Development of a Software Process Model for Multimedia CAL Systems by Applying Process Landscaping

Corina Kopka, Ursula Wellen
University of Dortmund
Chair for Software Technology
{kopka, wellen}@ls10.cs.uni-dortmund.de

Abstract

In software engineering a lot of different process models are established. They focus on different features or are adapted to different application domains. New roles in the development of multimedia computer-aided learning system (CAL) applications, hitherto not considered, require an additional process model. This paper presents an approach for the development of a software process model for multimedia CAL systems by applying the method of Process Landscaping. The resulting process landscape focuses on the different roles taking part in the underlying software process. Additionally, Process Landscaping provides a restructuring algorithm to achieve a more conventional view where activities are arranged by the order of their appearance within the software process. Its application is also discussed in this paper. There are mainly two advantages of providing both views of the same software process. The first is to employ the participating roles with a perspective of a process model they are familiar with but which also considers interfaces between the roles. The second is to provide a process model of the same software process which focuses on activities and their interrelations relevant for process management.

Keywords

computer-aided learning system, multimedia, process model, Process Landscaping, software process

1 Introduction

A development process "from the inception of an idea all the way to the delivery and final retirement of the system, is called a software production process" [GJM91]. In accordance with Ghezzi et al.'s definition we call the representing model a software production process model. In this paper, we use this term instead of the term *software process model* to distinguish between a model representing a software production process, and a model supporting the participating roles with guidelines and recommendations which is also called process model. The study of those software production process models shows an enormous variety of software development approaches. Main reasons are the evolution of different software life cycle approaches [Boe86, BP92, BD95], the distribution of development activities to different locations [NKF+95, Yan98], progress in the area of software technology [BRJ98, Bro96] and domain-specific requirements for different types of applications as discussed by Arndt [Arn99] and Calvez [Cal93].

During the recent past several multimedia CAL systems have been developed [ADK99, DEM+99, Wei00]. Initial experience shows that the underlying production processes involve additional roles like didactic, content and media experts, requiring the coordination of tasks arising from various disciplines. Currently, efforts are being made to adapt available software production processes from the domain of authoring systems [DEM+99]. But this approach seems to be insufficient because it is strongly customized to the application of programming tools. Another approach adapts software production processes used for the development of component-based software by generating learning modules as components for computer-based training systems [Wei00]. But in our opinion both approaches do not consider sufficiently the roles, their different responsibilities and their interrelations in the software production process.

The study of existing approaches towards a suitable software production process has resulted in the need to develop a production process by starting with special focus on the different roles instead of adapting existing processes. This process development requires a structured method which supports the identification of different roles, their appearance during the production process, and their interfaces and dependencies.

It is necessary to provide a view of the entire production process focussing on roles to obtain the acceptance of all participants. The domains they usually work in (e.g. didactics, media production, software development) are too different to provide an accepted process model where the focus is on the order of and dependencies between activities. But the latter perspective is useful for the process management because it supports analysis and control tasks. Therefore, we need an additional view of the same software production process which fulfils process management requirements without loosing information about associated roles.

In this paper we discuss the method of Process Landscaping [GW00a] as a suitable method for our purposes. Section 2 introduces the basic concepts of multimedia CAL systems and discuss software production processes for these systems. Section 3 explains the main ideas of Process Landscaping with focus on features relevant for the development of certain views of process models. In section 4, we apply the method by introducing first a role-based view for parts of the multimedia CAL production process (section 4.1) and by rearranging afterwards all activities according to the order of their appearance within the software process to fulfil process management requirements (section 4.2). Section 5 summarizes the results and offers a perspective of our future research.

2 Multimedia CAL Systems

In this section we discuss production processes for multimedia CAL systems and motivate the need of different views of the same entire production process. We discuss our approach of developing a software production process model which offers suitable views for the participating roles as well as for the process management. To understand the special features in the production process for this kind of software, basic concepts of multimedia CAL systems are introduced before.

2.1 Basic Concepts of Multimedia CAL Systems

Multimedia CAL systems are different from conventional software and have some special characteristics:

- For this kind of application the presentation structure depends on the didactical concept.
- The presented content in the CAL system also has to be considered.
- Because of the representation of domain content, multiple media also have to be integrated.

These characteristics of multimedia CAL systems have to be taken into consideration during the software production process. Nevertheless, a multimedia CAL system is also software and therefore software engineering methods should be applied.

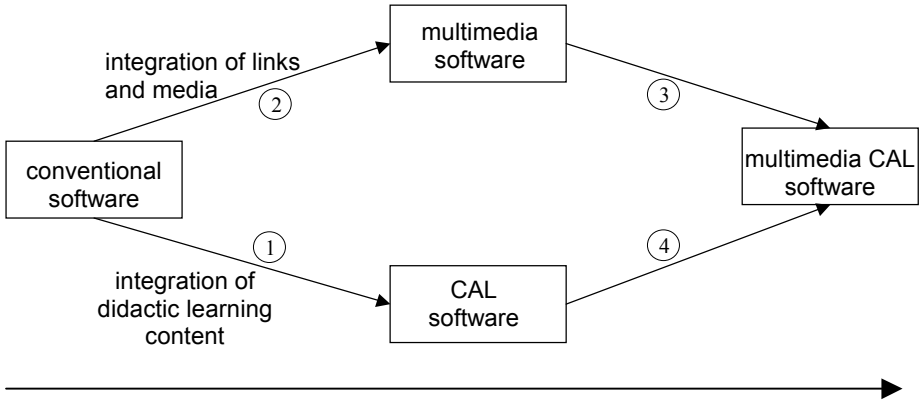

Figure 1: From conventional software to multimedia CAL systems

Figure 1 illustrates the evolution from conventional software to multimedia CAL systems along the time line, indicated by an arrow at the bottom of the figure. Conventional software systems such as information or management systems can be interpreted as representations for specific domain structures (e.g. hospital information systems). CAL systems differ from such conventional software systems since they prepare learning content didactically and integrate them [Bod90]. Didactical preparation, based on didactical concepts, means designing a virtual world with individual learning units. The virtual world is oriented on cognitive models and does not present the formal structure of learning contents. The evolution from conventional software to CAL software is illustrated in figure 1 by arrow 1. Multimedia systems, such as kiosk applications [Mül95] and product catalogues [SKT+97], derive from conventional systems by using media for visual representation of entities and links for representation of relations (see arrow 2 in figure 1). As indicated by arrows 3 and 4, multimedia CAL software integrate characteristic features of both, multimedia and CAL software. Summarizing all features, we describe multimedia CAL systems as a combination of multimedia and CAL systems.

