Interner Bericht

A graphical representation schema for the software process modeling language MVP-L

Alfred Bröckers, Christiane Differding, Barbara Hoisl, Frank Kollnischko, Christopher M. Lott, Jürgen Münch, Martin Verlage und Stefan Vorwieger

270 / 95

Fachbereich Informatik

Universität Kaiserslautern • Postfach 3049 • D-67653 Kaiserslautern

A graphical representation schema for the software process modeling language MVP-L

Alfred Bröckers, Christiane Differding, Barbara Hoisl,
Frank Kollnischko, Christopher M. Lott, Jürgen Münch,
Martin Verlage und Stefan Vorwieger

270 / 95

Herausgeber: AG Software Engineering
Leiter: Prof. Dr. H. Dieter Rombach

Kaiserslautern, Juni 1995

Abstract

Experience gathered from applying the software process modeling language MVP—L in software development organizations has shown the need for graphical representations of process models. Project members (i.e., non MVP—L specialists) review models much more easily by using graphical representations. Although several various graphical notations were developed for individual projects in which MVP—L was applied, there was previously no consistent definition of a mapping between textual MVP—L models and graphical representations. This report defines a graphical representation schema for MVP—L descriptions and combines previous results in a unified form. A basic set of building blocks (i.e., graphical symbols and text fragments) is defined, but because we must first gain experience with the new symbols, only rudimentary guidelines are given for composing basic symbols into a graphical representation of a model.

Table of Contents

Table of Figures

1 Motivation

The Multi-View Process Modeling Language (MVP—L) supports descriptive modeling of software development processes [1]. MVP—L allows the description of elements of software development projects in a textual form. The descriptive modeling activity includes a review of the process models by the people who perform those processes. In recent years MVP—L has been modified in response to evolving requirements. Experience gathered in several real-world projects has shown that pure textual representations of process elements are not readily understood by the project members (for example, see [2]). It is not cost-effective for project members to learn MVP—L just in order to validate the process models. Even the process modeler frequently has difficulty in creating and modifying the textual descriptions, which is caused by the fact that information may be spread over several models (e.g., the aspect product flow).

Different projects in which MVP—L was applied developed their own context-specific graphical representations (e.g., [3]), resulting in conflicting definitions. This report defines a graphical representation schema for MVP—L models to provide a standard for future representations. A basic set of building blocks (i.e., graphical symbols and text fragments) is defined, but because we must first gain experience with the new symbols, only rudimentary guidelines are given for composing basic symbols into a graphical representation of a model. Specific, meaningful representations are not excluded by this approach. We know from past projects that tailoring of representations to specific needs may be required.

A thorough understanding of this report requires some familiarity with MVP—L. The report replaces neither the language report (i.e., [1]) nor experience gathered by modeling some processes.

2 Terminology

An *MVP—L construct* is each non-terminal of the language report [1].[1] The basic MVP—L constructs in the context of this report are: project plans (<project_plan>), process, product, resource, and attribute models (<process_model>, <product_model>, <resource_model>, and <attribute_model>), objects (declared in <objects>), entry criteria, invariants, and exit criteria of process models (<criteria>), interfaces and their relations (<interface_refinement> and <interface_relations>), subprocesses, subproducts, and subresources (<process_object_relations> and <structure_rel>), instantiation parameters (<instantiation_parameters>), attributes (<exports>), and identifiers of models and objects (<ident> and <object_decl>).

An *MVP—L description* is a set of at least one project plan and related models. Models which do <import> each other are considered as related. MVP—L does not contain any other construct to express the relation of project plans and models (e.g., comparable to the construct *module* in Modula-2). Additional mechanisms are required to identify all parts of an MVP—L descriptions (e.g., a file containing all parts of an MVP—L description).

A *refined process* (*refined product*, *refined resource*) has an object specification (i.e., <body>) containing a refinement, which describes the process (product, resource) more precisely by listing subprocesses (subproducts, subresources) and the relations between the subprocesses (subproducts, subresources). The opposite of a refined process (refined product, refined resource) is an *elementary process* (*elementary product*, *elementary resource*).

A *representation schema* is a set of constructs and rules. The rules define how to get an explicit representation of information. The schema defines properties of all representations.

Graphical representations (short: *diagrams*) are constructed using a specific representation schema. Diagrams represent information by using a defined set of graphical symbols and text fragments.

