Didactic Models as Design Representations

Chris Stary

University of Linz, Department for Business Information Systems, Austria
Christian.Stary@JKU.AT

Abstract. The contribution focuses on the role of didactic knowledge when designing interactive e-learning environments. Several representational approaches for the preparation of domain content and learning support have been developed. However, for the context-sensitive design of interactive artifacts not only the representation of particular aspects of learning is essential, but rather the propagation of didactic knowledge to functional services and interaction facilities. Such an endeavor requires the explicit representation of relationships between structure and behavior elements. Model-driven design supports the distinctive representation of multiple perspectives while allowing the mutually tuned refinement of design elements. In this paper a model-based approach for self-organized e-learning is presented. It supports the design of learner-centered knowledge acquisition by specifying user roles and learning tasks. We discuss the required enrichments of traditional model-based design approaches, due to the consistent tuning of high-level design elements, and the coherent propagation of task and user information to interaction services.

Keywords: model-based design, e-learning, learning management, coherence, consistency, integrated specification.

1 Introduction

With the advent of didactic ontologies [5] design representations have become a crucial issue in e-learning. They refer to the pedagogical value of conveyed content and the related interaction features for learning management. They are supposed to capture all information relevant to this type of interactive contextualization and customization.

One of the advantages of model-based design is to address different perspectives on development information in a mutually dependent way [1]. As such, user characteristics, content, communication, and interaction can be represented in dedicated models as well as in form of mutually related models. Using this approach, when a user requests a particular piece of content or interaction media for use, it can be delivered in a way that is compatible with individual needs and preferences.

Given this context-sensitivity, there are two issues to be solved for development: Firstly, how can elements from one model be related to another at the time of design? Secondly, how can matching between two models be facilitated such that the appropriate information is delivered to the user, based on organizationally relevant roles and personal preferences? Both issues are addressed in the following, as we focus on the role of didactic knowledge when designing interactive e-learning environments.

J.A. Jacko (Ed.): Human-Computer Interaction, Part IV, HCII 2009, LNCS 5613, pp. 226–235, 2009.

Although several techniques for guiding the preparation and representation of content and transfer of knowledge have been developed, the benefits of model-based design have barely been explored. However, for context-sensitive design both, the representation and propagation of didactic knowledge to functional services, and the intertwining of model elements and relations, such as between the user and the domain model, are essential. They are discussed revisiting existing model-based approaches and reflecting recent e-learning platform developments.

2 Constructivist e-Learning: The Scholion Project

Constructivist approaches to e-learning provide environments to explore information and guide learners actively to build individual mental processes that occur during the construction of mental representations. Active (re-)construction is seen as particularly beneficial for learners as they can pursue their individual interests, while they are motivated to communicate their understanding to others. As we know from studies in constructionism, the situated and public nature of any construction activity is identified as important [3]. Both, the individualized content, and the social aspect of learning processes have to be tackled by learning management. Such an understanding of e-learning is grounded in mathetics, a special science of (scientific) teaching centered on the operational processes of the learning, and dealing in particular with the principal aspects of conceptualization and cognitive organization. Didactics, defined as the strict scientific core of pedagogy, are thought of as inconceivable from any viewpoint other than a mathetic one [13].

Computer technologies for learning have opened up new avenues for designing content. In the sense of mathetics they can trigger active learner participation along knowledge provision and acquisition processes. To meet the requirements for computer-mediated context-sensitive and collaborative learning, it must be possible for learners to explore different categories of information in virtual environments and to communicate, so that meaningful learning of a domain can proceed in tandem with establishing communities of practice. Still, the ultimate goal is to create personally meaningful mental representations [6].

The significance of being able to treat learning as a socially valid exploratory activity, rather than a linear, pre-planned activity in detail, is recognized by coaches and developers step-by-step. One appreciation can be gained through looking at deeper issues than domain-specific structures of knowledge, web-design of user interfaces, or domain-specific methods. It addresses context from different perspectives: (i) the *mathetic knowledge* that drives the provision and acquisition of knowledge – developers have to look for a proper preparation process, (ii) *communication channels* utilized for learning and guidance – developers have to look for links of communication entries to content items, and (iii) *information beyond the core of domain content*, such as cultural issues like ethno-computing in computer science education – developers have to look for additional information to facilitate comprehensive understanding a topic. Our research so far has not only been targeting at mathetically effective content development, but also towards context-rich guidance and situation-sensitive learning ([12], scholion.ce.jku.at, www.blikk.it).

