6.4 Administration Models and Management Tools

R. Hai, T. Heer, M. Heller, M. Nagl, R. Schneider, B. Westfechtel, and R. Wörzberger

Abstract. One of the vertical columns in the overall process/product model deals with the cooperation of subprojects I1 and B4. Both study the support for reactive process management in dynamic development processes. In this section we highlight the transition from application models developed in subproject I1 to tool models for reactive management of subproject B4. We summarize our findings w.r.t. the development of management tool models as well as their connections to application models. The section focuses on a process-oriented viewpoint. Products and resources of development processes can be discussed analogously. We identify the missing parts which need to be further investigated in order to get a comprehensive and integrated process/product model, here for reactive management.

6.4.1 Introduction

In this section, we describe the transition *from application models to executable tools*. The vertical column of the PPM from subproject I1 to subproject B4 is regarded, dealing with reactive process management, see Fig. 6.1. The application models capture process-related aspects of work processes within development processes in subproject I1, and subproject B4 has developed executable tool models to derive *process management* tools supporting development process managers and process engineers. So, this section is to describe and evaluate the *contribution* of both subprojects to the dominating problem of developing a PPM.

The section is *structured* as follows: First, we summarize the application models of subproject I1 and the executable tool models of subproject B4 in Subsect. 6.4.2. The main part of this section deals with describing the transition from application models to executable tool models (Subsect. 6.4.3), and its relation to the formal process/product model (Subsect. 6.4.4). We finish the section by giving open problems and a conclusion.

6.4.2 Application Models and Tool Models

In this subsection we describe the *models* developed within the *subprojects* I1 and B4. The integration of these models is addressed in the next subsection.

Summary of Application Models

An *overview* over the *application models* and the tool builder's models relevant for development process *management* is given in Fig. 6.7. In the upper part of the figure, the IMPROVE application domain models in chemical engineering are shown:

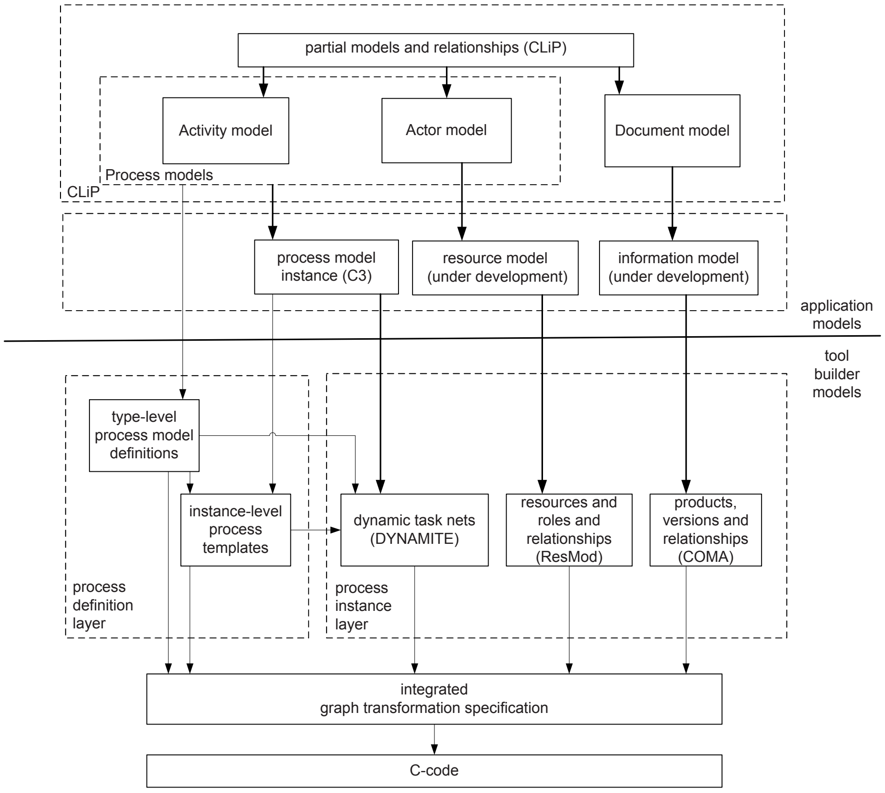
partial models and relationships (CLiP)
Activity model
Actor model
Document model
Process models
CLiP
process model instance (C3)
resource model (under development)
information model (under development)
application models
tool builder models
type-level process model definitions
instance-level process templates
dynamic task nets (DYNAMITE)
resources and roles and relationships (ResMod)
products, versions and relationships (COMA)
process definition layer
process instance layer
integrated graph transformation specification
C-code