1. Nonterminals are marked by surrounding angle brackets (‘<‘ and ‘>’; e.g., <exports>).

3 Assumptions and Principles

MVP—L is a prototype language, and we expect modifications of MVP—L in reaction to feedback from future projects. Moreover we cannot currently determine what elements (i.e., MVP—L constructs) a graphical MVP—L representation should incorporate. For these reasons, this report defines an initial set of building blocks (i.e., graphical symbols and text fragments).

Representation schemes are used as filters, and hence diagrams are often subsets of the textual information. The use of representation schemes may reduce the amount of information presented to users. Different types of diagrams have their individual advantages and disadvantages, depending on how they are used. Specific diagrams are derived with respect to specific requirements. This is the reason for avoiding a definition of graphical representations that are *complete* with respect to the textual information. The purpose of this report is to define single graphical symbols and text fragments. There are only a few rules provided which describe how to combine the building blocks. One is free to specify additional rules for selecting and arranging graphical representations.

No rules are provided which enforce graphical representations that are pleasing to the eye in their layout. We must gather experience with using these graphical symbols and text fragments before defining such rules.

Nevertheless, the information represented in a diagram must be expressible in MVP—L. We do not require that a diagram be transformable into a complete and consistent MVP—L description with respect to the language report [1] (e.g., because information is missing). For example, resources may not be represented as part of the refinement of a product.

The graphical representation schema described in this report is designed for modeling purposes (including review by project members). We are convinced that other schemas may be meaningful for supporting automated project guidance — such as designed for the process-sensitive software engineering environment MVP-S [4]. Such schemas will, for instance, consider temporal aspects (e.g., state transitions of a process over a project's lifetime).

The definition of the graphical elements is oriented strictly towards the syntactical structure of MVP—L descriptions. A direct transformation is done for the basic MVP—L constructs. Some information may not be represented graphically by restricting the allowed set of MVP—L descriptions (e.g., no use of 'I' operator) and by providing no graphical symbols for some MVP—L constructs which are:

- <global_attribute_model>, <product_attribute_model>, <process_attribute_model>, and <resource_attribute_model>
- <imports>

- <comment>
- <context>
- <attribute_mappings>
- <selection>

These MVP—L constructs were not incorporated in any previous examples of graphical MVP—L representations. Therefore we do not see any need to consider them in the definitions made in this report.

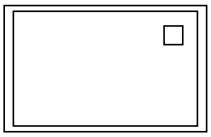
4 Representation

This section defines graphical symbols and text fragments of the representation schema.

4.1 Graphical Symbols

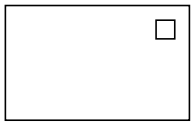
Project plans (<project_plan>) are represented by rectangles with dashed lines. The are similar to processes and are the root of a tree of process and product instances.

Elementary process models (<process_model>) are represented by double rectangles.

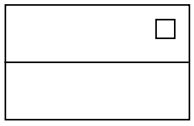


Refined process models (<process_model> with <refinement>) are represented like elementary process models, but are marked by an additional square in the upper right corner.

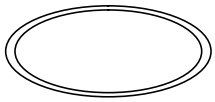
Elementary processes (declared in <objects>) are represented by rectangles.



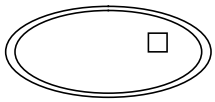
Refined processes are represented like elementary processes, but are marked by an additional square in the upper right corner.



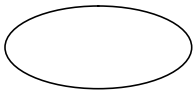
A symbol representing a process model or a process may have an additional horizontal line. Here a refined process is shown. The line separates two areas in which different sorts of text fragments are placed (see Section 4.2). Any <criteria> of the process model or process must be placed below the line.



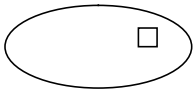
Elementary product models (<product_model>) are represented by double ellipses. A special case of an ellipse is a circle, but this has no additional semantics.



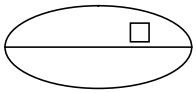
Refined product models (<product_model> with <refinement>) are represented like elementary product models, but are marked by an additional square in the upper right part of the symbol.



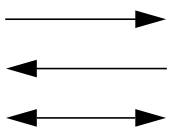
Elementary products (declared in <objects> of processes or project plans) are represented by single ellipses. Again, circles do not have any special semantics.



Refined products are represented like elementary products, but are marked by an additional square in the upper right part of the symbol.