The frame of reference developed in the Scholion project initially requires the didactically relevant decomposition of learning material into so-called blocks, such as 'definition'. They serve as focal point in the learning process and might be encoded into different media (text streams, images, videos etc.). Hence, multiple (re)presentations of content (polymorph content) may exist. Hyperlinks between blocks and media are likely, once linear learning material (information) is decomposed and de-linearized. In addition, different levels of detail (LoD) for each block can be specified, e.g., providing a slide for a definition on the top level (LoD 1) based on the full text of the definition on LoD 2, e.g., representing a textbook.

Annotations constitute individual views on content items by commenting, linking, or highlighting items, block elements, or enriching content blocks. Some of those annotations can be links to communication entries of the Scholion communication components (chat, forum, infoboard etc.). In this way communication elements are directly linked to content blocks and *vice versa*. Communication can be established among peers for learning, as well as between learners and coaches. The latter, as quality managers, are responsible for improving content and structures based on learner input and feedback.

In Scholion the content is arranged according to the aforementioned didactically relevant information types, conforming to current e-learning standards (cf. www.imsproject.org). They represent leaves in the hierarchy course→ module→ learning unit. Currently, 15 generic types of this sort are available as part of an XML schema. They comprise definition, motivation, background, directive, example, self test, and other domain-independent content structures. Domain-specific block types can be added to support dedicated applications, such as proof when handling content from mathematics. Each block type can be visualized in Scholion through a certain layout, e.g., green background for proof.

Block types allow learners and coaches to browse content along specific categories using a filter function. The workspace then contains only filtered block types. In this way learners might follow their interests and habits, such as starting to learn with studying background information.

In order to establish self-organized learning processes Intelligibility Catchers (ICs) are offered to learners [11]. ICs are assignments made available through the bulletin board in Scholion. They have been designed to foster in-depth understanding of a topic. Their *orientation* section addresses the stage of capacity building when the IC should be used and what learners can expect when accomplishing the IC tasks. This section should motivate learners. It can be compared to an advanced organizer. The *objectives* set the scope in terms of the topics that are addressed and the understanding that should result from exploring and processing the topic information. It reflects the didactic value of the concerned learning unit(s), and serves as a means of orientation for learners. The *task* section contains a list of tasks to be achieved. It comprises a documented work and an intellectual work part. The documented work lays ground for the tasks to be represented in task-based model representations (see next section). It encourages active information search and processing as well as focused communication. The *conference* section sets the rules for establishing a corresponding community of practice, in particular when meetings or interactions should occur. The *reference* section provides links to material that helps to accomplish the tasks. The *bulletins* can be dynamically created, and are available in the Scholion information

board. They are intended to facilitate the completion of the assignment. Finally, the *departmental cuts* reveal the estimated individual effort to meet the objectives. It might be expressed in terms of credit points or the estimated IC completion time.

The structure combines organizational with subject-specific information. The information is arranged from a mathetically relevant perspective. For instance, the orientation section in the beginning informs coaches and learners when to use this IC, addressing competencies, the content involved, the rationale for choosing this content and for co-constructing mental representations. Initially, the learners are encouraged to identify those blocks of learning units where information is already available, i.e. part of the prepared content. In case of an IC for learning UML this can be design class diagrams. Then the learners are asked to enrich particular content items, e.g., UML class diagrams with task-specific information stemming from a case study of their choice.

Typically after some practical work, such as modeling in UML, all results are shared with peers. In Scholion it is enabled by dedicated views on the original content. Views are like transparent slides put on top of the prepared content. They contain all content enrichments, such as comments, individual examples, and various types of links. As such, the content items are directly linked to discussion items according to the learning task, e.g., to develop an understanding about the proper use of UML class diagrams. All results can be validated by the coach through feedback, in order to ensure correct learner representations.