2.2 Production Processes for Multimedia CAL Systems

Unfortunately, none of the underlying software production processes for multimedia and CAL systems fulfil our requirements concerning a production process for multimedia CAL systems: the Relationship Management Methodology (RMM) [ISB95], Hypertext Design Model (HDM) [GPS93] and Object-Oriented Hypermedia Design Model (OOHDM) [SR95] are development models and methods used for hypermedia or multimedia applications. They can only be applied to well-structured domain data. This is not the case for multimedia CAL systems. Didactical and content preparation implies different design for individual scenes. This leads to individual and partially or even unstructured domain data. Application or adaption of the models and methods mentioned above is therefore not suitable. Morris and Finkelstein [MF96] propose a discourse-driven design process model to guide engineers developing software and designers generating content. This design process model integrates media designers explicitly, but does not focus on didactical aspects. All methods focus on the software design process and do not support the entire software production process.

Software production process models for CAL systems are for example described in [Bod90] and [GH91]. Similar to conventional software development processes, they are linear, iterative, prototyping-based or evolutionary. Roles are defined and assigned only to some core development activities. A detailed description of production processes, however, which consider each role is missing:

- analysis and design activities focus on didactical and content work,
- implementation activities focus on implementation by programmers ignoring the necessity of additional analysis and design activities by a software analyst and designer.

The development of multimedia CAL systems requires an interdisciplinary approach because it intertwines activities from teaching, publishing and software development. This experience is based on observations during the Altenberg Cathedral Project [ADK99], where software developers and art historians worked together: Beside software developers, we had non-technical developers, such as didactic and content experts, media designers and producers, as additional roles in the production process. They had their individual way to work in their discipline, but all had to work together throughout the entire software production process. For example, in the analysis phase a target catalogue was developed from the didactic expert and further used by the content expert for content structuring into learning units and scenes.

As mentioned above, it is necessary to employ participating roles with a perspective of a process model they are familiar with and to obtain the acceptance of all participants. Because roles have their individual way of working we want to consider a production process for multimedia CAL systems modelled in a prescriptive [Poh95] and role-based way. In our opinion this approach is useful for guiding the work of each role within the entire production process by the division of work according to role specific tasks. Nevertheless, the roles have to work together. Therefore, another aspect within the production process to be considered is that interfaces between the role-based process models have to be identified, indicating where and with which information objects the related roles work with each other. Finally, the role-based process models and their interfaces have to be integrated in an entire production process model for multimedia CAL systems. In doing so, we obtain a software production process model providing an overview for role activities and a role-based view of the process.

For process management a descriptive production process model [Poh95] considering activities, their interrelations and their order is necessary. The purposes of process management are e.g. process documentation, analysis and process improvement and require another perspective of the same production process. This perspective has no focus on role specific aspects. It considers temporal and causal dependencies of the different activities. This view abstracts from knowledge about the different roles and allows us to compare the entire production process model for multimedia CAL systems with others. Therefore, we introduce an approach which considers both perspectives mentioned above, because in the production process for multimedia CAL systems we need both. We start with the development of role-based process models to provide process understanding and guidance for each role. To identify interfaces as particular points of cooperation between roles we consider role tasks for analysis, design, implementation and test. This classification originates from traditional software phases and is now applied to the other roles. Because the roles have to work together, it is helpful to detect activities where more than one role is involved, and cooperation points between roles. These also form the basis of improving cooperation between roles. Identification of these interfaces supports the integration of role-based process models into one process model. Because a process management perspective is also needed, we restructure the role-based perspective. This ensures to preserve all information already modelled in the role-based process models and emphasizes information required from process management.

The approach presented in this paper includes consideration of both perspectives needed in the production process of multimedia CAL systems and the facility of changing the perspectives. It considers roles and their dedicated activities, and integrates them in the entire software production process. This requires careful elicitation of interfaces between roles and their activities. Detailed and consistent refinement of interfaces between roles requires a formal base. The latter is also required to change the role-based view of our software production process into a more conventional view considering the temporal and causal dependencies of the different activities.

We can now formulate our approach as a three-step procedure:

1. Identification of roles and their area of responsibility (core tasks) in the production process and identification of interfaces between them.
2. Integration of the process models for each role into one process model by connection of interfaces between role-based process models and refinement of the core tasks by identification of activities for analysis, design, implementation, and test for each role.
3. Restructuring of the role-based view into a view considering the temporal and causal dependencies of the different activities.

We implement this three-step-procedure for the development of a software production process model for multimedia CAL systems by applying the Process Landscaping method, because it supports us with

- treating interfaces as first class entities, which means that they are already considered at the beginning of each modelling project;
- flexible refinement, which helps keep an overview of the entire process model;
- providing a formal basis, essential for consistent refinements of interfaces and for changing the role-based view of our software production process into a view supporting the process management;
- restructuring without loosing information.

Before we discuss our approach in more detail, we first explain the key features of the Process Landscaping method.