Fig. 6.7. Application and tool models for management

1. The object-oriented data model *CLiP (Conceptual Lifecycle Process Model)* [14, 19] for product data of the design process and the corresponding work process, as described in Sects. 2.2 and 2.4, defines partial models structuring the engineering domain into several working areas. The relationships between the partial models are also contained in CLiP. For instance, there is a partial model `Process Models` (details below). Within `Process Models`, the model `Activity` and the model `Actor` are connected by the relationship `skill`.
   In this section, we mainly *focus* on *process-related modeling* and therefore we highlight the partial model `Process Models` (cf. Chap. 2). This submodel defines several modeling concepts to express important structures for work processes like `activity class`, `input and output information`, `tool`, `goal` and their relationships. For example, the `Simulation reactor` activity creates the information `Reactor simulation result`. This information is defined in the partial model `Document model`, which contains the information-related modeling concepts. The resource-related modeling concepts can be found in the CLiP partial model `Actor model`.

Roles are only implicitly defined, as a skill-oriented approach is used in order to define the work capabilities required for a certain role. For example, an activity `simulation reactor` may need actors with a skill `Reaction kinetic knowledge`.

2. Work processes on the *instance level* are modeled by the C3 process modeling notation [116]. On the instance level, complete work processes can be modeled and analyzed, e.g., to identify potentials as to shorten development time. For example, all important aspects of the Polyamide-6 case study used in IMPROVE have been modeled using the C3 process modeling language (cf. Sect. 1.2). For this purpose, C3 contains elements to model process-, information-, and resource-related aspects of work processes. The basic elements of C3 are `activity`, `role`, and `information` with relationships like `control flow` and `information flow`. Currently, the clear focus is on the process side while the information models and role models only cover basic aspects and are currently under development.

Summary of Tool Models

The *tool models* are displayed in the lower part of Fig. 6.7. While the semi-formal application models describe the application domain, the tool models have to be formal in order to be used for building process management tools.

1. In the center the process model for dynamic *task* nets (DYNAMITE), the *product* model (COMA), and the *resource* model (ResMod) are shown. These models are tightly integrated with each other and form an integrated management model allowing to describe development processes on type level (knowledge) as well as on instance level (concrete projects or templates).
   For *example*, a *dynamic task net* resembles a running development process with all tasks, resources, and all documents which are created during the process. Elements of dynamic task nets are `task`, `input parameter`, `output parameter`, `control` or `feedback flow`, `data flow`, etc.
   As stated above, in order to be able to build tools, the *execution semantics* have to be formally specified. The models are generic in the sense that no application-specific information is contained. They form the core of the process management system AHEAD which supports the interleaved planning and execution of development processes.
2. The generic models of AHEAD can easily be *adapted* to an application *domain* or to specific project *constraints* by defining standard activity types, standard workflows, and workflow templates. We have used the wide spread modeling language UML to model such constraints and parametrization aspects in an object-oriented way on the class level.
   Class diagrams and *object* diagrams are used to create process model definitions or process templates, respectively. For example, standard types for activities can be defined by introducing new classes like `Design Reactor`

or `Simulate Reactor` within in a class diagram together with relationships between classes, like `sequential control flow`.
3. The three generic models DYNAMITE, COMA, and ResMod are formalized in *graph transformation* specifications. Specific adaptations of these models defined in class diagrams and object diagrams are transformed to specific graph grammar specifications.
4. Both *generic* and *specific* graph *specifications* are *combined* and executable C code is generated from them. This C-code is embedded into a tool building framework. In our case, the configurable user interface and the core application logic of AHEAD can be generated semi-automatically as explained in detail in Sect. 3.4.

The *transition* between *application* models to executable *tool* models cannot be easily realized. For example, some information which is necessary in a tool model might not be explicitly modeled on the application layer, as different aspects are targeted at both layers. In the following section, we describe how all necessary information is obtained by either adding additional information or by deriving the information from existing application models.

6.4.3 From Application Models to Tools

In order to offer a domain-specific support system for the management of development processes, the information of all *tool models* is needed to build the AHEAD system (subproject B4). *Additionally*, the *application* models developed in subproject I1 are exploited to provide the missing context information, needed to adapt the AHEAD system to a specific context [154].

Instance Level Application Models

On the instance level, dynamic task nets, products, and resources used within a specific development process are modeled. On the application model side, similar process information is foremost contained in C3 nets. Application modeling experts create *process templates* in the form of C3 nets to define best-practice work processes. These C3 nets can be transferred structurally into *dynamic task nets*. Currently, a set of structural restrictions has to apply for the C3 nets used. Dynamic task nets can be generated on the tool side as the surrogates of the process templates modeled as C3 nets. We have realized an *integrator* for the mapping of C3 nets into dynamic task nets (see Sect. 3.2).