The annotation facility of Scholion is considered as the key to individualization. It is based on a flexible hypermedia scheme for the representation of content elements. It enables learners to (i) mark a specific position in a content element for learning, (ii) post questions, answers or comments, and (iii) additionally link the contribution to a discussion theme from the system's discussion board when working with content. The latter link may guide the user to the adjacent discussions of content. In case of real-time online connections, e.g., chats, the questions and answers can pop up immediately on the displays of all connected users (available in a buddy list). The referenced content elements can be displayed at the same time.

In the discussion board topics of discussions can be created either manually by users or triggered by coaches posting a learning input. For the sake of traceability, each discussion contains a vector of contributions, being part of a certain discussion group, and ordered by relevant themes.

3 Model-Based Design Representations

In model-based development the identification of models has been dominated by a top-down approach to the design and implementation of user interfaces[8]: The t*ask and domain (object) model* is located at the top layer of abstraction, as this model is expected to capture all activities that need to be performed in order to reach user goals. They might be detailed through properties (attributes), and arranged along a hierarchy representing causal or temporal mutual relationships. In addition, this model is expected to capture (business) objects that have to be manipulated in order to perform tasks. The a*bstract user interfac*e *model* is located below the top layer, as it contains the logical structure required for task accomplishment at the user interface.

Abstract elements of interaction styles are captured in their mutual context, e.g., an area for data manipulation in a browser-like way. The relationships to user tasks or goals are encoded, e.g., to indicate that at a certain point in time options for navigating in a task hierarchy have to be offered. The final codality of information for presentation (text, graphics a.t.l.) and user control (modality) are not specified at that layer of abstraction. The c*oncrete user interfac*e *model* captures the refinements of all abstract interaction elements to more concrete ones. The initial abstract specification of style elements is replaced with one containing concrete attributes and relationships between interaction elements, including the modality of interaction. At that point, platform- and media-specific information is attached, in order to enable physical access to an application. The instantiation of the concrete model leads to an actual user interface, utilizing (standard) programming technologies, such as Java.

The envisioned interplay of the models in the course of task-based design can be exemplified as follows: Consider booking a course at the university. It requires several sub tasks, such as finding proper options, selecting class data a.t.l.. This information is kept on the task (model) level, in addition to information for selection and manipulation, such as course. The subsequent abstract user-interface model captures all (categories of) interaction elements that are supposed to support each subtask. For instance, to identify course options an entry field for search has to be provided. Finally, the concrete user-interface model captures all specific interaction elements. For instance¸ in a web interface, search entry fields are supported by text bars after selecting a search engine (and site). The actual user interface can be constructed based on the specification of the concrete user interface model.

Such a top-down approach focusing on layered design perspectives facilitates the use of multiple platforms and various arrangements of interaction elements. It encodes rather than makes transparent Gestalt design knowledge - knowledge that is mainly created in the course of mapping the results of user and work analysis (located in the top level task/domain model) to abstract interaction elements. The initially acquired (and represented) context of an interactive application does not directly guide concrete user-interface designs.

A contextualized process throughout development requires a shift in model-based design (support), namely (i) a more detailed and flexible representation of users, work tasks, and domain elements, at least to the same extent as user-interface representations, and (ii) explicit conceptual relationships among design elements (inter- and intra-model-specific), considering universal access from a usability and software-engineering perspective in a more integrative but self-contained way.

Such an approach has to take a more networked rather than hierarchic perspective on model-based development. It requires a set of models that still enable the specification of an application model (in the sense sketched above), but details the user- and use-relevant elements, e.g., in terms of a task, user, domain, and an interaction model. The traditional engineering perspective has to be enriched with design variants capturing interactivity before considering functional implementation. The following set of models lays ground for model-based tools that recognize the need for supporting diverse user communities and situations of use [10].

The *task model* contains details of the organization at hand, the tasks, and the users' perception of work. Task modeling includes modeling of the objectives

users want or have to meet using an interactive application, probably in response to particular situations or events, as well as the different activities users have to perform to accomplish their tasks, as, e.g., given by global business processes.