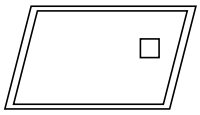
A symbol representing a product model or a product may have an additional horizontal line. Here a refined product is shown. The line separates two areas in which different sorts of text fragments are placed (see Section 4.2).



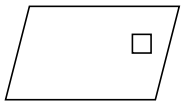
Product flow between processes is represented by arrows with filled heads. An arrow relates exactly one process and one product. An arrow pointing from a product to a process means ‘consume,’ and an arrow pointing from a process to a product means ‘produce.’ A double-headed arrow means ‘consume_produce.’
Product flows are specified either by refined processes (<interface_refinement> and <interface_relations>) - that means a process specifies the product flow between its subprocesses - or by project plans (<plan_object_rel>).
Arrows may be marked by names of local formal parameters of the process, which are defined in the product flow clause (see Section 5).

Elementary resource models (<resource_model>) are represented by non-rectangular[2] parallelograms. The direction of the slant has no meaning.

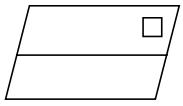
Elementary resources are represented by non-rectangular[2] parallelograms. The declaration of the referenced objects is specified in <objects> of processes (<process_resources>). The direction of the slant has no meaning.



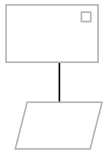
Refined resource models (<resource_model> with <refinement>) are represented like elementary resource models, but are marked by a square in the upper right corner.



Refined resources are represented like elementary resources, but are marked by a square in the upper right corner.



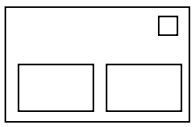
A symbol representing a resource model or a resource may have an additional horizontal line. Here a refined resource is shown. The line separates two areas in which different sorts of text fragments are placed (see Section 4.2).

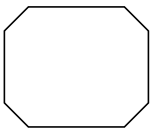


Resources and processes are related by a single line. Here the relation between a refined process and a resource is shown.

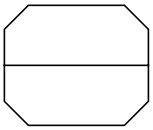
2. Small angles of inclination should be avoided to be able to distinguish between resource models and process models.

Refinements of processes, products, or resources can be expressed by arrows with a non-filled head. The same symbol may be used to represent specification of refinement in process models, product models, or resource models. The arrow always points from the aggregating object or model, which specifies the refinement, to an object of the next lower level of abstraction.
Refinements are defined in aggregating models (<structure_rel>), project plans (<plan_object_relations>) or processes (<process_object_relations>).
Another choice of representing refinements is the inclusion of symbols representing objects of lower levels of abstraction (on the left shown for processes). This style is not restricted to one level. A diagram may display many levels of abstraction at one time. The square must be placed in the upper right part.





Entry criteria, invariants and exit criteria (<criteria>) are represented by octagons. Such octagons are displayed within rectangles representing process models or processes. If present, all three octagons of a process model or process are shown (one for <entry_criteria>, <invariant>, and <exit_criteria>). All octagons of a process model or process are aligned horizontally. The sequence from the left to the right is important (left: <entry_criteria>, middle: <invariant>, right: <exit_criteria>). The line separates two areas in which different sorts of text fragments are placed (see Section 4.2).



Multiple instantiations of a process model, product model, or resource model using the *exp* operator are represented by shaded objects as shown in Figure 1.

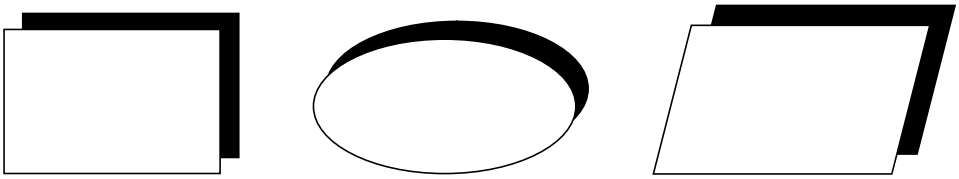
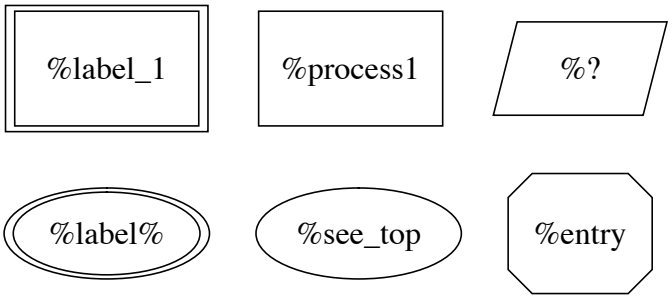


