative, belonging to the same descriptor; DETAIL - to switch from the actual net/subnet to a subnet, pointed to by a specified descriptor; ABSTRACT - to actualize the net/subnet containing the descriptor of the current net/subnet.

Structured modelling is supported by these operations: INSERT(DELETE) NET/SUBNET; DETAIL NEW - to insert a subnet, detailing a specified descriptor of the actual net/subnet; ABSTRACT INTO - to insert a descriptor, representing the actual subnet, into a higher level of abstraction; ALTERNATIVE NEW - to generate a design alternative to the current subnet; CONTRACT - to replace a part of a subnet/net by a descriptor, creating automatically a subnet, containing the removed structure; EXPAND - to replace a descriptor by one of its subnets.

Basic modelling operations are: INSERT (DELETE) BASEOBJECT/RELATION/COMPLEX/DESCRIPTOR/SUBNETPORT/NETPORT - to create or delete instances of the appropriate types; CONNECT/DISCONNECT - to insert/delete a connection of specified type (relation); COPY - to copy an object group with all its internal connections within the actual net/subnet, or into a new subnet, or to copy a subnet into the actual net/subnet. The CHANGE command is used to change attribute values of an object. JOIN joins the tuples of two relations of the same type, SPLIT tranfers a tuple subset of one relation to another of the same type.

All these operation involve both the problem data and their graphical representation. However, there are many operations, which deal with graphics only. They serve for improvement of the visual representation of the information, without modifying their semantic meaning. These graphic editing commands can neither change the topology of the model nor the symbols or the data, but they are used to change the route of connections, to shift, rotate, or scale symbols or groups of symbols, and to change the visibility of graphics information, in order to clarify the picture.

Besides logical zooming, based on the DESCRIPTOR-SUBNET-concept, graphical zooming, and panning are available.

Navigation in the model structure for analysis is done using the SEARCH operation, starting from a "actual position", a group of system managed data, consisting of the "actual object" and the "actual relation". Theses positional data may be pushed/popped. Direct search is possible too. Access to data of an actual object or relation is supported by appropriate statements.

4. SYSTEM ARCHITECTURE

The implementation environment of GRIMBI is shown in fig.10. GRIMBI is running on an IBM 370/168-3033 and a TEKTRONIX-4081 intelligent graphic terminal. Basic software on the host is the integrated kernel system REGENT /3/ (an ICES like system), sometimes called "engineering methods base system"; on the satellite the TEKTRONIX-DGSS software /4/ is used.

GRIMBI is composed of (fig.11) two REGENT subsystems (DBANET: data base administration for nets, DBMNET: data base management for nets) and a module running on the satellite (IGSNET: intelligent graphic satellite for nets). Additionally, fig.11 shows two other REGENT subsystems, which have been proved to facilitate

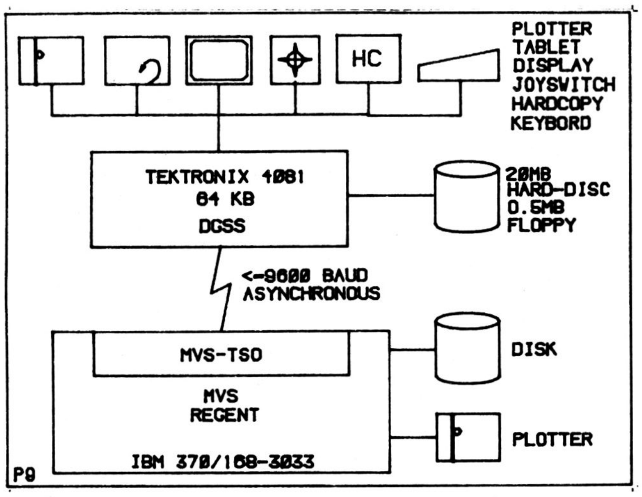

Fig.:10: Hardware and software used by GRIMBI

the implementation of new subsystems. PLS supports the definition of subsystem POLs and GIPSY, the REGENT graphic subsystem, allows to handle 2D and 3D graphics and to convert graphic representations into a portable graphic metafile. All REGENT subsystems may be used to implement new subsystems. The association of GRIMBI to REGENT provides a useful integration of batch activities (analysis of models) with interactive tasks (model synthesis).

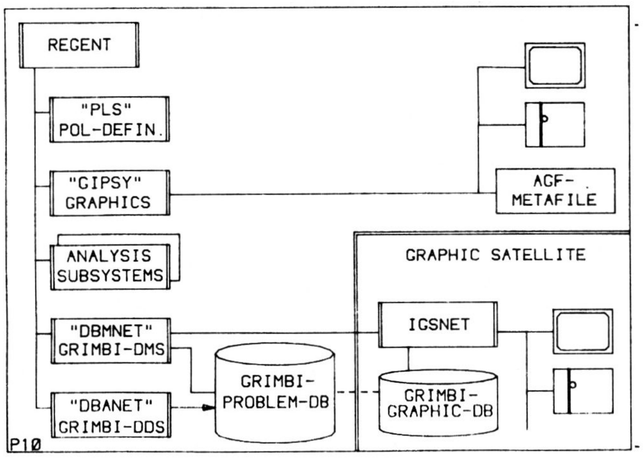

Fig.11: GRIMBI and REGENT

LOAD DISTRIBUTION STRATEGIES

Distributed systems aim at optimum utilisation of the available resources. A particular goal is the reduction of response time.