Product-related *information* can also be contained in C3 nets and be transferred into dynamic task nets or the product model of AHEAD, respectively. Currently, we do not extract these data with the integrator. In the C3 net, it can be captured that input or output documents of activities require a specific document type. *Resource*-related information like actors or actor roles required to perform an activity are, however, carried over into a dynamic task net.

Class Level Tool Modeling: Connection to Application Models

On the class level, we are dealing on the tool side with process model definitions and process templates to define structural and behavioral knowledge about processes. This information is located at the class or *type level*, e.g. task types, document types etc. and their relationships for a specific context can be defined.

The CLiP models and domain ontologies can be searched for *standard types* of work process activities or document types. The partial models usually contain such elements which have been identified to be of broader relevance to the respective application domain of the partial model. For example, the activity type `Design Reactor` might be contained in a UML class diagram according to the domain ontology of the partial model `Work Processes` where the activity concept is located. Essentially, the same or similar modeling concepts are used in CLiP and in the metamodel underlying the process model definitions and process templates of AHEAD. Currently, the information found in CLiP models has to be integrated manually into UML class diagrams used in the AHEAD approach. For example, for the activity type `Design Reactor` in CLiP, a new class `Design Reactor Task` is created in the class diagrams. In this case, the AHEAD system will allow for the instantiation of tasks with this specific type in dynamic task nets.

Relationships defined in CLiP models can be reflected in class diagrams by introducing new associations between classes. Currently, this is not possible in our approach, as only a limited set of default associations has been implemented to connect classes, like associations denoting control flow relationships or data flow relationships. We use UML stereotypes to define the type of an association link between two classes. Up to now, other relationships, e.g., those with a more semantical character, are neglected, although their integration is easily possible by simply using additional UML stereotypes as annotations for associations.

Application Modeling for Tool Adaptations

We use the broadly distributed UML notation for the explicit purpose that *application* modeling *experts* can create process model definitions and process templates for AHEAD. Nevertheless, application domain experts and *tool building experts* can work together to arrive at such models more quickly until the application expert has gained enough experience in adopting our specific use of the UML for process modeling purposes. Specific tasks, like introducing new unforeseen dependencies between classes or new requirements leading to necessary technical modifications of AHEAD, can be discussed in a short-circuit mode of cooperation. By the way, this approach of bringing experts of different domains closer together, was often followed in the IMPROVE project.

If process model definitions or process templates have been defined to introduce specific adaptations of the otherwise unrestricted generic AHEAD model, some further steps are needed. These steps are only necessary because of the specific tool generation approach followed in the B4 project, which is based on the semi-automatic generation of user interface prototypes from graph transformation specifications: (a) First, the tool builder uses a transformation tool to transform the UML class or object diagrams into graph transformation specifications, which contain the specific parametrization data for AHEAD. If no process model definitions are defined, AHEAD uses built-in default types for tasks and documents etc. (b) Second, the tool builder combines the newly generated specific graph specification with the generic graph transformation part containing the AHEAD management model. From this overall specification, executable C-Code is generated in an automatic step. (c) Finally, the tool builder has to integrate this C-code with the pre-configured user interfaces to form the overall AHEAD system.

6.4.4 Relation to the Overall Process/Product Model

In this subsection, we look at the application and tool models in order to present their *relations* to the layered overall *process/product model* of Fig. 6.1.

The overall process/product model is decomposed into five layers: application model layer, external model layer, internal model layer, mapping layer, and basic model layer. Going from top to bottom, each layer adds specific aspects which have not been covered at the layers above. We now discuss the *vertical column (d)* regarding reactive management (cf. Fig. 6.1).

The way from application models to tools for reactive management does not match smoothly with the idea of the overall process/product model developed so far. We are now going to identify which aspects of each layer are relevant for reactive management and highlight some open problems.

Layer 1 of the overall process/product model deals with all *application* domain *models* for the process and its products mentioned in this section. Among them, we can identify domain knowledge models to structure the application domain and organizational knowledge models which contain knowledge how processes are carried out in different subdomains or companies. For example, work processes modeled by application domain experts on a medium-grained level can be found on this layer. Similarly, medium-grained product models and resource models also belong to layer 1.

Layer 2 contains the *external models* of tools for different users. The user interface notations used for modeling different process or product aspects of tools should be found on this layer. Likewise, the representation of complex commands of the tools is located here, because they offer application-oriented functionalities for the user. Currently, we do not have such explicitly modeled external process or product models within IMPROVE. External models are indirectly introduced on the next layer.