The *user model* contains a role model by reflecting specific views on tasks and data (according to the functional roles of users in organizations). It also captures individual user characteristics, capabilities or specific needs that developers need to take into account.

The *domain (data) model* addresses all data, material, and resources required for task accomplishment in the application domain.

The *interaction model* is composed of device and style specifications that designers need to construct a user interface or to generate a user-interface prototype. Hence, the structure of modalities and its related behavior are captured in that model, both at an abstract, and concrete level. For the developers' convenience platform-specific specifications can become part of the interaction model. In the case where behavior specifications are missing, the preprogrammed, platform-inherent behavior might restrict the design space.

The *adaptation model* contains all rules that trigger a particular structure and/or behavior of an interactive application. As such it enables interaction adapted to user-model elements, such as left-hand assignments to interaction media (e.g., mouse buttons) based on consistent representations. It also enables the multiple presentation of task-information for navigation, e.g., tree views and acoustic trees. Since its scope includes all the previously listed models and their relationship, adaptation may concern task, data, interaction elements, and their inter-relationship.

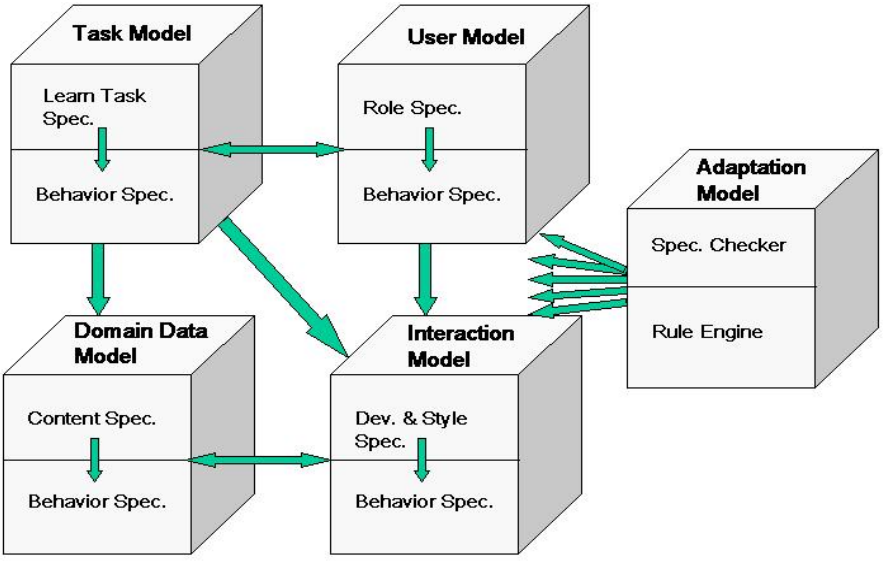

Fig. 1. Model-based specification for e-learning tasks in ProcessLens

The *application model* integrates all five models from a static and dynamic perspective. In this way, it defines the entire structure and behavior of an interactive application. For interaction the users' privileges, preferences, special needs, and tasks to be performed are taken into account when mapped to a concrete device or widgets of the interaction model.

The ProcessLens approach [2] also expands the model-based design space, providing a design ontology for task-driven design. It can be used for task-based e-learning design as shown. Figure 2 demonstrates the enrichment to traditional design. It leads to a context-sensitive specification based on the mentioned models. For learning management the task model has to contain the tasks of the documented work-part of the IC for learners (see section 2) and the preparation of the IC by the coach. 'Capture UML' is decomposed into 'IC-Specification', 'View Management' and 'Discuss'. ProcessLens also shows the role-specific responsibilities through a role-based user model. The coach 'handles' the preparation and the view management, whereas the learners handles views and forum entries in the course of discussion (see subtasks of 'Discuss' in Figure 2).

