The EQUIP system: a product type modeling system developed with the MAM method.
J.L.H. Rogier, M.G. Dittrich
TNO-Institute of Applied Physics, PO box 155, 2600AD Delft, The Netherlands
Email: jrogier@tpd.tno.nl, dittrich@tpd.tno.nl

Abstract
The TNO Institute of Applied Physics has developed the Multi Aspect Modeling (MAM) method for building product type modeling systems and product modeling systems more efficiently and has developed tools to support the process of building such systems. The EQUIP (Work Methodology for Development of Quiet Products) system is a design supporting tool for low noise design of mechanical products. The EQUIP system is the first of a series of product type modeling systems that has been built with this method. In this paper the MAM method is illustrated using the process of building the EQUIP system as an example. In the first part of the paper it is discussed how to differentiate between different types of knowledge when building a knowledge based system. The second part of the paper focuses on the architecture of the EQUIP system and the underlying BiCad data structure. In the third part of the paper the process of configuring the EQUIP system is discussed and a strategy for design support knowledge representation is introduced. In this part the term aspect vocabulary is introduced to identify the collection of terms used by a domain expert to describe a specific view on, or aspect of a product. The fourth part of the paper elaborates on the strategy for design support knowledge representation and a distinction is made between product type modeling systems and product modeling systems. Also the relation between both kinds of system is discussed and a strategy is proposed how to use the first to build the second.

Introduction
MAM is a method for building product type modeling systems and product modeling systems. It consists of a working approach to build these systems modularly and based on a specific internal data structure that is aimed at the representation of modeling data: the BiCad data structure [ROG1].

The EQUIP system is a design support tool for low noise design of mechanical products. It has been developed with the EQUIP project [DIT1],[DIT3], a European project funded within the Brite-Euram programme of the EU. EQUIP stands for “Work Methodology for Development of Quiet Products”. The project was carried out by TNO (NL, coordinator), BeSB (D),
University of Clausthal (D), Cetim (F) and the industrial partners Caterpillar (B), CIAT (F) and Faun (D), running from 1992-1996.

One of the project objectives was to provide a designer of machinery and equipment with a system to access specialist knowledge of low noise design. From the beginning of the project it was decided that a system should be built in which domain knowledge would be separated as much as possible from the modeling support facilities of the system. In other words, the aim was to build a general design supporting tool that, through the use of domain specific knowledge libraries would be adapted to design support for a particular knowledge domain: in this case the domain of acoustics and in particular: low noise design.

In an early stage of the project this resulted in a distinction between different kinds of knowledge to be stored in knowledge databases. The following knowledge types were defined:

- **component** knowledge representing the vocabulary used by domain experts to project their view on a piece of machinery (i.e. the acoustic vocabulary):
- **function** knowledge representing mathematical dependencies between attributes of terms that are used to describe a piece of machinery for a specific domain expertise:
- **modeling** knowledge representing various approaches that a domain expert takes when analyzing or designing a piece of machinery,
- **conversion** knowledge representing the conversion and translation rules required to project a different view on a product:
- **measured** data used to fill in parts of the description of products with default values.

Furthermore, the system should provide various kinds of help facilities in order to explain the semantics of each knowledge element stored in the systems libraries and on the working of the system itself.

The working methodology for building the system used prototype refinement as a basis. Based on the earliest prototypes of the system, requirements were specified concerning concurrent **alternative** product models, **validation variants** of the same product model and the possibility to project **different views** on a single model. To fulfil these requirements the BiCad data structure was used as a basis for the kernel part of the system.

The EQUIP project focused on building domain specific knowledge libraries and in order to support this process to build a set of tools that would help to design and fill these libraries. At the end of the project the working method that was initially developed with the EQUIP system, was supported by a set of software design support tools that would help to apply this MAM method.
The EQUIP system

Systems architecture
One of the major requirements of the EQUIP system was that it would anticipate developments within the course of the project: it had to be an open system. [ROG2][ROG3].
The system development and the corresponding architecture were inherently dependent on other tasks within the project. The main project tasks were:
1. Product specific acoustics,
2. Design methodology,
3. Development of a knowledge based system,

Initially a global system concept was defined as a working basis. This concept consisted of:
- a generic kernel modeling support facility that should translate all modeling actions into manipulations on the BiCad data structure used to store the product, and
- EQUIP specific system elements:
  - an EQUIP user interface to the modeling facility,
  - a collection of EQUIP libraries with various kinds of formally defined domain specific modeling elements which should be addressed from the context of the model, and
  - a collection of EQUIP help libraries that could explain each piece of knowledge.
Calculation should take place outside, although controlled by, the EQUIP system through the use of (preferably existing) calculation programs.
  - Control over the EQUIP system should take place by means of a macro facility that would allow the representation of modeling procedures. These macros would also be stored in a system library representing pieces of modeling knowledge.

