INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED11
15 - 18 AUGUST 2011, TECHNICAL UNIVERSITY OF DENMARK

THEORETICAL FRAMEWORK FOR COMPREHENSIVE ABSTRACT PROTOTYPING METHODOLOGY

Imre Horváth
Delft University of Technology, the Netherlands

ABSTRACT

Though abstract prototyping offers quality improvement and costs reduction in all branches of product development, it has gained popularity only in the software sector of the creative industry. This paper proposes a theoretical platform and an activity workflow for abstract prototyping of artifact-service combinations. First, the concept of abstract prototypes and the evolution of abstract prototyping are discussed. Then, an underpinning theory and a content-independent workflow are presented. It is proposed that the information constructs instantiated in abstract prototypes should demonstrate the real life operation and interaction/use processes, including the description of the conceptualized artifact-service combination, the human actors, and the surrounding environment. The stakeholders' needs should be taken into consideration not only in conceptualization of artifact-service combinations, but also at constructing the contents and demonstration of the abstract prototype. Narration and enactment are identified as two intertwined parts of demonstration. The follow up research focuses on testing the proposed methodology and its validation through complex industrial cases.

Keywords: abstract prototyping methodology, theoretical platform, information constructs, artifact-service combination, human actors, narration and enactment, stakeholder demands

1. INTRODUCTION

While models are interpreted as simplified descriptions of reality in various representation forms, prototypes are in general seen as means to bring future realities into present situations. The word prototype originates in the Latin words *proto*, meaning original, and *typus*, meaning type or variety. Hence, a prototype is a first representative example or completion of something, which is usually, but not exclusively a physical (material) artifact. From a technical perspective, prototyping is the creation and inaction of first implementations of artifacts based on the information available about the technical requirements, operational workflows, concerned stakeholders, use scenarios, appropriate materials, implementation technologies, presentation forms and business conditions in a particular phase of their development process. Rapid prototyping has been advocated as useful in practically all domains of design and engineering because it lends itself to intense interaction between customers, users and developers, and thus may result in early validation of concepts, specifications, artifact designs, and systems. The earlier prototypes are generated in the design process, the larger influence they can have on the final quality of products, though producing abundant prototypes may be time, capacity and money consuming [1]. These above have been found valid principles in the hardware, software and information system industries.
Given that prototypes may manifest in alternative forms, they can be sorted into categories, such as abstract, virtual, physical, augmented, and hybrid prototypes. The first three categories are often referred to as principal forms of prototyping. In each category low fidelity and high fidelity prototypes can be differentiated. Low fidelity prototypes apply strong simplifications in comparison with the modeled existing or imagined reality, while high fidelity prototypes seek to achieve the most congruent representation of reality. Made either by manual fabrication techniques, conventional manufacturing technologies, or dedicated rapid prototyping technologies, a physical prototype is a fully or partially operational materialized replica of an artifact [2]. High fidelity physical prototypes are supposed to be not distinguishable from the mass-produced product. Often partial and scale prototypes (mock-ups) are produced because complex, full-size and fully functional physical prototypes are in general costly and need a lot of time and resources to be produced. Physical prototypes are inflexible (difficult to change), therefore, they are usually fabricated when no dramatic changes in the design are expected.

A virtual prototype is a computer generated three-dimensional complex information model of an artifact, which (i) contains geometric, structural and physical information, (ii) is visualized and animated, and (iii) makes behavioral simulation possible [3]. Virtual prototyping (VP) has been interpreted in many different ways, and various terms have been introduced, but used interchangeably. VP is often mingled with virtual reality (VR) or virtual environment (VE) [4]. Virtual prototypes can be largely equivalent substitutes of physical prototypes, though they are not able to reproduce all physical sensations (e.g. touch and inertia). At the same time, they can be modified fast and with less effort than the physical ones. Since as-fabricated physical prototypes cannot reproduce all visual features of products or capture information about their surroundings, they are often combined with virtual contents. On the other hand, because of their limitations in presenting all tangible and perceptible properties of physical artifacts, virtual prototypes are often presented with the help of sophisticated physical sensation enabling technologies. The firstly mentioned mix of physical and virtual prototyping has been called augmented prototyping, and the second form of information enrichment is virtual reality.

Typically, it is the detail design phase of product development where virtual prototypes and physical prototypes are used. Over the years, the need for early prototyping in the idea generation activities, and even in the fuzzy front end of artifact development has gradually emerged. In line with this, the objective and focus of early prototyping gradually shifted to externalization (uttering) of product visions, embedment of products in real life situations, providing impressions about manifestations, and investigation of user experiences related to future artifacts, systems, operations, processes, services and methods. Obviously, this required novel expressive and revealing forms of prototyping in order to facilitate non-tangible communication with and demonstrations for stakeholders. Thinking in this direction has given floor *abstract prototyping* and has enabled various practical implementations of abstract prototypes (APs) [5]. Opposite to virtual, physical and augmented prototyping, various methods of abstract prototyping have been developed to provide support for the representation and assessment of often under-defined, incomplete and emergent visions, ideas, requirements and concepts. From a methodological point of view, abstract prototyping is not a low-fidelity approach to virtual or physical prototyping. It has its own distinguishing objectives, contents, methods and applications. Abstract prototyping operates mainly in between the mental and virtual domains of creative work, actually including both. Consequently, the objective of application is not testing the feasibility of concepts, or pre-manufacturing evaluation of a product, but to enable creative design activities. That is why abstract prototyping is now seen as the third principal form of prototyping. This is depicted graphically in Figure 1, which shows the domains of creative work, the major cognitive activities (inform, prepare, experience and reason), and the outcomes of the transitions amongst the domains [6].