Fig. 1: Representing multiple instantiations

4.2 Text Fragments

<id> : <Model-Id> <id>.<id> : <Model-Id>	Identifiers and types of process instances, product instances, or resource instances (<object_decl>) appear textually within the representation of the object they reference. **Example:** requirements_doc.a_req : Requirement
<Model-Id> <M-Id> (<val> : <type>)	Identifiers (<ident>) of process models, product models, and resource models appear textually within the representation of the model they reference. Instantiation parameters (<instantiation_parameters>) follow the model name like in the textual representation. **Example:** Requirement (default_priority : Priority)
<id> : <Model-Id> <id> : <M-Id> = <InstParam> <id> : <M-Id> := <Value>	Attributes of processes, products, or resources are represented textually as shown on the left. In contrast to the language report, instantiation values are preceded by an equals sign (i.e., '=') instead by an assignment symbol (i.e., ':='). This is to distinguish between instantiation values and states during enactment. **Example:** prio : Priority = default_priority prio := Priority := 'low'
<criteria_expression>	<entry_criteria>, <invariant>, and <exit_criteria> of processes are taken into diagrams directly from their textual MVP—L representation. **Example:** a_req.status = 'complete' and req.prio != 'low'

Labels are used within diagrams to avoid forcing long text strings into the small symbols. Labels have to be unique within a single diagram. The text referenced by the label has to appear directly below the diagram. Any characters may appear in a label, but labels must start with a '%', which is not a valid character within an MVP—L symbol. Labels are especially useful for referencing <entry_criteria>, <invariant>, and <exit_criteria>. Mixing MVP—L text and a label (e.g., '%left_part = 50') is not allowed. Labels and the text they stand for are separated by a colon. Examples of using labels are shown in Figure 2.



%label_1 : ProcessModel
%process1 : process : ProcessModel
%? : resource : ResourceModel
%label% : ProductModel
%entry : req.status = 'complete'
%see_top : product : Product

Fig. 2: Examples for using labels within diagrams

5 Relating Graphical Symbols and Text Fragments

Models and objects are identified using a unique name (i.e., identifier). Multiple graphical symbols as representations of the same model or the same object are not allowed on the same diagram. Therefore, the relation between an identifier and the representation of a model or object is known.

Arranging the first three text fragments of a model or instance, namely the object name with model identifier, model identifier with instantiation parameters, and attribute information within the corresponding symbols is shown in Figure 3 for process model and process representations.

Fig. 3: Arrangement of names and attributes within process model and process representations

The identifiers of objects and models always appear in the upper left corner and above the line of object and model representations. Attribute information is always shown beneath the horizontal line. It should be noted that all instances of a multiple instantiation are identified by the same name, as shown by using placeholder in Figure 4.

Fig. 4: Use of identifiers when instantiating multiply (Example processes and products)

A horizontal line within an octagon (representing <entry_criteria>, <invariant>, and <exit_criteria>) separates local criteria (shown above the line) from global criteria (shown below the line). If no line appears, the information is taken as local criteria (see Figure 5).

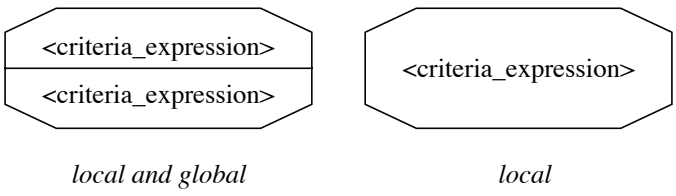


Fig. 5: Appearance of <criteria>

Objects can be identified by different names, depending on the context in which they appear. Because the scopes in which identifiers may appear do not overlap in MVP—L, an instance is referenced exactly once in each scope. For example, a product may be referenced in the aggregating product and in each process which has access to that product. All names identifying the object can be different. Through use of <interface_relations> and <interface_refinement> multiple references to the object are established. These multiple object references lead to problems in graphical representations because multiple MVP—L scopes appear on a single diagram. In particular, a rule is needed to describe which identifier to use in a symbol's upper left corner. We take the viewpoint that each diagram represents part of a project plan or refinement (process, product or resource). Within a symbol (rectangle, ellipse, or parallelogram with single lines) the name of the project plan or aggregating model should be used. A model's local name has to appear on the outside, close to the relating symbol (arrow or line). An example is given in Figure 6.