3 The Process Landscaping Method

Process Landscaping is a method which supports the modelling of a set of complex processes related to each other. Its purpose is to model processes to retain the overview of the entire process framework and to ensure the identification of all interfaces between the processes. The result of applying this method is called a process landscape. It represents processes and their interfaces, both on different levels of refinement. In this paper we do not present all details of the several method steps. We only

discuss features of interest for the development of a process landscape for multimedia CAL systems which have not been discussed sufficiently in other papers yet. For the sake of simplicity we do not explain these features along an example for multimedia CAL production processes but along a process landscape representing the well-known distributed development of component-based software applications. Process Landscaping has already been applied to several real-world processes, e.g. to the business of a telecommunications company [GW00a] and software development processes [GW00b].

Applying the method of Process Landscaping means at least executing the following steps:

- identification of core processes and interfaces
- refinement of both core processes and interfaces to (possibly) different extensions

A process landscape can be analyzed with respect to different properties, e.g. distribution and communication properties [GW01]. Petri nets [Rei86, Jen97] have proved to be a suitable formal basis for modelling and analyzing process landscapes. It enables the modeller not only to depict processes and interfaces on different levels of refinement, but also to define landscape properties simply as attributes assigned to activities, interfaces and relations. Such assignments are introduced as extensions of a "Petri net kernel" serving as the formal basis for a process landscape. In the following, we introduce extensions allowing us not only to define and assign locations where activities take place, but also to restructure the extended process landscape in order to consider locational distribution.

A process landscape is defined as a triple PL = (C, S, F), where C represents a set of activities, S a set of interfaces, and F a set of relations. With function *loc*: C $\rightarrow$ *P*(LOC), where *P*(LOC) represents a powerset of locations, we assign one or more locations to each activity in order to define where the different activities take place.

Each element l of the set LOC, which we call *local class*, is structured in the following way: l = (T, N) is a tuple, where T is a set of *local types* marked as either toplevel or sublevel [Poh00]. N is a set of *local names*. Local types group local names by their structure. For example, *city* is the local type for the local names Munich and London. Markings *toplevel* and *sublevel* indicate where process landscape elements are arranged, on the top level of the landscape or on a sublevel. The following example shows the procedure and a possible result of the assignment of locations to a process landscape. At first, we define the sets T and N. Their elements are listed in the table in figure 2.

local type T	**marking of T**	**local name N**
country	toplevel	Germany
country	toplevel	India
country	toplevel	Britain
city	sublevel	London
city	sublevel	Dortmund
city	sublevel	Munich
city	sublevel	New Delhi

Figure 2: Example for a local class

We distinguish between two local types *country*, marked as toplevel and *city*, marked as sublevel. For local type *country* we have defined three country names, and for *city* we have defined four different city names. Figure 3 indicates how the local types are assigned to different activities. Activities at this level of refinement are represented by rectangles, interfaces by circles. Activities with a bold frame indicate a higher complexity than activities represented by window symbols. *Domain Analysis* is an example for an activity which has already been refined further. We distinguish beween two types of interfaces, simple circles indicating interfaces between activities within a process model and circles with a vertical line showing interfaces between different process models.

Figure 3 shows parts of a component-based software development process, mainly activities for designing and implementing server and client software, and the domain analysis. It starts with requirements documents for client and server functionalities. The resultant software codes are forwarded to activities concerning domain analysis. Activities necessary for the development of client components are located in India and Germany, and activities concerning the development of server

components are only carried out in Germany. Domain analysis takes places in Britain and Germany, but for the latter case in another city than development activities.

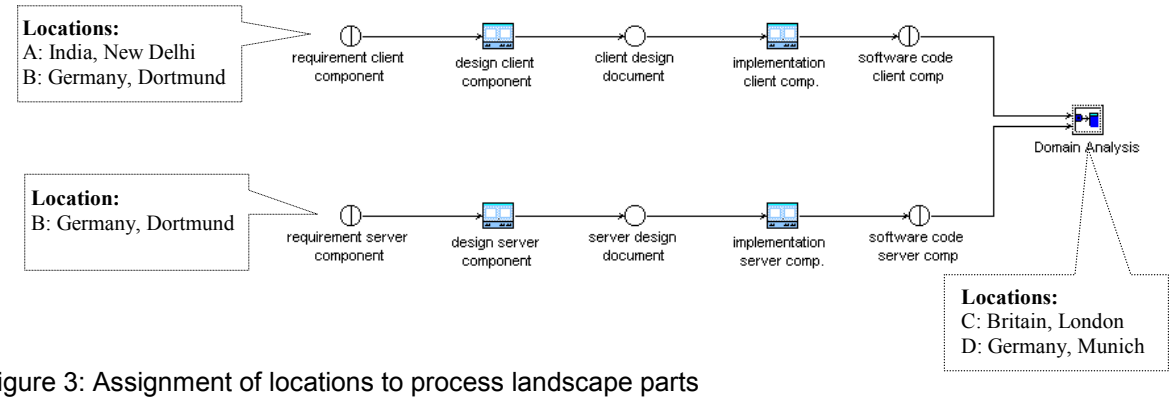

Figure 3: Assignment of locations to process landscape parts

Depending on the properties to be analyzed, different views of the landscape are required in order to emphasize different aspects. Note, that we use the term *view* in a special way: in [DKW98], Derniame et al. define a view as "the particular approach to a software process conveyed by a (sub-)model". They distinguish between models describing activities, organizational structures, products, resources and roles. This means different models representing different aspects of the same process landscape. In the context of Process Landscaping the term *view* is used for describing a certain perspective of always the entire process landscape, just by emphasizing different properties.

Process Landscaping distinguishes e.g. the logical and the locational view of a process landscape. When we use the term *logical*, we talk about causal and temporal logic of the activities' order. A locational view represents the distribution of activities among different locations. Figure 3 shows the logical view of the example where the process activities are modelled by their logical order. In this graphical representation it is not possible to see without additional notations where the different activities are taking place. This requires another view of the process landscape, the locational view.