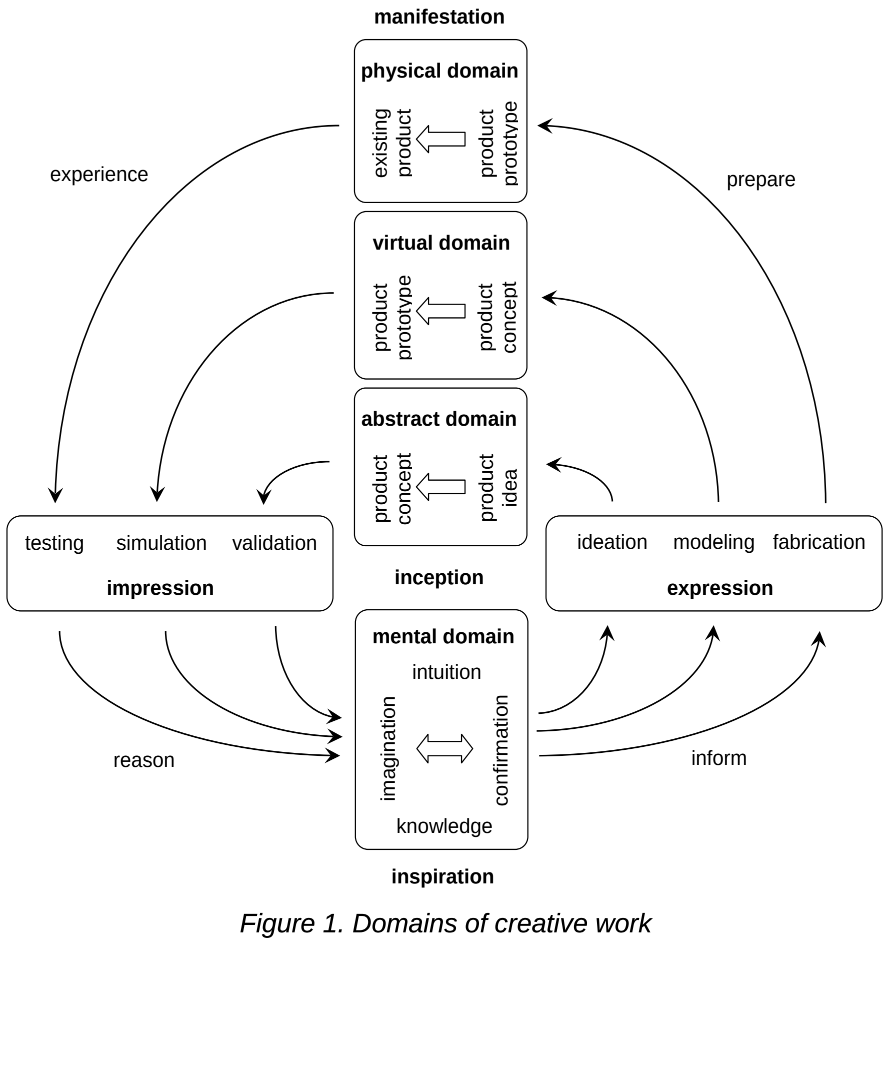

Figure 1. Domains of creative work

In a complex artifact development processes, the three principal forms of prototyping are usually employed in purposeful combinations. This form of exploitation of prototyping capabilities has been termed hybrid prototyping. A hybrid prototype is a multi-purpose and multi-representation compound, having strong couplings and information transfers between the involved prototypes. Hybrid prototyping creates a new situation for industrial product developers in which they can concurrently work in three activity and knowledge domains, namely, in the abstract (intangible), virtual (digital) and physical (material) domains, in addition to the mental (logical) domain.

While the methodologies of physical, virtual and augmented prototypes have reached a rather mature level, the development of sound methodologies for abstract prototyping is lagging behind. Besides the additional knowledge and efforts that are needed to develop comprehensive APs, this explains why abstract prototyping has gained popularity only in some specific sectors of the creative industry. Remarkable efforts have been made in the fields of software engineering, human-computer interaction, system engineering, and a few dedicated methodologies have been developed [7]. They are based on specific underpinning theories and offer content-dependent workflows for abstract prototyping. The objective of this paper is to introduce an epistemological framework to support early prototyping of designs and their creation processes. Furthermore, it also proposes a domain independent abstract prototyping methodology that can support early conceptualization and presentation of artifact-service combinations, as well as of the traditional applications. In Section 2, we discuss the known concepts, and approaches, the associated specific techniques, and major application domains of abstract prototyping. As element of the theoretical platform, Section 3 presents our most important assumptions and interpretations. Section 4 introduces a structured workflow and discusses some methodological issues of abstract prototyping. Finally, Section 5 makes some concluding remarks.

2. KNOWN CONCEPTS AND APPROACHES OF ABSTRACT PROTOTYPING

It is rather difficult to localize when and in which publication the term abstract prototyping was introduced first. There are papers from the mid-1980s that use this term to describe the process and means of enriching design information by testing the fulfillment of the requirements, functionality and usability of proposed solution concepts early in the development process via the involvement of potential users and other stakeholders [8]. The first footprints of using early prototyping can be found in software engineering, where a typical process is usually built up from the following elements: (i) identify the user's basic information requirements, (ii) develop a feasible concept of the software system, (iii) implement a working prototype and use the prototype system, and (iv) revise and enhance the prototyped software [9]. Oppositely, pre-production prototypes first received emphasis in engineering and consumer durable design. As early as the beginning of the 1990s, Cooprider and Henderson argued that in order to realize the most effective ways of applying technologies to the prototyping process, a deeper behavioral model of prototyping must be used [10]. Prototyping should strive for a more extensive cognitive interaction via prototypes, and start at the earliest possible stage of the development process. These objectives have actually placed the whole of early prototype making and testing into cognitive and social perspectives, and fostered the emergence of abstract prototyping [11].

Abstract prototyping has been variously interpreted by different authors over the years. Wood, D. P and Kang, K. C. differentiated two broad categories of prototyping approaches so as evolutionary and concept prototyping [7]. Evolutionary prototyping (also often referred to as field prototyping) involves the creation of a series of fielded prototypes. Concept prototyping intends to explore ideas without resorting to field deployment, and produces what are variously named as rapid, concept, throw-away, experimental and exploratory prototypes. Three main objectives have been identified for abstract prototyping: (i) create an enactable model of the system under development, (ii) create an environment in which the system model will operate, and (iii) create a validation suite comprised of operational scenarios. Another often used term and approach is soft prototyping that involves using computer models and databases to represent aspects of a product’s behavior [10]. Used in the specification phase of software development, a soft prototype can help determine customer needs and feature preferences, and to make data on the performance and characteristics of a product available for analysis and comparison. Chalasani, P. et al. propose a multi-period approach to software prototyping that takes into account the flexibility of being able to postpone the prototyping and design decisions [12].