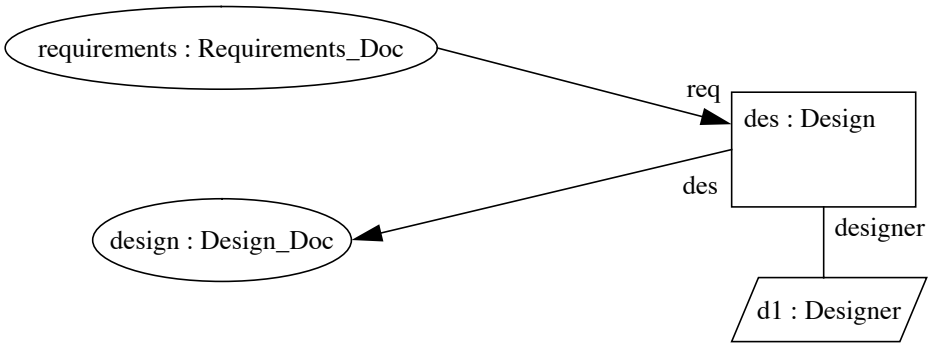


Fig. 6: Diagram of the project plan "sample"

An excerpt of the project plan for the diagram in Figure 6 is given in the following:

```
project_plan sample
imports
    product_model Requirements_Doc, Design_Doc;
    resource_model Designer;
    process_model Design;
objects
    requirements : Requirements_Doc;
    design : Design_Doc;
    d1 : Designer;
    des : Design;
...
end project_plan sample

process_model Design () is
...
product_flow
    consume
        req : Requirements_Doc;
    produce
        des : Design_Doc;
```

```
...
resource_allocation
    personnel_resources
...
        designer : Designer;
...
end process_model Design
```

It should be noted that the name *des* appears twice in the diagram (Figure 6) identifying different objects (a product and a process). This homonym should be avoided, despite the fact that it is allowed in MVP—L. The reason for the appearance of the same name for different objects is, as said before, that scopes are not separated in the graphical representation, whereas they are separated in the textual representation.

Former projects have demonstrated the usefulness of having symbols that represent products appear within process model representations (similar to project plans). An example is given in Figure 7. Forcing the placement of the product symbols outside the process models would result in complex and difficult to understand representations (with respect to product flow).