Figure 1 Global system architecture
The BiCad data structure

The BiCad data structure is central to the system. It is a highly complex, decomposition based data structure. It was originally developed in the context of a Dutch national research project on integration of AI and CAD: the IIICAD project [TOM], and was elaborated for modeling support in the EQUIP project. The main element in the structure is Part. A part has attributes that have a value and is by definition part of a higher aggregation level part. The functionality defined on the data structure meets the following requirements.

- it allows presentation of different views on a single Part.
- it allows existence of different alternative decompositions of the same Part in a single model.
- it allows different values and marked sets of values (called operating conditions) for attributes of a Part.

The BiCad data structure is illustrated in a simplified form in figure 2 using the NIAM [NIJS] representation.

Figure 2. The BiCad data structure (simplified)

This data schema defines a (product) Model (which is itself a subtype (specialization) of the modeling entity Alternative) as built from a collection of Parts. Each Part has a collection of Alternatives from which, for calculation purposes, one is the current alternative. Each Alternative contains a collection of Parts.

A Part has a collection of attributes with a (local) name. Attributes have a collection of values of which one is the current value. A value may be manually set (by the user or read from a database) or be the result of calculation through a function. When a value has to be the result of a function, the name of
this function and the origin of its parameters are stored with the function call
definition so that each time the value of the attribute has to be updated, the
system can generate a call to an external calculation program that will execute
this function.

Although each attribute has a name, the actual semantics of a part is de-

defined by a reference to an element in an external component definition library.
Such a library defines domain specific component types (in the case of the
EQUIP system, *acoustics* components with their own collection of attributes).
When linking a component definition to a Part, the system generates a mapping
table that specifies which defined component attribute name is linked with
which part value. Because more that one domain specific component (from dif-
ferent domains) can be linked to a Part, more that one domain specific compo-
nent attribute name can be mapped on a single Part attribute value; in other
words the same Part attribute may have different names in different views.

For example: when an acoustic component called *axial_fan* has a part
*rotor* that defines an attribute *length_of_blade* which is relevant for a certain
acoustic calculation, the value of this attribute may also be useful to define the
cylindrical space that is occupied when the fan is rotating. Geometrically
speaking this same attribute therefore is also the *radius* of the geometric com-
ponent *cylinder*.

To model this with the BiCad data structure the same Part has an *acoustic
component rotor* and a *geometric component cylinder* linked to it where by re-
spectively the attributes *length_of_blade* and radius refer to the same part at-
tribute. This attribute may have a local name which is arbitrary: when the Part
is viewed from an acoustic viewpoint the attribute is referred by as
*length_of_blade*, when viewed from a geometrical viewpoint the same attribute
is referred by as radius. As a consequence changing the value of the attribute
*length_of_blade* in the acoustic view will automatically adjust the value of ra-
dius in the geometric view and vice versa.

**Characteristics of the EQUIP User Interface to the BiCad data structure**

The user interface of MAM-method based systems is the main means of access
to all other parts of the system. It provides access to:

- the kernel data structure, by means of data type dependent editors and pop-
  up menus;
- system databases used to store component structure definitions and call
definitions for calculation routine;
- various help facilities;
- a macro facility that allows execution of modeling sequences.

The user interface is specific to an application domain because it is intended to
simplify the declaration of data [DIT2].
A particular feature of the EQUIP system is the large number of component definitions in the component definition library. For this reason, selection of a component type takes place after adding a component to the model and by means of a pop-up menu that presents a structured overview of the contents of the database.

Depending on the calculation type, acoustic modeling of the real world location of parts is not always relevant for a calculation. The number of connections between parts and the type of connection is more important. For this reason the EQUIP system uses an iconic user interface where every component is represented by means of a icon, which can be connected to other icons by means of different types of links.