In the field of software design, prototyping was widely accepted as an excellent mechanism for system and user requirements elicitation and validation [13]. It is often an explicit assumption in requirement elicitation that users know and can articulate their requirements, but it is in fact seldom true. An early prototype is a kind of mediator of formulating requirements for a particular product concept in a participatory manner. A broader concept of abstract prototyping appeared in association with the realization of usage centered design (UCD), and later on, user experience design (UED) [14]. Phalp, K. and Counsell, S. explain that an underlying assumption of abstract prototyping is that interactions (i.e., communication couplings) between the prototyper and the other parties involved in the development process are a key factor in the success of the process [15]. Hickey, A. and Dean, D. designed four forms of participative (face-to-face) prototype evaluation: (i) demonstration-based prototype evaluations, (ii) chauffeured scenario prototype evaluations, (iii) independent scenario prototype evaluations, and (iv) comprehensive prototype evaluations [9]. Constantine, L.L. and Lockwood, A. suggest that they drive their UCD approach by three closely relates abstract models: (i) a role model (salient characteristics of the roles the users play in relation to a system), (ii) a task model (a structure of tasks a user should accomplish to operate the system), and (iii) a content model (a representation of the contents and organization of the user interface (UI) needed to support the identified tasks). Actually, this last model has also been called by them *abstract prototype* [11]. The adjective 'abstract' is used to express that this prototype represents the contents of a UI independent of the actual appearance and behavior of the realized user interface.

In the field of system development and engineering, it has been a widely used strategy to build a scaled down system, or to construct a portion of a complex system in a short time, test it, and improve it in several iterations [13]. System prototyping is generally understood as the sum of all activities required to build and evaluate a first implementation in a real-world environment [16]. Vasconcelos, W. et al. describe an approach for rapid prototyping of large multi-agent systems [17]. Their methodology advocates a global (interoperation) protocol that describes all permitted interactions among the components of the system, as a starting specification. They decompose the prototyping process of a multi-agent system into four phases: (i) design of a global protocol, formalized as an electronic 'institution', (ii) synthesis of agents from the global protocol and their customization, (iii) definition of a prototype consisting of populations of the previously synthesized or customized agents to enact a global protocol, and (iv) simulation and monitoring of the prototype. According to Buchenrieder, K., three trends are going to dominate the future of hardware/software system prototyping: (i) the advent of very large reprogrammable devices combined with several processing elements on a single chip, (ii) the continuous expansion of embedded systems into virtually all electronic application fields requiring very flexible intellectual property modules, and (iii) the reuse of pre-existing components together with new software and hardware modules [18]. The clear advantage of system prototyping over simulation and emulation is that performance and function checks for components or the entire system can be executed and observed under real-life conditions.

Another area of research interest is prototyping of the activities and interaction of humans with (complex) systems. One of the many studied issues in human-work interaction design, where a distinction have been made between task-oriented and function-oriented approaches [19]. Task-oriented design (TOD) aims at providing information to enhance the performance of tasks, which can be predicted. To this end, TOD needs specific analysis methods, which can identify users' knowledge structure consisting of goals, procedures (including pre-condition, post-condition and states, actions and operations), related objects, and so on [20]. On the other hand, function-oriented design operates with information related to the purpose-implied functions and their connections to the work domain. In the field of design for interaction, early prototyping of products for human interaction and the design of UIs are often demarcated, but they are developing as strongly interrelated research fields. To support the design of user interfaces, various formalisms and means have been proposed. A specific approach to abstract prototyping of user interfaces is creating canonical APs [21]. Canonical abstract prototyping (CAP) involves the concepts of navigation maps and interaction spaces. CAP has evolved into a symbolic UI specification language that allows connecting the envisioned user actions with the concrete control elements of the UI, and vice versa [22]. This language enables the specification at a higher level of abstraction than that is typical for paper prototypes. Two basic entities of the language are container (image, data, object) and tool (function, control, link), which can be combined and further specified.

A demonstrative computer system supporting the use of these entities is the CanonSketch tool [23]. The specifications generated this way can be converted to Unified Modeling Language representation after syntax adaptation. This approach has been realized in the pilot system named DialogSketch up to the level of animating task execution and decision flows [24]. CAP systems allow specifying the contents and general layout of user interfaces without any commitment to the details of appearance, and enable experimenting with the use of the control elements. They offer a kit of standard abstract components described in a uniform notation, and provide means for their arrangement in the layout of the UI, including the position, relative size, nesting, and overlay of abstract components. Since the resulting layout resembles the actual UI, it can be transferred straightforwardly to concrete elements and realizations. There are even more sophisticated means, such as the Abstract Window Toolkit (AWT), which came originally to existence as the Java's platform independent user-interface widget toolkit including graphics and windowing. The AWT has become part of the Java Foundation Classes (JFC), which is the standard application programming interface (API) for providing a graphical UI for a Java program.

Proper consideration of the characteristics of humans is a research area on its own right in design for interaction, as well as in interface design. User interface prototyping tries to connect the functionality of the product with human control [25]. Da Cruz, A. M. R. and Faria, J. P. argued that interpretation of the essence of UI depends on the community the UI designer belongs to [26]. UI designers with software engineering background tend to highlight the system functionality issues and connect the interface and the user with the system model and behavior. The human-computer interaction community tends to focus on the user need, characteristics and experience, the way the user shall be and prefer working, and adapts the control tasks and derive rules for the system design based on these factors. Abstract UI prototypes offer designers a form of representation for specification and exploration of visual and interaction design ideas that are intermediate between abstract task models and realistic or representational prototypes. Designers have long used various information/knowledge constructs to represent humans [27]. These have been referred to by various researchers as persona, use case model, user profile (UP), and avatar. Actually, they all are built around hypothetical usage scenarios, and connect them with user behaviors [25]. However, behind these synonyms there are somewhat different contents and objectives of usage. For instance, UP is a concept and method frequently used by industrial companies to present data from studies of user characteristics and to represent real users. For some authors, a UP specification is actually a description of a fictitious persona supplemented with a description of relationships between other users [28]. The UP of a target group contains collective mental, physical and demographic data about a user population.

Fuelled by the knowledge of information system engineering, model-based UI design and development has been the latest advancement in the field of abstract prototyping [29]. Models are supposed to be sufficiently generic, comprehensive and flexible. This approach involves the development of various aspect models, such as actor models, role models, task models, scenario models, operation models, and information flow models. Many of them have been used in combination, or even integrated with others, but no all-embracing models have been reported in the literature. Clerckx, T. et al. discussed that model-based approaches to interface design involve multiple model components that are used to model the user’s tasks (task model component), behavior of the UI (dialog model component), the relevant context information (context model component) [30]. The whole operation process of the interface is documented as a state chart model, where the nodes are, for example, a specific task or a particular state of the UI, and the edges are state transitions. It is still in a premature stage to produce a concrete UI from abstract models automatically, though efforts have been made in this direction [31]. Model-based UI development approaches are gaining popularity in the mobile communication industry. It has been shown how a combination of a patterns-based modeling language and a generic adaptive architecture based on components with ports and utility functions for finding the optimal configuration in a given situation facilitates implementation of applications with adaptive UIs. [32]. They have been successfully extended to support collaboration and creative activities [33]. Many goals and principles of model-based interface design can be seen back in model driven architecture development.