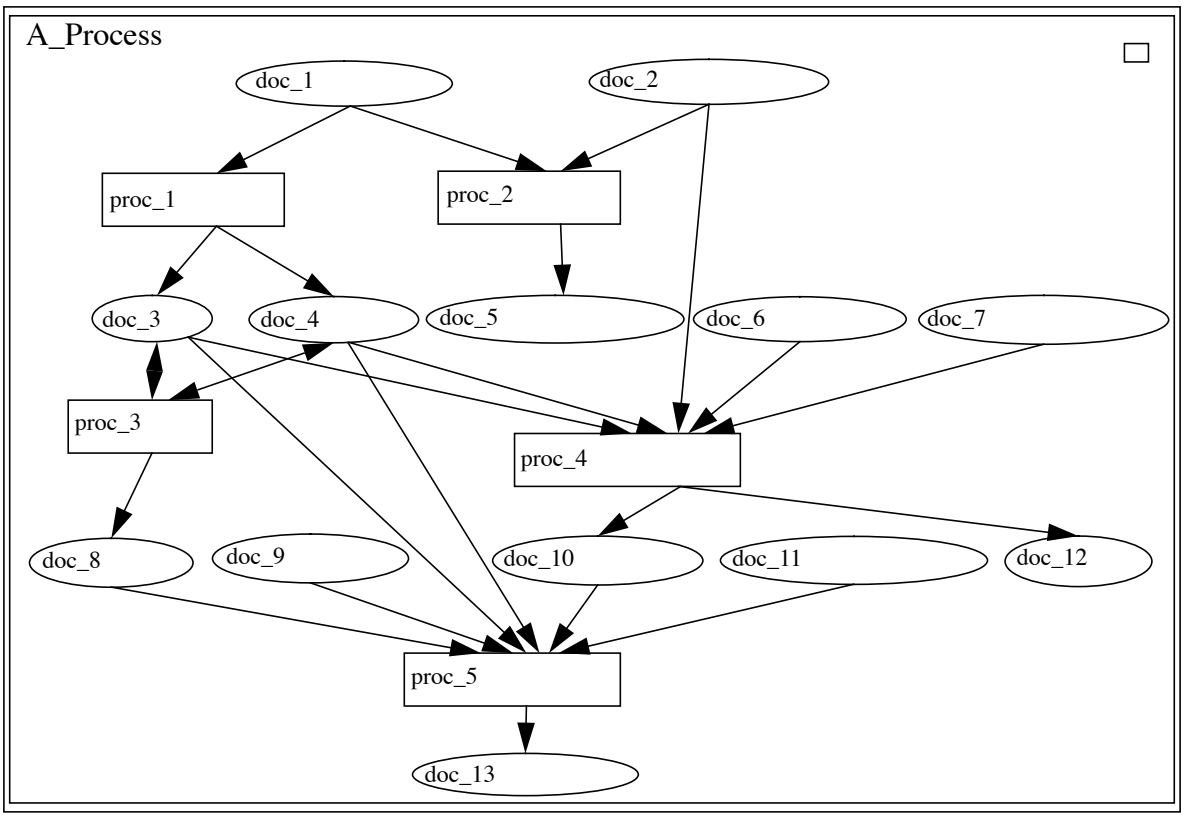


Fig. 7: Using product symbols within process model representations

6 Examples

This section gives a few sample diagrams. The MVP—L description, from which the examples are taken can be found as an appendix to the language report [1]. The following example representations are tailored to specific modeling and review purposes. That means, the schema defined in this report was used only partially. Some information for which symbols exist are not represented. For each diagram the reason why it was chosen is explained briefly. This makes the point that a representation cannot be considered useful without knowing the context it is used in.

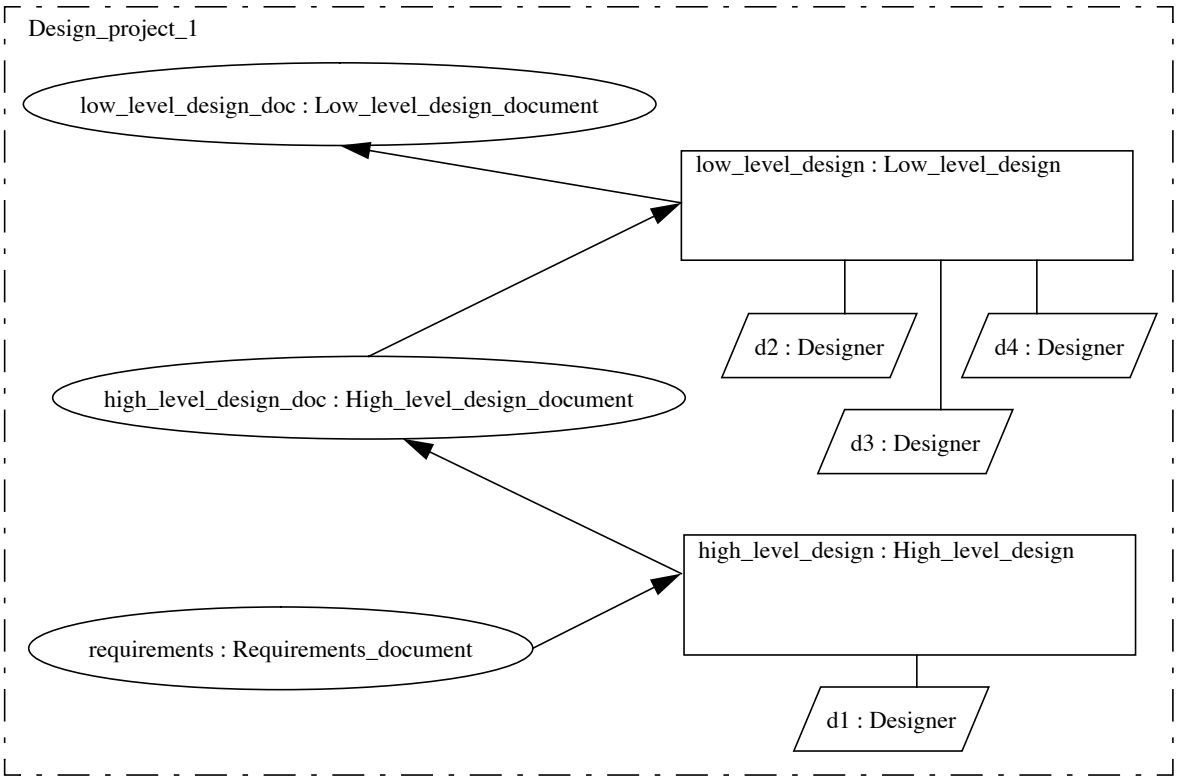


Fig. 8: Project plan Design_project_1 ([1], page 78)

The diagram shown in Figure 8 can be used to specify project plans. All objects of the <objects> clause are represented and their relations are defined. Some textual information is missing, for example local names of the objects are not represented because they are not necessary (the same information is represented by arrows used to specify product flow).