Both decomposition and declaration of component alternatives is supported in the BiCad data structure by building a hierarchy of diagrams on components. The drawing field in the user interface presents a single diagram, and browsing through the hierarchy is supported by selecting a component and opening the diagram on one of its alternative internal structures.

Validation of attributes of a component takes place by selecting the component and selecting an attribute of this component through a pop-up menu. Depending on the type of attribute as defined in the component definition database, the system answers with a type specific data editor. According to the BiCad data structure, an attribute can be validated either by a straightforward value or a function call. In the case of a function call the system will respond, if possible (if the external calculating program runs in the background and when all parameters are values or refer to values), with the calculated value.
Special data types such as, in the case of acoustic modeling, spectra are supported by internal data structures specific to the kind of MAM-method based system.

**Knowledge representation**

The whole aim of the MAM-EQUIP system approach is the separation of modeling functionality from applied domain knowledge and to make an open system for entering all kinds of domain specific knowledge. From an implementation point of view this means anticipation to open industrial standards like Open Data Base Connectivity and a client-server architecture. With the current implementation of the system on the Windows® platform this approach resulted in the use of ODBC and DDE.

At one abstraction level higher, the open system approach led to the definition of different types of knowledge libraries that should be used in the system. Definition of the type of library was based both on the origin of the knowledge and the application of it in design processes. A distinction was made between:

- component knowledge,
- calculation knowledge,
- modeling knowledge,
- conversion knowledge, and
- straightforward product knowledge or sources of measured data.

The end user is free to update each of the knowledge sources which are stored in different data bases and accessed through the ODBC mechanism or in the case of calculation addressed through DDE.

**Configuring the EQUIP system**

At this point in the paper the global architecture of the EQUIP system as designed at the end of the second project year has been described. The next part contains a description how to build the system.

To build the system a formal approach was taken. At the start of the project the project partners agreed upon the use of NIAM, Express(G) [EXPR] and SADT. This choice was made in 1992. With the current state of the art in system design, OMT [RUMB] or Shlaer-Mellor [SHLA] is a more adequate choice. At that time these techniques were not yet commonly known and experimenting with these would have been too time consuming.

- The system was implemented in the programming environment Smalltalk [GOLD], for two simple reasons. Smalltalk is most suitable for building prototype software (and currently also to build run-time executables) and allows major structural changes to be implemented rapidly.
- All software was developed in an iterative process of prototype refinement supported by formal specification and documentation of the applied data
structures and program elements. To support this process, NIAM was used as a basis for data definition. A customized NIAM editor developed at TNO was used that was able to generate Smalltalk basic code and documentation on the implementation.

Libraries were defined by means of a formal specification in Express(G) and by means of a customized editor developed at TNO. This editor can generate data bases in a format required by the MAM-method based system kernel. In practice there is no great difference in expressive power between NIAM and Express. However, to use the first mainly for software data structure definition and the second for database structure definition allowed tuning both modeling tools for specific needs. The reason to choose Express(G) was the conceptual correspondence between the BiCad method and the approach of the international standardization effort on product data definition with STEP. The work on acoustic data modeling in the EQUIP project can easily be compared with the work on other aspects of product modeling as found in STEP [STEP]. Using Express will make it possible in future to integrate results of the EQUIP project with STEP.

Modeling knowledge was represented in macros. SADT was chosen to define macros graphically.

**Library definition**

The Express(G) editor is a central tool for defining the libraries. This editor is used to define the structure of the component and function call library. One of the major issues with tool development was to customize this editor to support the integration of schemes originating from different partners and to automatically generate as any MAM-method based system libraries as possible from this schema.

The structure and definition of acoustic components was provided by the Institut für Maschinenwesen (IMW) Clausthal University, Germany. Calculation function calls were provided by TNO. The Express(G) editor was used to integrate the component schema of IMW with the function call Schema of TNO into a single schema. Integration of these schema’s into a single schema allows type checking of function results with attribute values such that when choosing a function call for an attribute value the system may look up which functions are appropriate. This schema is called an Aspect schema or Aspect Vocabulary because it is a collection of all terms used to describe an aspect of a product.