Three basic strategies may be distinguished with respect to the assignment of tasks to the host or satellite respectively /5/:

- Fixed distribution: All tasks are divided into two classes; One class is always allocated to the satellite, the other to the host.
- Controlled distribution: Some tasks may be allocated to either the host or satellite depending on explicit statements of the operator at run-time or system initialisation time.
- Dynamic distribution: Some tasks may be allocated to either the host or satellite depending on the actual work load on these partners and on characteristic values of the task. The system itself performs the allocation.

Combinations of these strategies may be found. However, as tasks should be considered as steps in a process /6/, whose state is characterized by a possibly large set of data, shifting a task from the host to the satellite or vice-versa requires transfer not only of the operations (which could be implemented on each partner computer) but of all data that represent the process state. As the data link between host and satellite has a rather limited capacity in our environment (as in many others), optimum response performance calls clearly for a design that minimizes data transfer. Thus, no benefit can be taken from the conceptual advantages of dynamic (and also controlled) distribution of tasks. Consequently, for GRIMBI a fixed distribution of tasks was chosen.

THE FIXED DIVISION OF LABOUR IN GRIMBI

The task allocation is primarily determined by the system response time, which the operator expects in a certain design situation. Therefore in /5/ three classes of tasks are distinguished: lexical, syntactic and semantic. For each of this classes, the operator will accept a typical response time. For GRIMBI we decided to handle all tasks of lexical and syntactic level on the satellite, that means tasks, which should be executed in a few seconds and less. The distribution of tasks (functions and data) therefore is as shown in fig.12.

Of course, the distribution is always a matter of judgment. In particular, the subtask of checking problem restrictions using the schema information alone (e.g.: checking attribute values for validity or checking special kind of connection restrictions) is a candidate for being allocated to the satellite. This would imply duplication of the logic schema for the satellite, to prevent schema access via data link. However, input checking involving context information (problem data) always remains as a host task, because the problem data base management, including analysis, requires resources which are available on the host only.

```
+--------------------------------------------+
|  SATELLITE                                 |
|                                            |
|   dialog handling                          |
|   split-screen management                  |
|   data representation graphics             |
|     - positioning of symbols               |
|     - routing of connection-lines          |
|     - identifying data (pick)              |
|   editing of blockdiagrams (DRAWING)       |
|   plotting                                 |
|                                            |
|   graphic data base                        |
|     - symbol library (graphic schema)      |
|     - blockdiagrams                        |
|                                            |
|  HOST                                      |
|                                            |
|   data definition (model class definition) |
|   problem data management                  |
|     - handling of problem restrictions     |
|       using the logic schema               |
|     - handling of problem restrictions     |
|       using context informations           |
|   data base analysis                       |
|                                            |
|   problem data base                        |
|   - schema                                 |
|   - problem data                           |
+--------------------------------------------+
```

Fig.12: Task distribution of GRIMBI

PARALLEL EXECUTION OF SUBTASKS

The task (or process /6/) concept of distributed systems offers the feature of parallel execution of tasks on the host and the satellite. For illustration we use the command INSERT, starting the following sequence of subtasks:

1. input of a command
2. interpretation of command (resulting in the display of a graphical symbol)
3. positioning of symbol
4. insertion of symbol in graphic data base
5. checking of validity
6. if ok then insertion of data in problem data base else error message
7. if error message then delete symbol from graphic data base

There are two groups of tasks: (1) editing of the graphical data representation, burdened with human interaction, and (2) handling of problem data. As both groups require a few seconds each, response time could be gained by parallel execution as shown in fig.13.

```
+-------------------------------------------------+
|+-------------------+!                           |
|| 1.getting a command|!                          |
|| 2.interpretation of|!                          |
||   command          |!                          |
|+-------------------+!                           |
|          ↓          !                            |
|          |----------!-----------+                |
|          ↓          !           ↓                |
|+-------------------+!+--------------------+|
|| 3.positioning of  |!| 5.checking of       ||
||    symbol         |!|    validity         ||
|| 4. insertion of   |!| 6.if ok             ||
||    symbol in      |!|   then insertion    ||
||   graphic data base|!|   in problem base   ||
|+-------------------+!|   else error message||
|          |          !+--------------------+|
| synchron. |          !           ↓                |
|          |----------!-----------+                |
|          ↓          !                            |
|+-------------------+!                           |
|| 7.if error message |!                          |
||    delete symbol   |!                          |
|+-------------------+!                           |
|          SATELLITE ! HOST                        |
+-------------------------------------------------+
```