3. IMPORTANT ASSUMPTIONS AND INTERPRETATIONS

Exists there any grand theory or a set of elementary theories that can explain these various manifestations of abstract prototyping and that can be transformed into a generic prescriptive theory of abstract prototyping? What kind of application independent generic methodology can be developed based on this underlying theory of abstract prototyping? These have been the major research questions in our research and below we summarize our first results achieved so far. We assumed that, by generalization of specific interpretations and implementations, we can in an inductive manner devise a theory that is able to explain the essence, constituents and conduct of abstract prototyping beyond the phenomenology of the existing approaches of abstract prototyping. We set the target of our effort so as to support the early conceptualization of *artifact-service combinations* (ASCs) in real life circumstances. Obviously, this implied the need for converting the compiled prescribing theory into a comprehensive methodology, which is advises simultaneously both on the procedure (workflow) of abstract prototyping and on the contents to be built into APs.

As a first step, we explored the commodities and the complementing features of the various approaches briefly analyzed in Section 2. Obviously, in order to be able to construct a theory, we had to make a number of assumptions. The most important ones are those listed below:

a. An artifact-service combination is an existing or to-be-built product (a system, taken as a whole) that consists of a set of industrially made *tangible objects* and the inseparable realization of a set of *intangible and variable benefits* by making use of their functionality and/or operation of the artifacts.
b. The purpose of abstract prototyping is to demonstrate and make assessable foreseen (non-existing) *real life processes* that are established by the functioning of a new ASC in various application environments. The AP models both the contents and the logics of the foreseen process,
c. The process established by the ASCs may comprise *sub-processes*, such as (i) the physical operation of the tangible objects, (ii) the controlled and/or autonomous interoperation of the tangible objects, (iii) the sequence of activities of individuals and/or collaboration of teams of people, (iv) the operationalization of the services, and (v) the interaction of individuals and/or teams with the tangible objects.
d. In addition to model the elements of foreseen process, the AP should describe the *contributors to the process*, namely (i) the concerned ASC itself, (ii) the human beings who are involved in the functioning of the ASC, and (iii) the environment in which the functioning takes place. The contents and logics of the process can be captured by multiple scenarios. A scenario includes information about objectives, objects, relationships, actions, interactions and events.
e. The AP is presented to *stakeholders,* who have some specific relationships with the developed ASCs, and interested in the assessment and improvements of its concept. The stakeholders are external to the foreseen process, while the human beings who are involved in the process are internal actors.
f. The contents of the AP (i.e. the process established by the ASC and the contributors) are conveyed to the stakeholders through two major ingredients of the AP, namely by the narration and the enactment. *Narration* is a simplified synthetic description and presentation of the outline or story of the to-be-demonstrated process. *Enactment* is a media-enabled dramatization and visualization of the foreseeable performance and particulars of the process.

Based on the above assumptions and the explored common conceptual elements, we have developed a reasoning model, which is graphically shown in Figure 2. The reasoning model is built around the assumption that the to-be-demonstrated process is the focus of abstract prototyping, and the objective is to specify its contents. The three contributors to the manifestation of the process are: (i) the conceptualized artifact-service combination (Ξ), (ii) the involved human actors (H), and (iii) the embedding environment (Σ). The demonstration of the technical contents of the AP is based on media-enabled visualizations and accounts (M), which do however not contribute to the technical contents. The process is demonstrated for stakeholders (SH) who are interested in the conceptualized ACSs from business, implementation and/or application perspectives, and shall receive information about the proposal through the demonstration of APs. Taking these into consideration, an *abstract prototype* has been defined as a self-contained, digitally recorded, multi-medium enabled information structure which represents real life processes established by particular manifestations of artifact-service combinations.

According to our reasoning, the AP should include at least the below information about the sub-process, the contributors, the media and the stakeholders, based on the following considerations:

a. As specified above, the modeled process is a composition of sub-processes, which have to be modeled in terms of their interactions, not only separated. This implies the need for a generic representation that can capture the courses and features of all sub-processes. For this reason, a process is considered as a purpose-driven, time-ordered (and self-irreversible) *flow of changes*. The changes can in turn be captured as a web of transitions between subsequent start (input) and