On layer 3, *internal models* of tools are located. They are best represented by formal and executable models which are immediately usable to derive tools. All tool models belong to this layer, such as the AHEAD management model

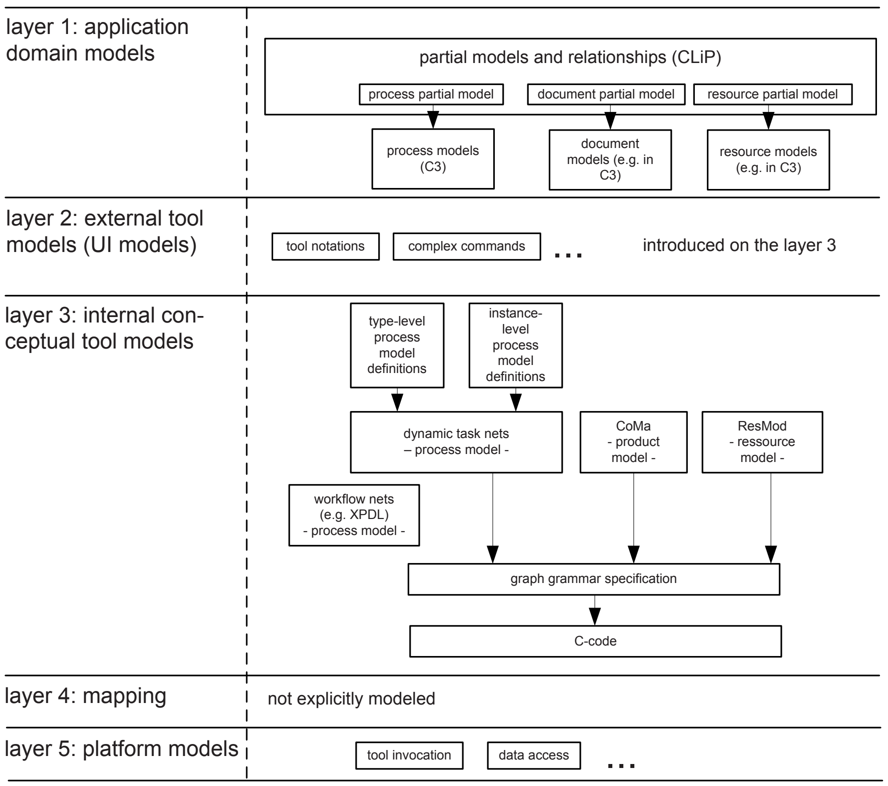
layer 1: application domain models
partial models and relationships (CLiP)
process partial model
document partial model
resource partial model
process models (C3)
document models (e.g. in C3)
resource models (e.g. in C3)
layer 2: external tool models (UI models)
tool notations
complex commands
...
introduced on the layer 3
layer 3: internal conceptual tool models
type-level process model definitions
instance-level process model definitions
dynamic task nets – process model -
CoMa - product model -
ResMod - ressource model -
workflow nets (e.g. XPDL) - process model -
graph grammar specification
C-code
layer 4: mapping
not explicitly modeled
layer 5: platform models
tool invocation
data access
...

Fig. 6.8. Application models and tools models

(DYNAMITE, COMA, and ResMod). These models have to ensure that the user interface and functionality described on layer 2 is fulfilled by the tools. To be more specific: UI details are found here, in particular there are specs for complex commands. How the UI is built is not specified but introduced in the tool construction process.

Finally, layers 4 and 5 contain models containing all *platform aspects* and a *mapping* from conceptual models to platforms. These models are only relevant in order to implement the tools independently of specific execution platforms. As previously mentioned, the realization of the complex tool components on the basis of operating system processes is handled on this layer. Currently, the tools for reactive management developed within IMPROVE are limited to specific operating systems and programming languages. However, AHEAD is coupled with other tools of IMPROVE using platform services such as access to document repositories.

6.4.5 Open Problems and Conclusions

In this section, we discussed for the vertical column (d) reactive management how the *transition* from *application models to tools* takes place. We focused on the process-side and explained, how application models and tool models are related to each other. Finally, we discussed the relation of achieved results to the process/product model.

Although we have acquired a good understanding of application models and tool models corresponding to management support on a medium-grained level, some *open problems* still remain:

1. It has to be investigated if some of the *application models* could be *extended* in order to match the tool models more closely, or vice versa. Currently, these model gaps are bridged manually and for specific cases only.
2. As we stated above, a tool-independent process/product model on the *external model* layer is still *missing.*
3. We have achieved some results regarding the transformation of application models to tool-specific models when specific tools are chosen [154]. For that purpose, we have already described how a framework for the definition, analysis, improvement, and management of inter-organizational design processes involving different modeling formalisms and heterogeneous workflow tools could look like. This framework combines models, methodologies, and tools from the IMPROVE project. Its key features are to bridge the gap between modeling and execution of inter-organizational design processes and the seamless execution support for dynamic and static parts of the overall process, both by appropriate process management systems. These *results* need to be *generalized.* For example, a methodology could be developed for the extraction of common modeling concepts and the harmonization of the process models into a single uniform model which could serve as a mediation model between specific process models. The same applies for product models.