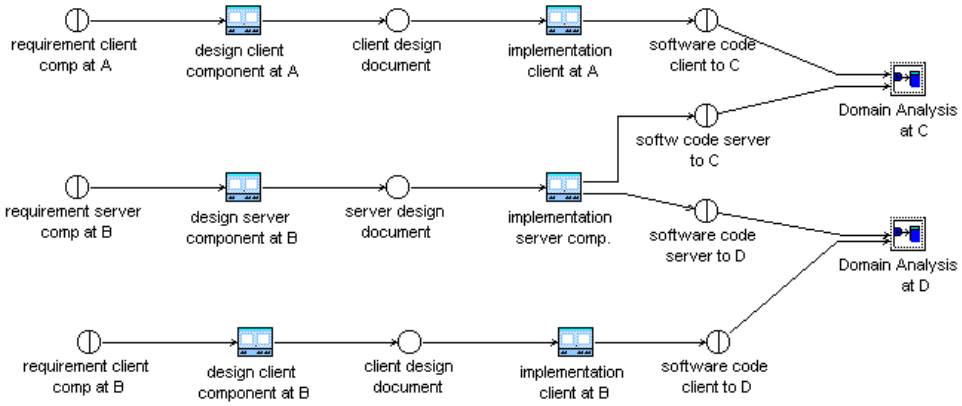

Figure 4: Locational view of a process landscape

Figure 4 shows the same parts of the software development process as depicted in figure 3. It emphasizes locational aspects of the process landscape. For the sake of simplicity, we represent the local names introduced in figure 4 as A, B, C and D. Now we can see that two client components have to be developed at two different locations A and B. One server component also has to be developed at location B. The software code of all three components is sent to two different locations C and D where domain analysis has to be carried out.

The process landscape in figure 4 is the result of a restructuring algorithm applied to the landscape in figure 3. The algorithm uses the location attribute to restructure a logical into a locational view. To obtain this view, the different locations have been assigned to all activities with function *loc*. For example, *loc*(design client component) = {(India, New Delhi), (Germany, Dortmund)} and *loc*(design server component) = {(Germany, Dortmund)}. No information of the logical view is lost, but distribution information has been emphasized. Now it is possible to analyze distribution properties in a more comfortable and transparent way.

In this paper, we do not discuss the formalism of the restructuring algorithm. We just want to make obvious the necessity of different views of one process landscape by the example of logical and locational views. The definition of other types of views like role-based views is also possible. But we can also start the modelling of such a role-based view on a process landscape and restructure it to another view, if required. For this purpose we have to adapt the idea of a class of toplevel and sublevel types and affiliated names to a given view by introducing a suitable "logical class". The restructuring algorithm uses this logical class to develop the logical view. The implementation of this idea is presented in section 4.2. Its motivation has already been discussed in sections 1 and 2.

4 Developing the Software Production Process Model for Multimedia CAL Systems

Until now, we have introduced the characteristics of multimedia CAL systems and our approach to obtain a suitable software production process model in section 2. In this section, we apply the Process Landscaping method presented in section 3 to develop a role-based process landscape representing parts of a software production process for multimedia CAL systems. Furthermore, we restructure this landscape to obtain the more conventional logical view without loosing information about the participating roles.

At first, we introduce in more detail two specific roles within a software production process for multimedia CAL systems. Because of the high complexity of this production process we restrict ourselves in this paper to these two roles and their refined process models. They form the basis for the application of the Process Landscaping method in this paper:

- **Didactic expert.** This role is important for multimedia CAL systems especially for selection and application of pedagogical principles according to suitable learning theories. Additionally, the didactic expert decides about the usage of certain types of media.

- **Content expert.** The main tasks of the content expert are the provision and preparation of contents. This is different to approaches for authoring systems where additional tasks such as didactical conception and parts of media design [GH91], or content preparation (e.g. storyboards) are carried out [SS96]. Both are in the responsibility of an author.

4.1 Development of a Role-based View of a Software Process Landscape

To obtain the role-based view of our process landscape we now apply the three-step procedure discussed in section 2. The first step is the identification of roles, their area of responsibility (core tasks) and the identification of interfaces. Applying Process Landscaping to this step means the identification of core processes, called clusters, and interfaces.