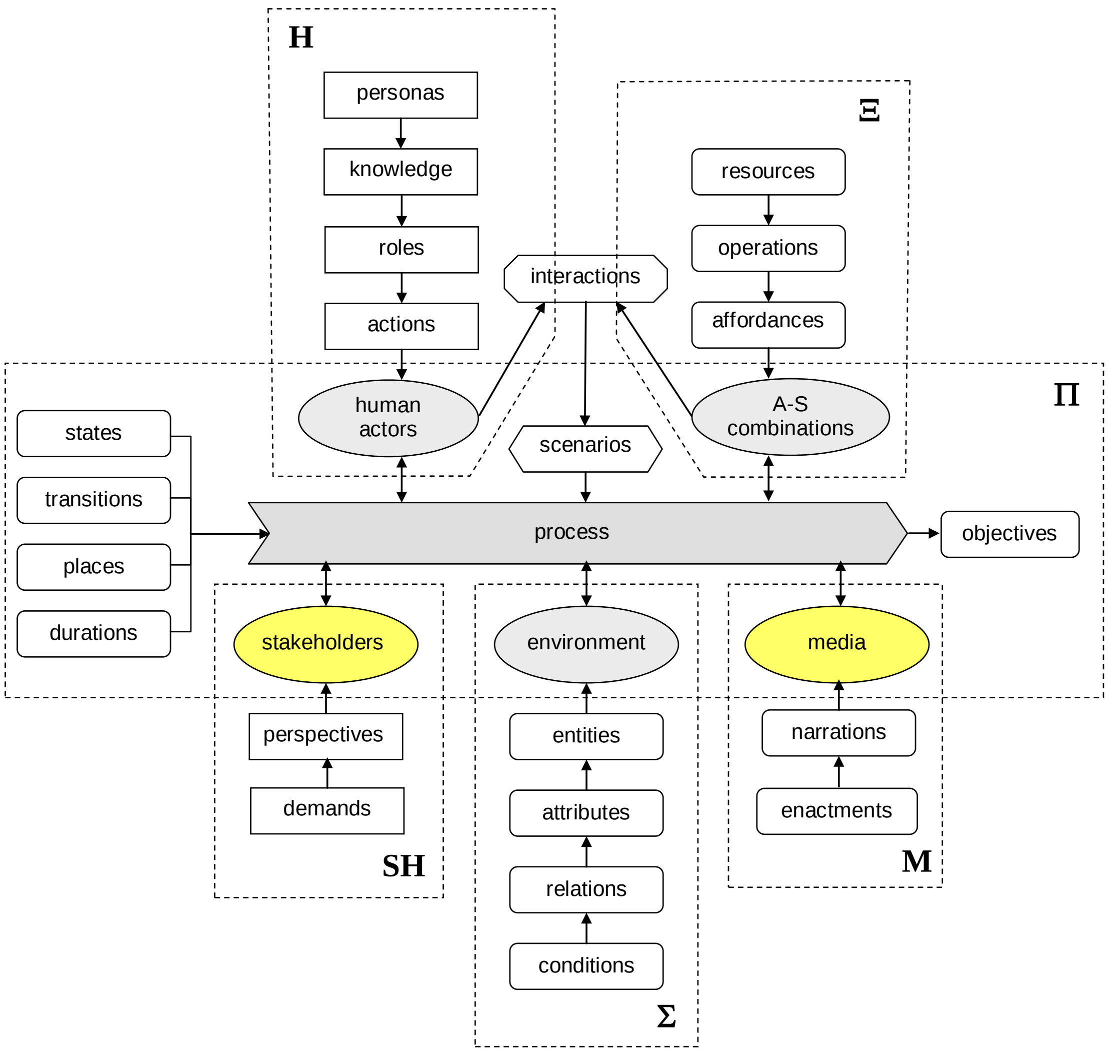

Figure 2. The information constructs to be instantiated in abstract prototypes

end (output) states. For modeling of the operations and interoperation of sub-processes, he objective, states, transitions, places and durations of the ordered changes must be specified as *information constructs*.

b. The operation of the ASCs is added up by the *controlled changes* of its functional/structural components, including the tangible objects and the means of realization of the services. They are considered and described as resources for the realization of the operation and interaction sub-processes. The harmonized operations and interactions of these resources create various affordances for the human actors. A powerful means for a combined representation of the resources and state transformations is *scenario*. A process scenario provides a high level logical model of the envisaged manifestation of the real life process, and can be converted into concrete workflows in the later phases of development. The scenario-based representation can be used as a control mechanism for a computer-based physical simulation of the concrete operation and interaction processes [Wilfred].

c. The operation sub-processes are intertwined with interaction sub-processes, which depend on the intents and actions of the human actors. The AP should typically model the whole body, the face, and specifically the *manipulative organs of the human actors*, and represent their actions as state changes. In addition, the AP should demonstrate the roles, tasks, features and related behavioral characteristics of human actors. In our conceptualization, the involved human actors are modeled as personas, which represent a cluster of individuals, rather than a single individual. Consequently, personas have generalized, but distinguishing personal characteristics, and physiological and cognitive capabilities. Following from their roles, personas fulfill tasks and make specific control and use actions. They are also described in terms of their knowledge concerning the ASC and the surrounding environment.

d. The surrounding environment is seen as a *composition of various artifactual and natural entities,* which are in dynamic interactions with the embedded ASC and each other. The entities may have unvarying and/or variable characteristics, and facilitate or restrict the operation and interaction sub-processes of the actors and the resources. These can be represented as conditions.
e. The strategic goal of abstract prototyping development is to involve the stakeholders in the assessment of the ASC concepts in an early phase of their development. A pragmatic goal is to facilitate the informed decision making of the stakeholders on the proposed ASC (but not on the AP itself). Although they are not described explicitly in the APs, the *analysis of the stakeholders* is an important element of the abstract prototyping process. For a successful abstract prototyping, it is indispensible to (i) identify all stakeholders, (ii) clarify their perspectives (the aspects of their decision making), (iii) analyze their concrete demands concerning the functionality and usage of the ASCs, (iv) adapt the information delivered by the AP, and (v) making insightful decisions on the form of delivery of the AP.
f. The demonstrative media plays a crucial role in the success of delivering the contents to and communication with the stakeholders. In order to be objective and expressive, rather than overwhelming and misguiding, the media must be carefully selected according to the needs originating in the contents to be demonstrated and to the particular characteristics of the stakeholders. In our conceptualization, two complementing presentation forms should be included in each and every prototype, namely *narration* and *enactment*. The narration compiled from information that enables the passing on of the story and impacts of the process best, while enactment makes it visible and hearable how the process is actually happening (the dramaturgy of the happenings). The enactment should cover what cannot be included explicitly in the narration, and complement what can just partially be included. Thus, narration reflects an external perspective, while enactment reflects an internal one view.

4. SOME METHODOLOGICAL ISSUES OF ABSTRACT PROTOTYPING

At large, abstract prototyping is not else than instantiating the above specified information constructs using reliable data and expressing the aggregated contents by means of narrations and enactment modalities. There is an inherent plurality associated with abstract prototyping, which means that multiple processes may be instantiated in practice based on alternative operation and interaction scenarios, and more than one AP can be developed for each process. In addition to the variable contents, alternative narrations and enactment modalities can be considered. This offers enormous flexibility for abstract prototyping, but it also increased the need for insightful decisions to optimize the end results, and rigorous (systematic and computer supported) conduct in order to cope with the inherent complexity. This calls for a structured process, which is supposed to be provided by an application-independent abstract prototyping.

Systematic abstract prototyping is a task for knowledge engineers, or those designers who are specialists in abstract prototyping. From a methodological point of view it is assumed that at the beginning of the AP development process, the conceptualization of the ASC have been completed by the responsible designers and engineers, and the results are available for abstract prototyping. Complex industrial cases require a strong cooperation of, and an intense information exchange among the ASC designers and engineers, unless all tasks are put on the shoulder of product designers. The prototyping process starts with the investigation of the ASC concepts selected for demonstration, with the goals of making the specifications complete and clarifying the context of prototyping. The proposed process of abstract prototyping is described as a workflow diagram in Figure 3.