The Express(G) editor also provided parts of the MAM-method based system that indirectly depend on the Aspect schema:

- the system uses icons to represent components on the drawing field of the user interface. There is a one-to-one relation between the component and the icon. Therefore the Express(G) editor includes a facility to add an icon to a component.
There is a one-to-one relation between help information on a component or a function call and the component or function call. Therefore the Express(G) editor includes functionality to link help data to a component definition of function call.

Lastly, every MAM-method based system generated for a specific domain (e.g. acoustics) is expected to manage domain specific data types such as for instance the type **Acoustic Data**. The definition of this data type should correspond with the one found in the aspect vocabulary on acoustics. In order to guarantee correspondence between both definitions the Express(G) editor includes functionality to generate implementation code (in this case Smalltalk) for a domain specific design support tool. Smalltalk’s feature to use link libraries on parts of the executable code, guaranteed the possibility to modularly upgrade the system in future.

**How to build a MAM-method based system**

During the course of the EQUIP project a stepwise approach has been developed for building MAM-method based systems. Although in this project these steps were not taken sequentially, in finalizing the project it is possible to describe how future projects can be organized. At TNO currently about five different projects have been started which will result in a new MAM-method based system. The main advantage of this approach is that, because each of these systems use the same internal representation, it is easy to let a department semantically enrich a model on a product built by another department for their own purposes and without the burden to build a complete product model from scratch.

The structure of this strategy is illustrated in figure 4.

![Tooling support for BiCad method for product type modeling systems.](image)

- The first step is to define an aspect vocabulary. An inventory has to be made of component-oriented terms used by experts when describing their
view on a product: which attributes they use to describe these components, and which functions are used to generate the value for attributes based on the values of other attributes. When taken in mind that the system is intended to apply existing calculation modules, an inventory of what these programs require as input may be a good starting point.

The next step is to enrich the elements in the aspect vocabulary with additional data required for implementing the libraries and the system’s user interface. This means

- if an iconic user interface is used, definition of the icons; otherwise depending on the kind of user interface, definitions of special component presentation widgets;
- creation of help databases that should be linked to the system and provide help on each of the elements in the aspect vocabulary;
- identification of specific data types (like the type Acoustic Data) that will require special handling in the system.

After defining each of these elements the Express(G) editor will generate the appropriate databases and part of the implementation code on specific data types.

The next step is to define the interface to existing calculation programs. Depending on the calculation program facilities this may vary from low level file exchange between the modeling system and the calculation program to more advanced client server communication protocols (the EQUIP system applies both file exchange, DDE communication and OLE automation control).

Building of the system is then a matter of defining the elements of the user interface. Support on this is provided by the customized NIAM editor which also generates documentation of the implementation automatically and in a standard format.

Additionally databases with measured data can be added to the system. For this the Express(G) editor is able to define the data structure required for instance for modeling system specific data types (i.e. Acoustic Data).

Finally, modeling sequences can be defined by means of the (customized) SADT editor which (in the near future) will generate macros that can be used to automate modeling actions.

One step further: Generating product type-specific knowledge based systems

With a MAM-method based system it is possible to build instantiable product type models. If a single domain specific view is involved, such as for instance acoustics, this model implies the acoustic view on the product type crane, bridge, gearbox for instance. If more that one domain specific view is involved, like for instance both acoustics and mechanics, this model implies an integrated view on a product type that allows evaluation of the same collection of product data from different viewpoints.
However it is questionable if an end user always needs the flexibility of a MAM-method based system when evaluating a product. For scientific research purposes it is useful to have a system that allows you to modify a product structure, experiment with different aspect vocabularies, experiment with more advanced calculation functions or simply have a means to document research results in a structured way.

An end-user however, will be more interested in a well defined software tool on a specific type of product that will help him to evaluate this product more easily. In other words the end user will be more interested in a quite specific kind of expert system. This leads up to the subject of building a model structure versus applying a model for product evaluation purposes.

**Product type modeling systems versus product modeling systems**

A *product type modeling system* (for example, the EQUIP system) is a system that is aimed at building a product type model. A *product type model* (for example, an acoustic model for *steel railway bridges*) is a model representing a type of product whereby the aggregation (decomposition) structure is fixed (including the possibility of having different alternatives for decomposing a part in a single model) but whereby the values for attributes are only partly set (through for instance the use of default values) and should be set by the end user.