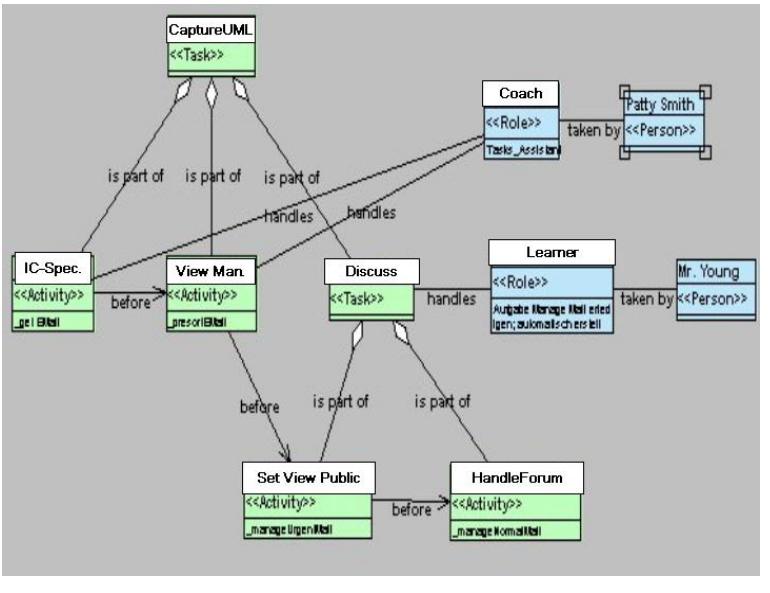

Fig. 2. Part of e-learning task hierarchy and roles in ProcessLens

A typical example of a design relationship is 'handles' between roles and activities. It binds responsibilities and allows for role-specific interaction domains. 'Before' demonstrates a causal relationship that has to be transformed to a set of temporal constraints in the behavior specification. A behavior specification is exemplified in Figure 3. It shows a coherent propagation to problem domain elements for being invited to a discussion. In the course of 'Discuss' peers have to be invited to join a certain community of practice. It is done by e-mail which requires to handle e-mails from the task and the problem-domain perspective. As content and communication are of equal importance in e-learning, communication facilities have to be represented in the problem domain model besides the content (decomposed to block types).

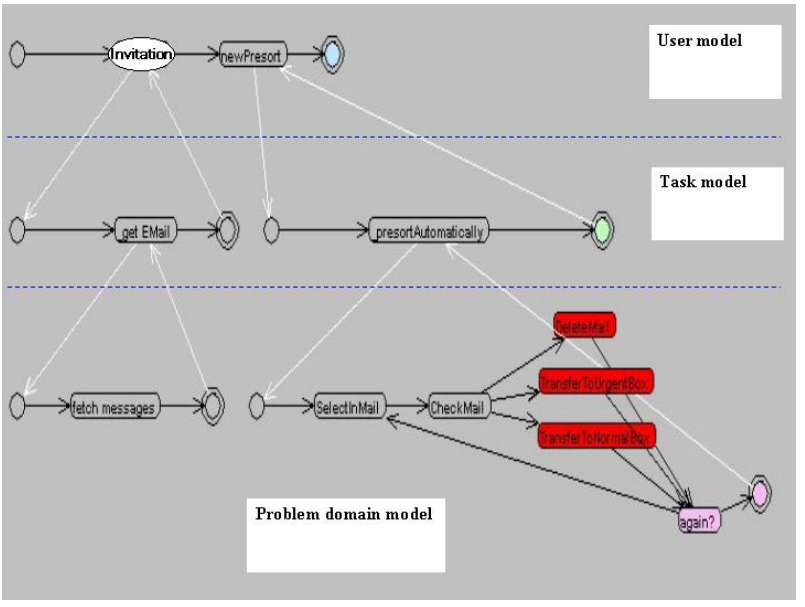

Fig. 3. Model-based behavior coupling in ProcessLens

The interaction model is also decomposed into a structure and behavior part. The latter contains the life cycle of each widget and interaction style as required for device types. For instance, Scholion instances can also be used on web mobiles (see www.mobilearn.at). In ProcessLens the abstract and concrete layer of description is captured within a single model. The coupling of behavior sequences from other models is done analogously to the synchronization shown in Figure 3. Dedicated synchronization relationships have been developed to ensure behavioral consistency of design specifications.

In order to implement a user interface for learners, relations have to be set between the task model and interaction model, as well as between the problem domain and the interaction model. The first link is required to navigate according to the tasks of the IC-documented work part. The latter is required to annotate content, and to communicate on the basis of linking content elements to communication entries.