The abstract prototyping process decomposes to four phases. Phase 1 is dedicated to requirement engineering and concept development. Requirement engineering involves the exploration and processing the stance and demands of the stakeholders against the demonstrated process and the form of delivery, as well as the concrete design requirements for the implementation of APs. The first phase also identifies prototyping opportunities based on the conceptualization of the operation and/or interaction process and conceptualization of the contents and the presentation of the AP. The former is a task for the development of ASCs, the latter

Phase 2 concentrates on the contents development for the AP. It includes three interwoven sub-processes, namely: (i) conceptualization and specification of the end-users as persona, (ii) modeling and specification of the artifact-service combination as a functional and structural system, and (iii) modeling and specification of the environment in which the operation of the ASC and the end-user

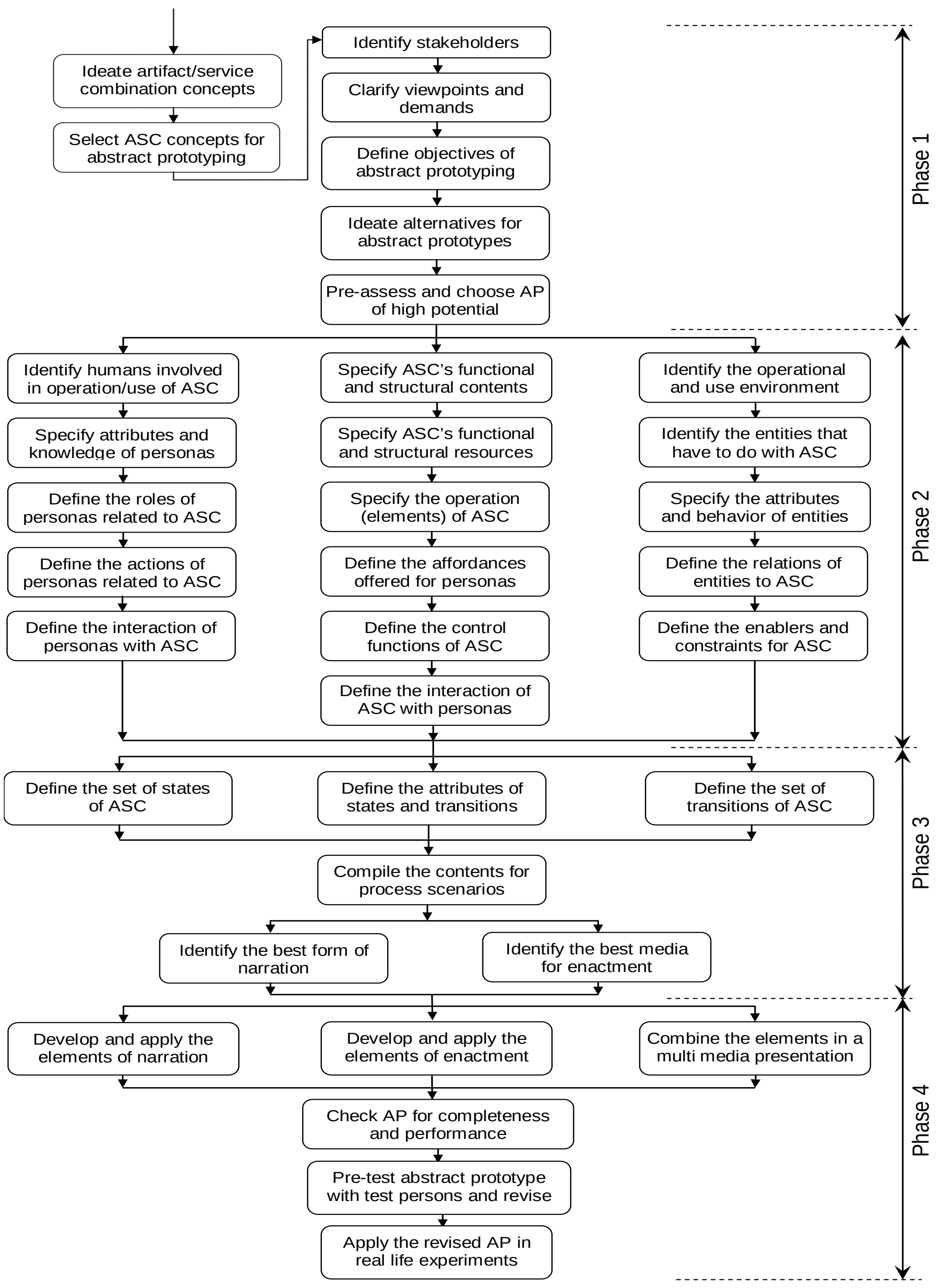
Ideate artifact/service combination concepts
Select ASC concepts for abstract prototyping
Identify stakeholders
Clarify viewpoints and demands
Define objectives of abstract prototyping
Ideate alternatives for abstract prototypes
Pre-assess and choose AP of high potential
Phase 1
Identify humans involved in operation/use of ASC
Specify attributes and knowledge of personas
Define the roles of personas related to ASC
Define the actions of personas related to ASC
Define the interaction of personas with ASC
Specify ASC's functional and structural contents
Specify ASC's functional and structural resources
Specify the operation (elements) of ASC
Define the affordances offered for personas
Define the control functions of ASC
Define the interaction of ASC with personas
Identify the operational and use environment
Identify the entities that have to do with ASC
Specify the attributes and behavior of entities
Define the relations of entities to ASC
Define the enablers and constraints for ASC
Phase 2
Define the set of states of ASC
Define the attributes of states and transitions
Define the set of transitions of ASC
Compile the contents for process scenarios
Identify the best form of narration
Identify the best media for enactment
Phase 3
Develop and apply the elements of narration
Develop and apply the elements of enactment
Combine the elements in a multi media presentation
Check AP for completeness and performance
Pre-test abstract prototype with test persons and revise
Apply the revised AP in real life experiments
Phase 4

Figure 3. The workflow of abstract prototyping

interaction happens. Depending on the objectives, these specifications can be low-fidelity or high-fidelity representations of the end-users, the ASC, and the environment. The levels of specification of the three above constituents need to be in harmony. This implies that the complexity and the challenges of abstract prototyping grow exponentially with the increase of the required fidelity of representation.