A *product model* (for a example, a model of an existing *railway bridge* somewhere in the world) is a model of a product. A product model has a fixed structure and a full set of validated attributes describing a single or a couple of states intrinsically of the modeled product. In object oriented terms: a *product model* is an *instance* of a *product type model*.

MAM-method based systems are intended to build product type models. In order to experiment with these models however, they include functionality that allows the user to instantiate them. The product of a MAM-method based system however, is a product type model. Instantiation of such a model should therefore also be possible in the context of a system that is specifically tuned for a product type model. Such a system requires less flexibility than is provided with a MAM-method based system.

**Product Modeling support for acoustics**

At the end of the EQUIP project, a number of example models have been built. In order to use these models, product modeling systems have been built. An important element of the MAM method is to reuse as much data as possible when building a product modeling system after the product type model has been built with the product type modeling system.

In order to find out how to use as much data as possible collected using the EQUIP system, during the last year of the EQUIP project some experiments were performed. One of the experiments concerned the building of an expert system for acoustic evaluation of steel railway bridges. This system is used to
predict the noise production of trains on a steel railway bridge and to evaluate design alternatives.

The basis of the system is a product type model on the combination of a (different alternatives for a) train, the (different alternatives of) steel bridge(s) and the bridge environment. The structure of this model is fixed allowing the user to choose for a specific alternative for both the train and the bridge. The validation of the model is supported by providing access to a limited set of validatable attributes of the model through custom browsers and selection of measured data items stored in a database.

Many elements that can be collected with the EQUIP system can directly be used as a basis for the implementation of the product modeling system.

The result of this effort is illustrated in figure 5 which presents an extension of the strategy used to define a product modeling tool.

To illustrate the process of building a product modeling system, the following description starts at the point that a product type model has been designed with the EQUIP system. So there already exists a vocabulary on acoustics, an extensive collection of calculation functions, icons for presenting component types in the user interface, databases with measured data and extensive databases with help data. What is actually done when building a product type model is selecting a subset of all the EQUIP system data that is aimed at a specific product type.
The next step to be taken is to export this model in a format that can be used as a basis for implementing an product modeling support tool.

It is now possible export a product type model built with the EQUIP system in NIAM model format such that it can be used as a basis for implementing a product modeling system with the NIAM editor.

To include calculation functionality in order to be no longer dependent on external calculation programs it is necessary to compile the calculation functions and their evaluator into a link library. In the EQUIP system MATLAB® is used as a calculation program. Additional tools for MATLAB® provide facilities to generate C++ from functions so that DLLs can be generated.

The result is a product type model, compiled into a product modeling system, built from a predefined set of aggregation levels each presented in a different diagram, a user interface that is mainly inherited from the EQUIP system itself, and a calculation tool that is extracted from a subset of predefined calculation routines.

Conclusion

The MAM method provides an efficient and cost effective way to build product type modeling systems and product modeling systems. The method requires formal representation of domain specific knowledge. This ensures an unambiguous and non-redundant description of domain knowledge which can be used as a basis for further research in that domain. By independent modeling of domain specific knowledge, the MAM method formally allows projection of different domain views on the same product model. The developed EQUIP system illustrates the applicability of the MAM approach.

Planned topics for further research are:
- new product type specific knowledge based systems for low noise design;
- other application domains (new aspect libraries) and new product type specific knowledge based systems for these aspects;
- research on process modeling by separating process class and process type specific modeling

Acknowledgements

The EQUIP project was funded by the European Commission DGXII, whose support is gratefully acknowledged (Contract number BRE2-CT92-0339, project number 5983).

References

Appendix A. Tooling

Parallel to the development of the EQUIP system, development of MAM-based system configuration tools took place. Three different tools were developed, each having a specific purpose for the system configuration. Apart from their purpose for system configuration, each of these tools also served a purpose for system documentation and support in the process of communication between project partners. For this reason each of the tools is based on an internationally accepted graphical, formal specification language. However, although based on a formal language, the functionality provided with each tool is on one hand limited, at the other hand extended to what is useful for configuring a MAM-
based system. Also each of the tools is embedded in the development environment used to build MAM-based system.

The NIAM modeler

NIAM is a graphical conceptual schema and relational database language. Concepts in NIAM are *Entity*, *Label*, *Role*, *Fact*, *Relationship* and *Constraint*. Apart from their use in describing relational databases, these concepts can also be applied when describing the structure of object-oriented systems. NIAM was used due to the lack of internationally accepted standards for describing an object-oriented system at the start of the EQUIP project.