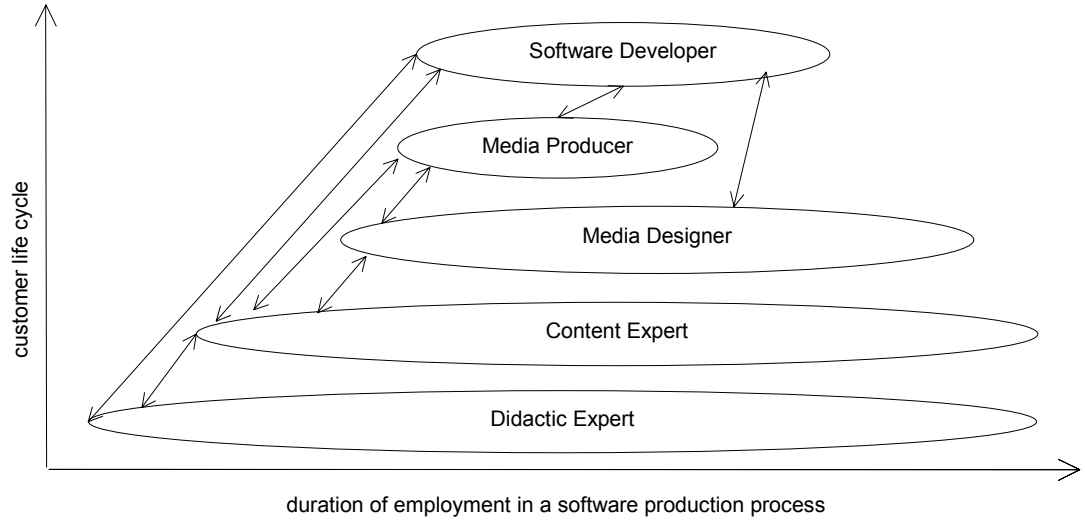

Figure 5: Top level of a process landscape for multimedia CAL system development

Figure 5 shows a process landscape for multimedia CAL system development indicating the most important roles. This representation abstracts from project and quality management and their affiliated roles. It focuses on roles for development activities, modelled on a top level as clusters. These are the roles of the didactic expert, the content expert, the media designer, the media producer and the software developer. Interfaces between the role-based clusters are indicated by bidirectional arrows (see figure 5). At this level, process landscape elements are arranged according to the temporal order in which they may appear first within a customer life cycle (Y-axis) and the duration of their employment (X-axis). A customer life cycle starts when a customer requires a multimedia CAL system and ends when the software production process has been finished.

The didactic expert starts the production process with analysis and design activities. The content expert continues by using didactic documents. Therefore, in figure 5 the ellipse depicting cluster *Didactic Expert* starts in the bottom left corner of the system of coordinates. Within the system of coordinates, cluster *Content Expert* is arranged more to the right, because the affiliated activities are carried out later in the customer life cycle. In the further proceeding of the production process specifications and guidelines from didactic and content experts are used by media designers, media producers and software developers. In doing so, knowledge is transferred from some roles via the indicated interfaces to others in order to extend or specify it.

In our three-step approach a more detailed description of interfaces is possible, but not required as second step (according to the method of Process Landscaping). Figure 6 illustrates the refined interface between clusters *Didactic Expert* and *Content Expert*. Refining an interface means to identify concrete information objects to be exchanged and to define the direction of information exchange. the interface illustrated in figure 6 contains *target catalogue, coarse didactical concept, employed media types, fine didactical concept, structural* and *interaction storyboards*. The structural storyboard contains sequences of learning units, structured learning paths and the definition of structural links. The interaction storyboard contains more detailed information about learning paths, depending on user answers and actions. The content expert completes both storyboards with content specific details or additional information.

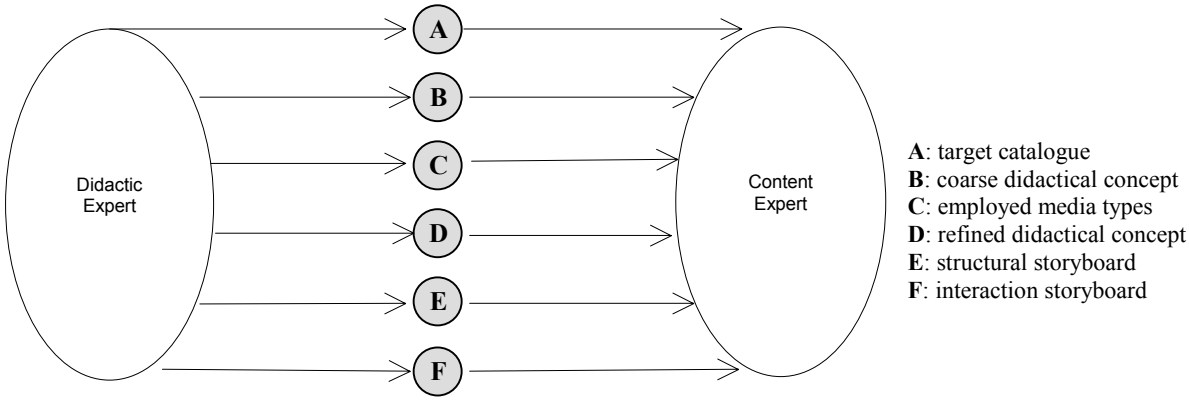

Figure 6: Interfaces between the clusters *didactic expert* and *content expert*

The direction of the data exchange has also been refined. For example, *content catalogue* is sent from the didactic expert to the content expert. Figure 6 does not show all information exchanged via this interface. The method of Process Landscaping allows us to extend each interface whenever one has identified further information objects. Summarizing the execution of an interface refinement, we have refined the interface between clusters *Didactic Expert* and *Content Expert* by identifying at least six documents to be exchanged (indicated in figure 6 by letters A to F). All of them are sent from didactic expert's activities to content expert's activities.

The second step of our approach is the refinement of each top level cluster by modelling all activities belonging to the different role responsibilities. This has to be done for each role by identifying analysis, design and implementation activities as (complex) tasks. Applying Process Landscaping for this second step therefore means refinement of the role-based clusters.

Figure 7 shows the role-based view of the process landscape depicting the production process for multimedia CAL systems at different levels of refinement. The roles didactic expert and content expert are illustrated at the top level of the landscape as clusters together with their interface. These clusters are refined to two process models. The interface between them is also refined and corresponds to the integration step of our approach. The hierarchical relationship between the entire process landscape, the role-based clusters and their affiliated process models is indicated by white-headed arrows (see fig. 7). For the sake of simplicity, figure 7 illustrates only parts of the process models corresponding to the didactic and the content expert. The analysis and design activities of both are represented in the two process models. The structural and interaction storyboards, produced by the didactic expert and required by the content expert, are used later, during the implementation phases of both roles.

Target catalogue and *coarse didactical concept* are analysis results of the didactic expert's preparatory work. The didactical targets and the developed didactical structure have impact on the work of the content expert. He needs this information for his own analysis, namely the *structuring of content*, in order to define learning units and scenes. *Employed media types* are needed from the content expert to decide about the media used and their combination in scenes. *Refined didactical concept* forms the basis for further content design.