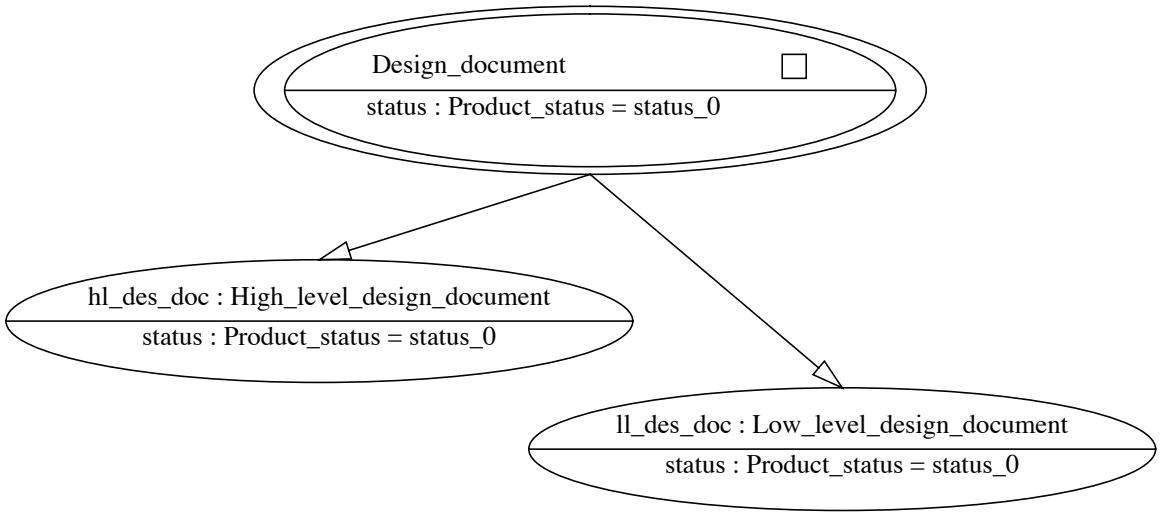
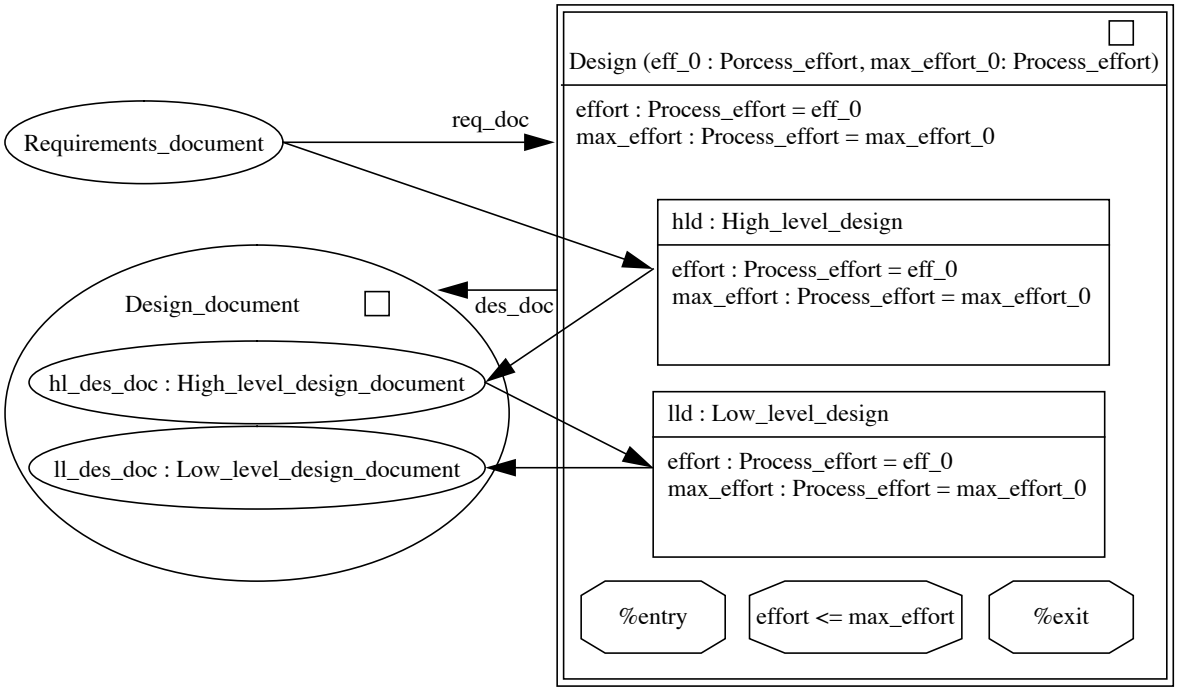