![Figure A.1 The user interface of the NIAM modeler](image)

With the NIAM modeler it is possible to build a NIAM schema of the implementation of an object-oriented system implementation. Besides the drawing functionality, the NIAM modeler provides the following:

1. It allows a user to add a textual description to instances of each of the above-mentioned concepts. The complete model including all text is used to build a document automatically (RTF: rich text format). This document is intended to be used a description of the implementation of a system.
2. It allows the user to generate Smalltalk code automatically. This code is used as a basis for the implementation of MAM-based systems.
3. It allows a user to extend the description of the NIAM model elements with implementation issues: Each NIAM entity is implemented as a Smalltalk Class: a Class is described by its attributes and its methods. Each named
Role is translated to an attribute: methods can be documented as additional information of an entity.

4. As the NIAM modeler is embedded in the same development environment as the MAM-based system it was possible to include functionality that allows comparison of design and implementation of a MAM-based system. This allows to keep the model of the implementation up-to-date with the implementation itself easily.

5. Code generation facilities of the modeler are not fully explored yet. The idea behind this is to maintain consistency between the design of a system in NIAM and the implementation of the prototype in Smalltalk. According to what a user wants, a prototype system will be modified. In order to maintain consistency between model and prototype the NIAM modeler includes functionality to synchronize the model with the implementation. When finalizing the prototype phase the updated NIAM model can be used to build code in a more suitable implementation language. Currently only a possibility to generate SGML is implemented and very limited facilities to generate C-code.

The Express(G) modeler

Express is the international standard language for Product Model Data exchange as defined in the context of ISO 10303. Express(G) is a graphical subset of the Express language. The Express(G) modeler developed in the context of the EQUIP project is aimed at the configuration of MAM-Based systems of which the EQUIP system is an example. For this purpose the Express(G) modeling language had to be extended with some symbols in order to cover a larger subset of the Express language. The extension concerns the visualization of relation between the SELECT type and the entities it selects from, of DERIVED attributes of an entity and of the FUNCTION type. On top of the Express language definition each Express element description is extended with a set of properties that allows control when generating code from models. One undesired element in the definition of Express(G) has not been included in the implementation: according to Express(G) when an entity has a relation with an entity that is visualized in another diagram a rounded rectangular shape has to be added to both diagrams to express this relation. In the Express(G) modeler this concept is replaced by allowing entities to be presented in different diagrams concurrently and then presenting them in a different color.

The Express modeler has a purpose for generating databases that are used with MAM-based systems. One of the most important databases of a MAM-based system is the Aspect Vocabulary database. This database contains the definition of the building blocks of a model for a certain aspect. Apart from the description of these building blocks in terms of entities with attributes and specialization relations with other entities, the aspect vocabulary database also contains data on the definition of icons to be presented in the toolbar or on the canvas of the modeler and the names of help-files that should be included with
the system. Summarizing: the Express(G) modeler is used to organize all knowledge elements that should be part of a MAM-based system. In order not to become dependent on a specific database, the Express(G) modeler generates SQL instructions to build databases.

With the Express(G) modeler it is possible to build an Express(G) schema of the specific databases used with MAM-based systems. Besides the drawing functionality, the Express(G) modeler provides:

1. It is possible to generate a report on a MAM-system database in a standard format that can be used both to support the process of communication when building the system and to document that system.
2. It is possible to extend the regular Express definitions with additional data relevant to build MAM-based system specific databases.
3. It is possible to generate database generation code (SQL) in the specific format that is required by MAM-based product type modeling systems and product modeling systems
4. It is possible to read in text-based Express models and to use these as a bases for building their graphical representation in Express(G). This process is partly automated: the user chooses the screen position of entities and types: the system adds the graphical representation of relations to the diagram.

The SADT modeler

SADT is a process description language. With SADT a process is described as a collection of activities linked to each other by means of data flows. Each ac-
tivity has zero or more input data flows: zero or more output data flows: may have control data input used to control data transformation in the activity and may have resources used to perform the process of data transformation in the activity. On of the first decisions in the EQUIP project was to describe the process of low noise design formally and by means of SADT. The reason to choose SADT was that it was the most well known technique among all partners.