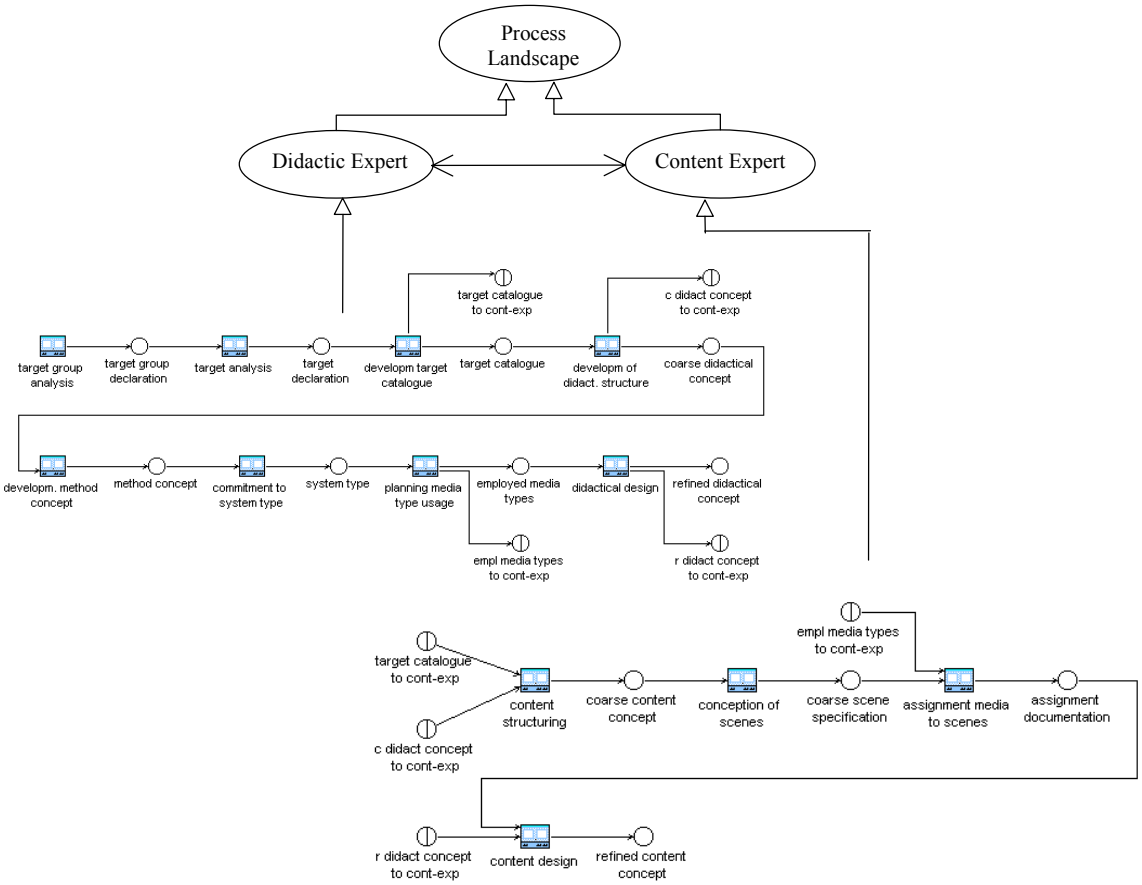

Figure 7: Role-based view of the software production process landscape

Until now, we have applied Process Landscaping to develop a role-based view of the process landscape representing parts of a software production process for multimedia CAL systems. This role-based view is useful for considering activities for each role together with their relations to each other. It serves as a guideline supporting the understanding of role-specific activities during the production process and the interrelations of the different roles.

In the following we restructure this landscape to obtain the more conventional logical view.

4.2 Restructuring the Process Landscape to Obtain the Logical View

We have to abstract from role aspects and have to emphasize logical dependencies to restructure the role-based view of a process landscape for multimedia CAL development to the more conventional logical view. For this purpose phases within the software production process, such as analysis, design, implementation and test phases, are of particular interest. For a logical view of the software production process each of the identified activities in the role-based view is assigned to one of the mentioned phases.

According to the introduction of a local class (as discussed in section 3) we now introduce a logical class to restructure a given process landscape. Figure 8 shows this logical class and illustrates the relation of logical types and activities of the example process landscape. In our example we have only one logical type marked as toplevel, but a further extension to logical sublevel types is possible. Analysis and design phases are represented by the logical names *Analysis* and *Design.* They are assigned to activities of the didactic and the content expert modelled as clusters in section 4.1. For example, for the content expert *content structuring* is assigned to *Analysis* and *conception of scenes* is assigned to *Design*.

activity	**logical type**	**marking**	**logical name**
target group analysis	core activity	toplevel	Analysis
target analysis	core activity	toplevel	Analysis
development target catalogue	core activity	toplevel	Analysis
development of didact. structure	core activity	toplevel	Analysis
content structuring	core activity	toplevel	Analysis
development method concept	core activity	toplevel	Design
commitment to system type	core activity	toplevel	Design
planning media type usage	core activity	toplevel	Design
didactical design	core activity	toplevel	Design
conception of scenes	core activity	toplevel	Design
assignment media to scenes	core activity	toplevel	Design
content design	core activity	toplevel	Design

Figure 8: Assignment of logical types and names to activities

Restructuring the process landscape leads to the logical view shown in figure 9. At the top level of the process landscape we now obtain the core activities *Analysis*, *Design* and *Implementation* represented as clusters. *Analysis* and *Design* have been refined further to two subprocesses, integrating the corresponding activities of the didactic expert and the content expert. For example, analysis activities of the didactic expert, namely *target group analysis, target analysis, development target catalogue* and *development of didactic structure* are correlated to activity *content structuring* for which the content expert is responsible. The resulting subprocess represents a logical view of analysis activities. The same integration step has been done for the design activities of both roles.