Fig. 9: Product hierarchy Design_document ([1], page 84)

Figure 9 gives an example of a product refinement. The diagram considers only subproducts and attributes. This subset of the schema defined in this report is used when a product model is specified. The diagram represents the refinement only partially; for instance <attribute_mappings> is missing.



%entry : (req_doc.status ‘complete’) and (des_doc.status = ‘non_existent’ or des_doc.status = ‘incomplete’)

%exit : des_doc.status = ‘complete’

Fig. 10: Process model Design ([1], page 85 and 86)

The example shown in Figure 10 represents process model information. The diagram is suitable for supporting understanding what the goal of each process of type *Design* is (i.e., the criteria in the lower part of the model representation) and how this goal is achieved (refinement into subprocesses and product flow between them). The local names for the products are given only for the aggregating model. It is important to note that two arrows point from the product of type *Requirements_document*. One arrow relates the product with the process model *Design*, the other relates the product with *Design*'s subprocess *hld*. Another important fact is the specification of product flow on both levels of abstraction (between *Design_document* and *Design* on the one hand, and between *High_level_design_document*, *Low_level_design_document, High_level_design*, and *Low_level_design* on the other hand). Although on the higher abstraction level only the relationship 'produce' is specified, the product flow relationship 'consume' is used on the lower level of abstraction. The first enacted subprocess (*hld*) produces a document is consumed from subprocess *lld* but this fact is hidden from the outside.

7 Summary and Future Work

Applications of the software process modeling language MVP—L identified the need for graphical representation of the models, objects, and project plans. This report defined a graphical representation schema that is limited to the elementary graphical symbols and text fragments which directly correspond to MVP—L constructs. Only a few rules were given for how these building blocks may be related to each other. Some examples illustrated the application of the defined representation schema for different modeling purposes.

The graphical representation schema should be understood as a basic set which can be extended or reduced when used for a specific task. We are convinced that for different modeling tasks different representation schemes will be considered suitable. Consequently, the definitions in this report build a foundation for the following challenges:

- Development of a set of standard representation schemes and definition of rules for transforming textual representations into graphical ones and vice versa.
- Development of tools to present and manipulate information according to the definitions of the standard representation schemes.
- Definition of representation schemes for supporting project enactment. Of course, these kind of representations comprise more information than the one defined in this report (e.g., temporal aspects, quality models).

The definitions of this report are an extension to the MVP—L language report [1] and will evolve together with the language.

References

[1] A. Bröckers, C. M. Lott, H. D. Rombach, and M. Verlage. MVP–L Language Report Version 2. Technical Report 265/95, Department of Computer Science, University of Kaiserslautern, 67653 Kaiserslautern, Germany, February 1995.

[2] C. D. Klingler, M. Neviaser, A. Marmor-Squires, C. M. Lott, and H. D. Rombach. A case study in process representation using MVP–L. In *Proceedings of the 7th Annual Conference on Computer Assurance (COMPASS 92)*, pages 137–146, June 1992.

[3] B. Hoisl. A process model for planning GQM-based measurement. Technical Report STTI-94-06-E, Software-Technology-Transfer-Initiative Kaiserslautern, University of Kaiserslautern, 67653 Kaiserslautern, Germany, April 1994.

[4] C. M. Lott, B. Hoisl, and H. D. Rombach. The use of roles and measurement to enact project plans in MVP-S. In W. Schäfer, editor, *Proceedings of the 4th European Workshop on Software Process Technology*, pages 30–48, Noordwijkerhout, The Netherlands, April 1995. Lecture Notes in Computer Science Nr. 913, Springer-Verlag.