To be able to integrate the results of process modeling in the EQUIP system, a SADT modeler was build that was integrated in the same development environment as the rest of the tools. The idea behind this approach was that a description of a design process could be translated into macro like, context sensitive procedures. These procedures would become part of the design system in the form of a library of design action procedures. This idea has not yet been fully implemented. The SADT modeler used in the project allows to build SADT models: to generate a report on the model in the same format as the other two modelers: it includes functionality that allows to generate a preliminary version of a scenario language that is expected to become executable in future versions of MAM-based systems.

Figure A.3 The user interface of the SADT modeler

Appendix B. Implementation platform dependency

An important aspect of the MAM-technology is platform independence of the results. This was a reason to anticipate as much as possible international standards for formal specification languages and internationally accepted industrial standards. Also the choice of the programming language was based on this con-
sideration. Another important aspect for language choice was its feasibility for prototype implementation. Until, recently Smalltalk seemed to be a good choice. However the current attention for Java requires reconsideration of this choice for future MAM-based systems. Implementation of currently running MAM-based systems apply besides other more low level concepts, the following standards: ODBC, DDE and OLE automation.

**Smalltalk**

Smalltalk [GOLD], [RUMB], is one of the first object oriented programming languages. It was developed at XEROX PARC. In comparison with many other object oriented languages, Smalltalk is a strict object oriented language which means, it is almost impossible to program in another than an object oriented style with Smalltalk. As a consequence, the language is extremely suitable for educational purposes. This gave the language an academic character. Only recently, during the last two years, the development environment included more advanced features to build run time applications. However the increasing attention for Java [SYM] made us reconsider the choice of the implementation language. Although the current support for Java programming is hardly comparable with the support provided with the various Smalltalk environments, it is to be expected that this situation will change very rapidly which makes Java a suitable alternative.

**ODBC-SQL**

One of the major problems with program building and delivery was the availability of database programs at different end-user locations. Instead of developing a database as integral part of the software: we decided to use Microsoft’s ODBC interface [MICR] . This allowed us to distribute SQL code for database creation in the end-users own database format. As SQL is a standard language for almost every available commercial database, the choice for this language on one hand allowed us to develop databases without being forced to buy each database used by an end-user. On the other hand we anticipated future versions of MAM-based systems not only running on Windows® systems.

**DDE**

There are several possibilities to interface with existing software. Dynamic Data Exchange is a mechanism that allows programs to exchange data and commands with each other in a client-server architecture. To be applicable both programming environments should support the protocol. The Smalltalk environment provides both facilities to build DDE-clients and -servers. The program environment MATLAB® used with the EQUIP system; also provides facilities for both these issues. In the EQUIP system configuration a setup was chosen whereby the MATLAB® calculation procedures provided calculation facilities for EQUIP models.
To make this concept work also for complex data structures as for instance acoustic data, we implemented a stack like facility in MATLAB®. In the Express(G) model on acoustic aspect terminology, we defined a general data structure for acoustic data. With the code generation facilities of the Express(G) modeler we generated code on this structure and implemented both this structure as a collection of objects in Smalltalk and as a collection of global variables in MATLAB®. When Smalltalk and MATLAB® communicate about acoustic data they do this by making Smalltalk fill and/or read the contents of this pseudo stack.

**OLE-Automation**

A more advanced client-server communication protocol is OLE Automation [BROC]. At the time we chose to include this possibility for data and command exchange between programs in the EQUIP system, it was one of the least documented facilities of the Microsoft environment. In spite of what the name suggests there is a big difference between OLE and OLE Automation. OLE is a document-part annotation technique. In the Microsoft environment it is possible to build aggregated documents build from different sources which maintain a reference to the application used to build these sources. As a consequence it is possible to include for example spreadsheet data in a text document. When one wants to modify it, it will start up the spreadsheet program from which it originated. OLE Automation is a completely different concept.

The best way to understand this concept is to compare it with access to functionality defined in a Dynamic Link Library: a DLL. Dynamic Link Libraries have as a disadvantage that the program that applies their functions should know in detail how to address it. OLE Automation partly solves this problem by means of a definition file that can be evaluated by the calling program before addressing functionality and by making the Operating System responsible for control on data exchange between OLE Automation Client and OLE Automation Server. Data exchange is further standardized by defining Data exchange elements from which the VARIANT structure is the most flexible.