The interfaces between clusters of the role-based view of the process landscape (see fig. 7) and their affiliated documents are integrated in the new clusters of the logical view. The new interfaces consist of those documents which are produced during the analysis phase and required for the design phase. In the logical view the interface between clusters *Analysis* and *Design* contains at least the *coarse didactic concept* and the *coarse content concept* indicated in figure 9 by circles with a vertical line. They are represented twice, as output documents of cluster *Analysis* and as input documents for cluster *Design*. A restructuring algorithm identifies them in the role-based view by looking for places in the underlying Petri net fulfilling the following condition: pre- and post-transitions are taking place during different phases.

With the resulting logical view of the software production process we have finished step 3 of our approach (see section 2). It enables control and analysis tasks of the process manager at different levels of detail. It also enables the process management to compare this process model easily with others, which are often structured similar, namely according to their temporal and causal dependencies.

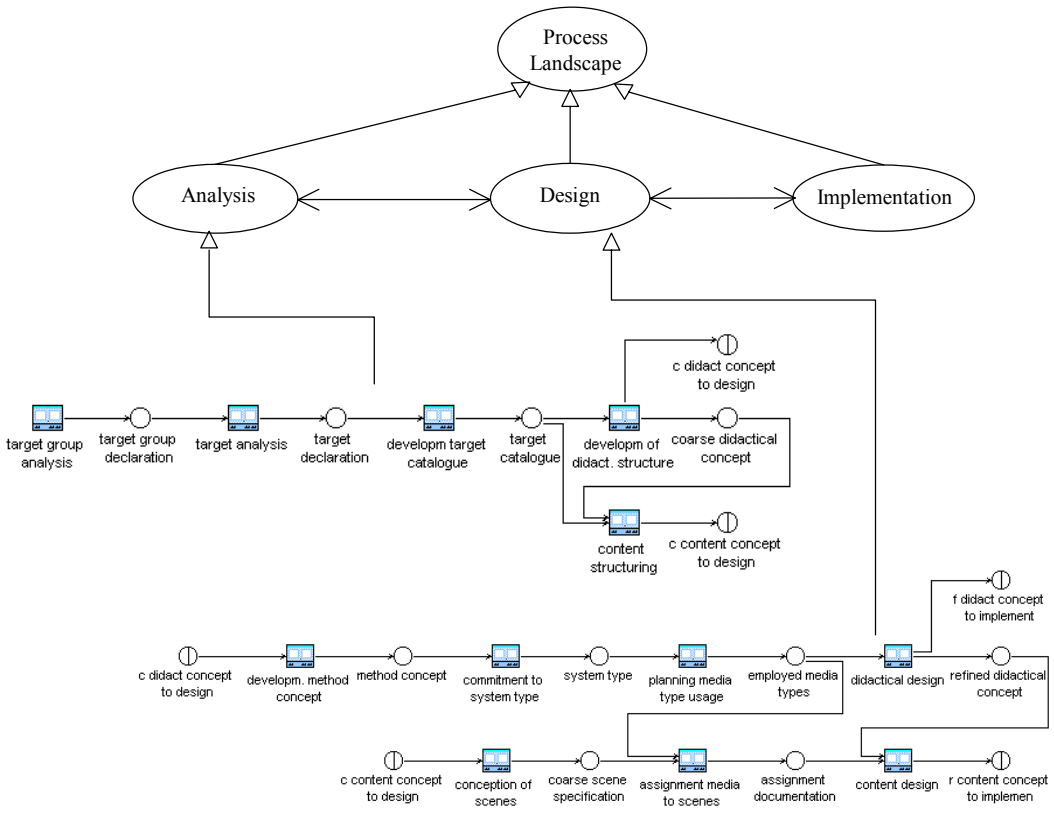

Figure 9: Logical view of the software production process landscape

5 Conclusions

In this paper we have discussed the differences between multimedia CAL systems and conventional software and their affiliated software production processes. We introduced a role-based view of the production process of multimedia CAL systems and restructured it to the logical view, both by applying the Process Landscaping method. For the restructuring we first defined logical types assigned to each activity. This resulted in toplevel clusters, submodels and interfaces focussing on logical dependencies, thus the logical view.

Besides the development of a role-based view of a software production process landscape, we have restructured this landscape to the logical view. Thus, we have emphasized the logical aspects in this landscape without loosing information about associated roles. Both, the role-based and the logical view, are necessary during the production process of multimedia CAL systems. The role-based view is useful for the participating roles to follow the production process. Beside process understanding, the role-based view provides process guidance required for the work of each role and for the cooperation between them. This ensures the acceptance of the participating roles for a process model they are familiar with. The logical view is necessary for process management to understand the entire production process. The process landscape in this view is a result of restructuring the role-based view. Understanding the descriptive logical view means understanding of a production process model concerning process traceability in order to compare it with other production processes modelled in a similar way. This is to make clear the different objectives in both views.

Our future research will focus on the extension of the role-based view of our process landscape to all roles taking part in the represented software production process for multimedia CAL systems. Further logical view refinement by assigning sublevel types to all activities and applying the restructuring algorithm on each cluster will also be done.