Phase 3 sets up a complete scenario of system operation, human actions, human-system interactions, and environment effects. The state-transition based representation offers an easy to alter representation for a logical specification of the flow of the process. A scenario is complete if it ensures that the specified operations and interactions produce all states and transitions that are required to realize and demonstrate the planned process successfully, but without any surplus. Obviously, there are no general, explicit, and formal sufficiency conditions. The major qualitative criterion is that the rapid prototype, more precisely, its demonstration and assessment should provide sufficient feedback for optimization of the concept of ASC and for the enhancement of the interaction and experience of the end users. The higher the fidelity of prototypes, the reliable the feedback can be. If high-fidelity abstract prototyping is carefully done in the necessary iterations, the end result is the precise specification of the final system. In phase 3, decisions are made on the best forms of the elements of narrations and the best media for visualization of the elements of the enactment.

Phase 4 deals with the design, media-enabled implementation, recording and integration of the elements of the narration and enactment into a complete demonstration material. In principle, narration and enactment can be applied apart (unconnected), or together (in concert), depending on the needs. In practice, many different combinations and mixes can be employed, depending on the process to be presented, the objectives, the available assets, and the constraints of abstract prototyping. As methodological issue, it is important to note that abstract prototyping should obey the principle of parsimony, that is, it should achieve a trade-off in terms of the investments and the return on the investments (feedback or approval). In practice it means that the media representation of the AP should convey all necessary pieces of information about the process using the most appropriate media form, but not more than that is sufficient. The appropriateness of the contents and the implementation of the abstract prototype need to be tested by carefully selected test persons, before it is used in ‘field’ tests. This is in particularly important because the impression made by the presentation quality of the AP is a confounding variable from the aspects of judging of the contents of the prototyped process, and this interplay should be minimized.

Finally, let us have a look at the possible forms of demonstration sessions, and comment on their benefits and deficits. The presentation of APs for stakeholders in demonstration sessions may happen in: (i) reflexive, (ii) interactive, and (iii) constructive forms. In a *reflexive demonstration*, the APs (i.e. the narration and the enactment is presented to the stakeholders without interruption, and the processing of the presented knowledge happens afterwards. There are two advantages of this form, namely, that (i) it guarantees that the whole of the AP is presented in the timeframe allocated for the demonstrations, and (ii) the makes it possible to separate the presentation and the assessment in space and time. A shortcoming is that it does not support the formation of the shared awareness through immediate reactions, though a high level of shared understanding can be achieved by a well-constructed abstract prototyping. An *interactive demonstration* allows the stakeholders to make comments and ask questions at any moment in the course of presentation. This requires a stoppable and resumeable AP design, as well as strong moderation. The interaction supports the rapid formation of the shared awareness among the designers and the stakeholders. This may lead to a better assessment of the proposed ASC on the side of the stakeholders, and collecting more information for enhancements on the side of the designers. This form of presentation does not support real time changes in the contents of the demonstrated AP. This is however an explicit goal of a *constructive demonstration* session. Modifications can be introduced by substituting certain modules of the modularized prototype by pre-prepared modules, or by providing computer based means for a real-time and fast modification or regeneration of certain information constructs according to the requests and advices of the stakeholders. The advantage of this approach is that the creative interaction may result in appropriate and innovative solutions. The pitfalls are that it: (i) requires higher level involvement of the stakeholders, (ii) significantly extents the time of the demonstration and assessment, (iii) the procedure may be hanged on by lack of information, and (iv) requires sophisticated AP editing tools.

5 CONCLUDING REMARKS

Physical prototyping and virtual prototyping are widely used for pre-production modeling and testing of products. Abstract prototyping has not received sufficient attention in domains of product development, other than software design and engineering. Nowadays, there is a growing need for this novel form of computer support in the field of development of artifact-service combinations. The understanding has been that the return of the investments in abstract prototyping should be expected in the downstream phases and in the better quality of the ASC, rather than saving time, costs and efforts in the ideation and conceptualization phases. Our current research endeavors the establishment of an underpinning theory and application-independent generic methodology for abstract prototyping of ASCs. Towards this end, we reinterpreted certain concepts, determined the major players, provided formal specifications, and worked out a practical implementation and application process flow. A detailed workflow of abstract prototyping is presented and some issues related to an optimum implementation of the methodology are discussed. The proposed methodology have been applied in various graduation projects, its full scale industrial validation has not yet been made. Actually, we are at this moment involved in an industrial project in which the proposed abstract prototyping technology is applied to develop computer support for digital shoe design, effective shoe sample making, and sale activities. Nevertheless, our first results encourage us to claim that abstract prototyping offers advantages in conceptualization of solutions, as well as in demonstration and assessment of concepts with the involvement of stakeholders. In addition abstract prototypes can also be used as an effective research means in design inclusive research.

In software engineering, prototyping has in the last two decades been widely used beyond requirement testing. There have been many specific techniques proposed to enable evaluation of functions, processes, resources via user and developer interaction specified by operational scenarios. The generic methodology of abstract prototyping offers means for capturing, specifying and demonstrating both the assumed operation sub-processes of ASCs and the foreseeable human interaction (use) sub-processes. In general, the major benefits of abstract prototyping are that it influences the most creative phases of ASC development and opens the way towards intelligence that cannot be obtained otherwise. The major deficits are that it needs not only extra efforts and knowledge, but also a comprehensive thinking and a systematic way of working. Considering the needed extra efforts and resources, the vagueness and incompleteness of knowledge about implementation and realization, and the possible concerns about the trade-off of the investments and the professional gains, industrial ASC developers need more evidence about the effectiveness and usefulness of abstract prototyping. Our follow up research will focus on providing practical evidence for these, and on the development of an effective abstract prototyping software tool.

REFERENCES

[1] Hardgrave, B. C. When to prototype: Decision variables used in industry, *Information and Software Technology*, 1995, 37(2), 113-118.

[2] Chua, C. K. Three-dimensional rapid prototyping technologies and key development areas, *Computing and Control Engineering Journal*, 1994, 5(4), 200-206.

[3] Wang, G. G. Definition and review of virtual prototyping, *Journal of Computers and Information Science in Engineering*, 2002, 2(3), 232–236.

[4] Horváth, I., Gerritsen, B. and Rusák, Z. A new look at virtual engineering, in: *Proceedings of Mechanical Engineering Conference*, Budapest, 2010, 1-10.

[5] Opiyo, E. Z., Facilitating the development of design support software by abstract prototyping, *Ph.D. Thesis*, Delft University of Technology, Delft, 2003, 1-206.

[6] Gielingh W., Cognitive product development: A method for continuous improvement of products and processes, in: *Proceedings of the Seventh International Symposium on Tools and Methods of Competitive Engineering*, Vol. 1, Delft University of Technology, Delft, 2008, 25-37.