Fig.13: Parallel subtasks

AUTONOMOUS SUBTASKS

To improve the response time behaviour is one goal in implementing a CAD-system on a host/satellite system. But another goal of using such a configuration with two processors is to achieve maximal independence of both subsystems, to be able to use parts of the CAD-system, probably with reduced demands on the abilities. This improves the over-all availability of the CAD-system. Typical subtasks, which should be executable in a stand-alone manner (without support of the partner computer) are for the host:

- to parse and analyse the problem data base
- to change values of elements for parametric analysis of the model
- to define new models in batch mode using a problemoriented alphanumeric language (Addition of the graphical representation of the data should then be possible interactively later on).

The satellite should be able to process the following subtasks without host interaction:

- to draw blockdiagrams without problem dependent checking (DRAWING)
- to browse and edit data representations (blockdiagrams belonging to a problem data base)

To solve these problem without host support, not only representation graphics has to be available, but also parts of the problem data: the model topology, to prevent topology violations while editing blockdiagrams. Therefore an independent, self-contained subtask for blockdiagram manipulation is implemented on the satellite. These blockdiagrams, usable as drawings and in host interaction (MODELLING) mode as graphic representation of problem database data, are characterized by:

- a set of symbols with ports, graphic attributes (to supplement the basic symbols, f.e: signal negation point at a gate port) and text windows (describing position, lines, columns) for attached texts,
- a set of relation-symbols, each representing a set of connections of the same type in a blockdiagram (connection legend)
- connections of given types.

Using these independent abilities, GRIMBI is able to realize a usefull "split-screen"-technique, which allows to work effectively with two subtasks at the same time.

(1) modelling a system with host interaction, obbeying problem restrictions (e.g. fault tree synthesis)
(2) browsing or editing another data base of probably other model class (e.g. R&I-data) to get information about how to model (fault trees). This is done without host interaction.

Implementing a CAD-system like GRIMBI in a timeshared host environment using a satellite to improve the response time behaviour and the overall availability of the system (including subsystem availability), a fixed distribution of labour and data is a usable solution, especially, if the data link is the main bottleneck. Analysis of the CAD-system should be done with regard of subtasks (or subprocesses) instead of subfunctions.

Thereby two subgoals should be considered:

(1) Separation of parallel executable subtasks, to be able to optimize the response time behaviour of the system, if a multitasking operating system available on the satellite
(2) Separation of autonomous subtasks to be able to use parts of the CAD-system with only one processor.

5. CONCLUSION

GRIMBI is a CAD-system supporting functional modelling. In a data definition phase, model classes with problem oriented modelling restrictions are described, using the GRIMBI-DDL, based on a data model, suited to functional modelling. Modelling operations are bound to theses model classes, thus enforcing consistency of the models. Modelling is supported by working methods, usual in systems sythesis: stepwise refinement/abstraction, management of design alternatives. Model analysis algorithms are written using the GRIMBI-DML, an extension PL/1. Special DML-statements are available to facilitate navigation in the data structure. GRIMBI is implemented as a subsystem of REGENT on a main frame, delegating interactive and graphic tasks to a satellite. It enhances the batch system REGENT by a problem oriented data base component and a graphics oriented dialog component. GRIMBI is presently used in safety analysis of nuclear plants for synthetizing fault trees, which are used as a basis for analysing the risk of nuclear energy.

ACKNOWLEDGEMENT

The author would like to thank Mr. E. G. Schlechtendahl for his helpfull comments during the preparation of this paper.

REFERENCES

/1/ Caldarola,L.: Generalized Fault Tree Analysis Combined with State Analysis. Report Nr.: KfK-2530, Kernforschungszentrum Karlsruhe, West Germany, 1980
/2/ Leinemann,K.: Ein System zum funktionellen Modellieren unter Verwendung von Datenbanktechniken und interaktiven graphischen Arbeitsmethoden. Report Nr.: KfK-3217, Kernforschungszentrum Karlsruhe, West Germany, 1981
/3/ Schlechtendahl,E.G.,Leinemann,K.: The REGENT-system for CAD. In: Allan,J.J.III: CAD Systems. IFIP Working Conf. on CAD Systems, Austin (Texas), North-Holland Publishing Company, Amsterdam (1975).
/4/ TEKTRONIX: Plot80-DGSS Reference Manual. Beaverton(1978)
/5/ Foley,J.D.: A Tutorial on Satellite Graphics Systems. Computer (1976)8 p.14-21
/6/ Schlechtendahl,E.G.: CAD Processes and System Design. In: Encarnacao,J.: Computer Aided Design, Modelling, Systems Engineering, CAD-Systems. Lecture Notes in Computer Science Vol.89, Springer, Berlin (1980).