6 References

[Arn99] T. Arndt, *The Evolving Role of Software Engineering in the Production of Multimedia Applications*, In: Proceedings of the Int. Conference on Multimedia Computing and Systems, Florence, Italy, pages 79-84, June 1999

[ADK99] K. Alfert, E.-E. Doberkat, C. Kopka, *Towards Constructing a Flexible Multimedia Environment for Teaching the History of Art*, Technical Report No. 101, Software Technology, Department of Computer Science, University of Dortmund, September 1999

[Bod90] F. Bodendorf, *Computer in der fachlichen und universitären Ausbildung*, Oldenbourg, 1990, in German

[Boe86] B. W. Boehm, *A Spiral Model of Software Development and Enhancement*, ACM, New York, 1986

[Bro96] A. W. Brown (ed.), *Component-based Software Engineering – Selected Papers from the Software Engineering Institute*, IEEE Computer Society, 1996

[BD95] A.-P. Bröhl, W. Dröschel, *Das V–Modell*, R. Oldenbourg Verlag, 1995, in German

[BP92] W. Bischofberger, G. Pomberger, *Prototyping - Oriented Software Development Concepts and Tools*, Springer Verlag, 1992

[BRJ98] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, Reading, Massachusetts, 1998

[Cal93] J.P. Calvez, *Embedded Real-Time Systems. A Specification and Design Methodology*, John Wiley, 1993

[DEM+99] R. Depke, G. Engels, K. Mehner, S. Sauer, A. Wagner, *Ein Vorgehensmodell für die Multimedia-Entwicklung mit Autorensystemen*, In: Informatik-Forschung und Entwicklung, Vol.14, No.2, pages 83-94, June 1999, in German

[DKW98] J.-C. Derniame, B.A. Kaba, D. Wastell, *The Software Process: Modelling and Technology*, In: Software Process: Principles, Methodology and Technology, appeared as Lecture Notes in Computer Science No. 1500, Springer Verlag, pages 1-12, 1998

[GH91] K. Götz, P. Häfner, *Computerunterstütztes Lernen in der Aus- und Weiterbildung*, Deutscher Studien Verlag, 1991, in German

[GJM91] C. Ghezzi, M. Jazayeri, D. Mandrioli, *Fundamentals of Software Engineering*, Prentice-Hall, 1991

[GPS93] F. Garzotto, P. Paolini, D. Schwabe, *HDM – A Model-Based Approach to Hypertext Application Design*, ACM Transactions on Information Systems, Vol. 11, No. 1, pages 1-26, January 1993

[GW00a] V. Gruhn, U. Wellen, *Structuring Complex Software Processes by "Process Landscaping"*, In: Reidar Conradi (ed.), 7th European Workshop on Software Process Technology, EWSPT 2000, Kaprun, Austria, February 2000, appeared as Lecture Notes in Computer Science No. 1780, Springer Verlag, pages138-149, 2000

[GW00b] V. Gruhn, U. Wellen, *Process Landscaping – Eine Methode zur Geschäftsprozessmodellierung*, In: Wirtschaftsinformatik, Vol. 4, Vieweg Verlag, pages 297-309, August 2000, in German

[GW01] V. Gruhn, U. Wellen, *Analyzing a Process Landscape by Simulation*, accepted submission in Journal of Systems and Software, Elsevier, to appear in 2001

[ISB95] T. Isakowitz, W. A. Stohr, P. Balasubramanian, RMM*: A Methodology for Structured Hypermedia Design*, Communications of the ACM, Vol. 38, No. 8, pages 34-44, August 1995

[Jen97] K. Jensen, *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use*, Volume 1, second Edition, Springer Verlag, 1997

[MF96] S. J. Morris, A. Finkelstein, *An Experimental Hypertext Design Method and Application in the Field of Art History*, Computers and the History of Art, Vol. 2 (part 2), pages 45-63, 1996

[Mül95] W. Müller, *Interaktive Medien im professionellen Einsatz*, Addison Wesley, 1995, in German

[NKF+95] B. Nuseibeh, J. Kramer, A. Finkelstein, U. Leonhardt, *Decentralised Process Modelling*, In: W. Schäfer (ed.), 4th European Workshop on Software Process Technology, EWSPT'95, Noordwijkerhout, The Netherlands, April 1995, appeared as Lecture Notes in Computer Science No. 913, pages 185-188, 1995

[Poh95] K. Pohl, *A Process Centered Requirements Enginering Environment*, PhD thesis at RWTH Aachen, 1995

[Poh00] A. Pohlmann, *Visualisierung und Simulation von Prozesslandschaften – Die lokale Sichtweise*, master thesis at the University of Dortmund, Department of Computer Science, Software Technology, May 2000, in German

[Rei86] W. Reisig, *Petrinetze*, Springer Verlag, Germany, 1986, in German

[SKT+97] J. Schneeberger, N. Koch, A. Turk, R. Lutze, M. Wirsing, H. Fritsche, P. Closhen, *EPK-fix: Software Engineering und Werkzeuge für elektronische Produktkataloge*, In: M. Jarke, K. Pasedack, K. Pohl (eds.), Informatik 97. Informatik als Innovationsmotor, Berlin, Springer Verlag, pages 446-455, 1997, in German

[SS96] J. Sander and A.-W. Scheer, *Multimedia Engineering: Rahmenkonzept zum interdisziplinären Management von Multimedia-Projekten*, Research Report No. 132, Institut für Wirtschaftsinformatik, University of Saarbrücken, July 1996, in German

[SR95] D. Schwabe, G. Rossi, *The Object-Oriented Hypermedia Design Model*, Communications of the ACM, Vol. 38, No. 8, pages 45-46, August 1995

[Wei00] C. Weidauer, *Generatorunterstützte objektorientierte Entwicklung multimedialer Lehr- und Lernsysteme zur Effizienzsteigerung und Qualitätsverbesserung*, In: K. Mehlhorn , G. Snelting (eds.): 30. Jahrestagung der Gesellschaft für Informatik, Informatik 2000, Berlin, September 2000, Springer Verlag, in German

[Yan98] Y. Yang, *Issues on Supporting Distributed Software Processes*, In: V. Gruhn (ed.), 6th European Workshop on Software Process Technology, EWSPT'98, Weybridge, UK, appeared as Lecture Notes in Computer Science No. 1487, Springer Verlag, pages 143-147, September 1998