[7] Wood, D. P. and Kang, K. C., A classification and bibliography of software prototyping, *Technical Report CMU/SEI-92-TR-13*, ESD-92-TR-013, Software Engineering Institute, Carnegie Mellon University, 1992, 1-86.

[8] Rich, C. and Feldman Y. A., Seven layers of knowledge representation and reasoning in support of software development, *IEEE Transaction on Software Engineering*, 1992, 18(6), 451-469.

[9] Hickey, A. and Dean, D. Prototyping for requirements elicitation and validation: A participative prototype evaluation methodology, in: *Proceedings of Americas Conference on Information Systems (AMCIS 1998)*, 1998, 797-800.

[10] Cooprider, J. G. and Henderson, J. C. Technology-process fit: Perspectives on achieving prototyping effectiveness, *Journal of Management Information Systems*, 1991, **7**(3), 67-87.

[11] Constantine, L. and Lockwood, L. Software for use: A practical guide to the models and methods of usage-centered design, Addison-Wesley, Reading, 1999, 1-12.

[12] Chalasani, P., Jha, S. and Sullivan, K. The options approach to software prototyping decisions, *Technical Report CMU-CS-97-161*, SCS, Carnegie Mellon University, 1977, 1-27.

[13] Shao, G. and Hanna, W. Soft prototyping in the design of military electronics, in *Proceedings of the IEEE National Aerospace and Electronics Conference '90*; Dayton, OH, 1990, 729-735.

[14] Karat, J. and Karat, C. M. The evolution of user-centered focus in the human computer interaction field, *IBM Systems Journal*, 2003, 42(4), 532-541.

[15] Phalp, K. and Counsell, S. Coupling trends in industrial prototyping roles: An empirical investigation, *Software Quality Journal*, 2001, 9(4), 223-240.

[16] Kordon F. and Luqi, X. An introduction to rapid system prototyping, *IEEE Transactions on*

Software Engineering, 2002, 28(9), pp. 817–821.

[17] Vasconcelos, W., Robertson, D., Sierra, C., Esteva, M., Sabater, J. and Wooldridge, M. Rapid prototyping of large multi-agent systems through logic programming, Kluwer, Amsterdam, 2003.

[18] Buchenrieder, K. Rapid prototyping of embedded hardware/software systems, *Design Automation for Embedded Systems*, 2002, 5, 215-221.

[19] Ham, D.-H., Wong, W. and Amaldi, P. Comparison of three methods for analyzing human work - in terms of design approaches, in: *Proceedings of the Workshop Describing Users in Context - Perspectives on Human-Work Interaction - INTERACT 2005,* Rome, Italy, 2005, 7-11.

[20] Reichart, D., Forbrig, P., Dittmar, A. Task models as basis for requirements engineering and software execution, in: *Proceedings of Task Models and Diagrams for User Interface Design Conference*, 2004, pp. 51-58.

[21] Constantine, L. Canonical abstract prototypes for abstract visual and interaction design, in: *Proceedings of 10th International Conference on Design, Specification and Verification of Interactive Systems*, LNCS 2844, Springer-Verlag, Berlin, 2003, pp. 1-15.

[22] Constantine, L., Windl, H., Noble, J. and Lockwood, L. From abstraction to realization: Canonical abstract prototypes for user interface design, *Working Paper, Version 2.0*, Constantine & Lockwood, Ltd., 2003, 1-12.

[23] Campos, P. F. and Nunes, N. J. CanonSketch: A user-centered tool for canonical abstract prototyping, in: *Proceedings of EHCI-DSVIS 2004*, LNCS 3425, 2005, pp. 146-163.

[24] Nóbrega, L., Nunes, N. J. and Coelho, H. DialogSketch: Dynamics of the canonical prototypes, in: *Proceedings of Task Models and Diagrams for User Interface Design Conference,* Gdansk, Poland, 2005, 26-27.

[25] Mikkelson, N. and Lee, W. O. Incorporating user archetypes into scenarios-based design, in: *Proceedings of the UPA 2000 - Designing the New Millennium Today*, 2000, Asheville, N.C.

[26] Nielsen, L. From user to character: An investigation into user-descriptions in scenarios, in: *Proceedings of the 4th Conference on Designing Interactive Systems*, London, 2002, 99-104.

[27] da Cruz, A. M. R. and Faria, J. P. Automatic generation of user interface models and prototypes from domain and use case models, in *User Interfaces*, Intech, Vukovar, 2010, 35-60.

[28] Liu, Y. H., Osvalder A.-L. and Karlsson, M. A. Considering the importance of user profiles in interface design, in *User Interfaces*, Intech, Vukovar, Croatia, 2010, 61-80.

[29] Mikovec, Z., Maly, I., Slavik, P. and Curin, J. Visualization of users' activities in a specific environment, in: *Proceedings of Simulation Conference*, Washington, DC, 2007, 738-746.

[30] Clerckx, T., Winters, F. and Coninx, K. Tool support for designing context sensitive user interfaces using a model-based approach, in: *Proceedings of Task Models and Diagrams for User Interface Design Conference*, Gdansk, Poland, 2005, 11-18.

[31] Yu, L. Prototyping, domain specific language, and testing, *Engineering Letters*, 2008, 16(1), 1-6.

[32] Nilsson, E. G., Flocha, J., Hallsteinsena, S. and Stava, E. Model-based user interface adaptation, *Computers & Graphics*, 2006, 30(5), 692-701.

[33] Memmel, T. and Reiterer, H. Model-based and prototyping-driven user interface specification to support collaboration and creativity, *J. of Universal Computer Science*, 2008, 14(19), 3217-3235.

Contact: Prof. Dr. Imre Horváth
Faculty of Industrial Design Engineering/ Delft University of Technology
Department of Design Engineering
Landbergstraat 15, 2628 CE, Delft
the Netherlands
Phone: 31 15 278 3520
Fax: 31 15 278 1839
E-mail: i.horvath@tudelft.nl

Imre Horváth is full professor of Computer Aided Design and Engineering at the Faculty of Industrial Design Engineering of the Delft University of Technology, the Netherlands. He received his Ph.D. title from the TU Budapest (1994), and C.D.Sc. title from the Hungarian Academy of Sciences (1993). He also obtained two honorary doctor titles. He initiated the International TMCE Symposia, was chair of the CIE Division of ASME, and is co-editor in chief of Journal CAD.