For interaction modeling besides generic UML structures (as used in ProcessLens), e.g., for modalities [7], XML derivates, such as USiXML [4] are widely used. They facilitate transformations as well as the dynamic creation of models, relying on various models. For instance, XIML [9] as user-interface description language supports the specification of various (linked) interaction models simultaneously. Its definition allows to capture task, user, presentation and dialog models as well as platform details. However, there is no executable relationship between the various interaction model parts, as provided by ProcessLens.

4 Conclusions

As the pedagogical value of conveyed content and the related interaction features for learning support has become a distinctive factor of e-learning environments, design representations have moved to the center of interest in development. They have to capture the structure and dynamic context of subject items. Model-based design techniques not only allow different perspectives to be addressed, they also allow for capture of mutual relations between design models, both from the structure and the behavior point of view.

In this paper we have proposed a model-based frame of reference and a corresponding tool supporting the development of e-learning applications. It focuses on the relationships between model elements to ensure consistent tuning and coherent propagation of design knowledge for self-managed learning processes. Of major importance are learning tasks to be accomplished by learners, as they require dynamic handling of domain content and communication elements at the user interface. Although the proposed adaptation model provides design constructs to represent this system dynamics, there is still research to be performed to achieve algorithmic support for automated model transformation.

References

1. Calvary, G., Coutaz, J., Dâassi, O., Balme, L., Demeure, A.: Towards a new generation of widgets for supporting software plasticity: The ”Comet”. In: Bastide, R., Palanque, P., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 306–324. Springer, Heidelberg (2005)
2. Dittmar, A., Forbrig, P., Heftberger, S., Stary, C.: Tool Support for Task Modelling – A ‘Constructive’ Exploration. In: Bastide, R., Palanque, P., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 59–74. Springer, Heidelberg (2005)
3. Farmer, R.A., Hughes, B.: A Situated Learning Perspective on Learning Object Design. In: Proc. ICALT 2005. IEEE, Los Alamitos (2005)
4. Limbourg, Q., Vanderdonckt, J.: Addressing the Mapping Problem in User Interface Design with UsiXML. In: Proceedings TAMODIA 2004, pp. 155–163. ACM, New York (2004)
5. Meder, N.: Didaktische Ontologien. In: Ohly, G.R.H.P., Siegel, A., Ergon, W. (eds.) Globalisierung und Wissensorganisation. Neue Aspekte für Wissen, Wissenschaft und Informationssysteme, vol. 6, pp. 401–406 (2000)
6. Norman, D.A., Spohrer, J.C.: Learner-Centered Education. Communications of the ACM 39(4), 24–27 (1996)
7. Obrenovic, Z., Starcevic, D.: Modeling Multimodal Human-Computer Interaction. IEEE Computer 37(9), 65–72 (2004)
8. Paternò, F.: Model-Based Design and Evaluation of Interactive Applications. Springer, Berlin (1999)
9. Puerta, A., Eisenstein, J.: XIML: A Common Representation for Interaction Data. In: Proceedings IUI 2002 International Conference on Intelligent User Interfaces. ACM Press, New York (2002)

10. Stary, C.: TADEUS: Seamless Development of Task-Based and User-Oriented Interfaces. IEEE Transactions on Systems, Man, and Cybernetics, Part A 30(5), 509–525 (2000)
11. Stary, C.: Intelligibility Catchers for Self-Managed Knowledge Transfer. In: ICALT 2007, 7th International Conference on Advanced Learning Technologies, pp. 517–522. IEEE, Los Alamitos (2007)
12. Stary, C.: Sustainable e-Learning Communities. In: Akoumianakis, D. (ed.) Virtual Community Practices and Social Interactive Media: Technology Lifecycle and Workflow Analysis, IGI Global, Hershey, pp. 398–411 (2009)
13. Verpoorten, D., Pumay, M., Ledercq, C.: The Eight Learning Events Model: A Pedagogic Conceptual Tool Supporting Diversification of Learning Methods. Interactive Learning Environments 15(2), 151–160 (2007)