UCProMo—Towards a User-Centred Process Model

Tom Gross

Human-Computer Interaction Group, University of Bamberg, Germany
(<firstname.lastname>(at)uni-bamberg.de)

Abstract. The field of Software Engineering has a long tradition of developing sophisticated process models and methods and tools for its support. At the same time in the field of Human-Computer Interaction process models, methods, and tools have been developed and standardised internationally. Approaches from both fields have a lot to offer. However, despite great approaches for joining strengths and advantages of both fields, synergies are not yet fully used. In this paper I present the *UCProMo* User-Centred Process Model that provides an integrated approach by leveraging on existing process models, methods, and tools from both fields. *UCProMo* capitalises on clear phases, iteration, and strong involvement and participation of users throughout the whole process, which leads to integrated results and models of technology (esp. software) and ultimately to smooth user journeys through the whole system.

Keywords. Software Engineering; Human-Computer Interaction; Process Model; Methods and Tools.

1 Introduction

In any kind of endeavour to design and develop systems, a structured approach is indispensable. This particularly applies to Software Engineering (SE) and Human-Computer Interaction (HCI). Process models support a structured approach by suggesting process phases and the order in which those phases should be gone through.

In SE over the last decades many great process models have been presented. The traditional waterfall model already provided a list of steps [18]. Later, Boehm published the famous 'Spiral Model of Software Development and Enhancement' in [3]. It suggests to go through the steps in a spiral from inside out and to continually expand the results of each phase in each circle. The Unified Process [14] has been a big leap and seen many variations and refinements. Many other process models contributed to a heterogeneous landscape of process models.

In HCI a parallel emergence and evolution of process models could be witnessed. These models have many similarities with those in SE. Yet, two distinctions are that in general in HCI the involvement of users throughout the whole process played a central role, and the evaluation of the results with users had a high priority. For instance, the 'Star Life Cycle' of Hartson & Dix [10] suggested that from any phase there should be a connection to an evaluation phase that is in the heart of the process model. The diversity of process models within HCI, eventually led to a standard pro-

cess model for the 'Human-Centred Design of Interactive Systems' recommended by the International Standardisation Organisation (ISO) in the ISO 9241-210 [13].

Great contributions have been made towards combining approaches from SE and HCI. Most prominently, Usage-Centred Design is based on the idea to use 'abstract models to solve concrete problems' [8, p. 26]. It combines the HCI perspective of an early focus on users, their tasks, and their contexts with the SE paradigm of a strong focus on clear abstract models for analysis and design. Later, Activity Theory was integrated into the Usage-Centred Design model to become the Human Activity Modelling approach with better representations of human use of tools and artefacts [6]. Nunes picked up the strong orientation of actual usage or use and suggested a use-case-driven software development approach to combine SE and HCI [17].

Despite such great approaches for joining strengths and advantages of both fields, the potential for synergies is not yet fully used. Clearly both communities—the SE and the HCI—have reached out mutually. For instance, agile approaches put a strong focus on users and early on put a priority to user stories and user interfaces, etc. [1]. Yet, a challenge that remains is that some basic paradigms in SE and HCI are not commensurable. For instance, by and large in SE the perspective is abstract that leads to a great overview by focusing on the fundamental structure and behaviour of the overall system, and in HCI it is concrete that gives a detailed impression of the user interface by early focusing on the user experience. These perspectives shine through—even in combinations such as Usage-Centred Design there is a clear priority for models over interfaces; the authors call it a 'model-driven approach' [8, p. 42]. Other approaches such as the Human Activity Modelling offer a compromise of perspectives, but at the price of losing the original expressive power of both sides (i.e., high abstraction in SE; high concreteness in HCI).

In this paper I present the *UCProMo* User-Centred Process Model that provides an integrated approach by leveraging on existing process models from SE and HCI. Using the *UCProMo* model is easy and straightforward—designers and developers individually or in teams just need basic knowledge and experience in either field. Overall the approach follows the requirements for light, agile, and lean development published very recently in [15]. In the next section I discuss the background and related work of process models in SE, HCI, and beyond. Then I present the *UCProMo* User-Centred Process Model with its generic method-agnostic processes. A discussion and conclusions summarise the contributions and glance at future work.

2 Background and Related Work

Three categories of process models are relevant to our approach—process models from SE, from HCI, and combinations.

2.1 Process Models in SE

The field of SE has a long tradition of sophisticated process models and methods and tools for its support. Sommerville explains: 'the systematic approach that is used in software engineering is sometimes called a software process. A software process is a

sequence of activities that leads to the production of a software product.' [19, p. 9]. And he continues: 'a software process model is a simplified representation of a software process. […] These generic models … are abstractions of the process that can be used to explain different approaches to software development.' [19, p. 28].

The waterfall model and the spiral modal are important early predecessors. The waterfall provided a detailed list of steps everybody should follow: system requirements; software requirements; analysis; program design; coding; testing; operation. It foresaw small iterations [18]. Its fundamental contribution was to lay out basic steps that are still relevant today. Later, Boehm published the 'Spiral Model of Software Development and Enhancement' [3]. It suggests to go through the steps in a spiral—inside out—and to continually expand the results of each phase in each cycle. The very important take away message—that is still important today—is to iterate and especially to continually re-evaluate the results.

More recently, the Unified Process was suggested as a 'set of activities needed to transform a user's requirements into a software system' [14, p. 4]. It is use-case driven (i.e., it departs from users and functionality for them); architecture-centric (i.e., all static and dynamic aspects of the system to be built); and iterative-incremental (i.e., it 'divides the work into smaller slices or mini-projects.' [14, p. 7]). Each cycle has four phases: inception (i.e., development of ideas), elaboration (i.e., specification of use cases and design of system architecture, construction (i.e., development of the system), and transition (i.e., movement from development via first beta-tests towards deployment). Orthogonal to the phases the Unified Process defines five core workflows. Requirements mainly fall into inception and elaboration; analysis mainly into elaboration; design between elaboration and construction; implementation into construction; and test between construction and transition [14]. The Unified Process was probably the biggest leap towards systematically including users and users' needs and requirements. Since then many variations and refinements were suggested—a very wide-spread being the Rational Unified Process by Kruchten [16].

2.2 Process Models in HCI

In HCI many process models have been suggested. Despite the fact that the basic goal and also some basic steps are the same as in SE there are quite some differences.

For many years the HCI community has been using a standard process model with the title 'Human-Centred Design of Interactive Systems'. It is now part of the ISO 9241 on Ergonomics of Human-System Interaction in the part ISO 9241-210 Human-Centred Design Processes for Interactive Systems [13] (formerly it was published in ISO 13407:1999 [12]). Its processes are: identification of the need for human-centred design; understanding and specification of the context of use; specification of the user and organisational requirements; production of the design solutions; and evaluation of the design against the requirements.

Also Unified Reference Frameworks have been developed to facilitate the process of developing user-centred systems by abstracting from hardware properties in abstract user interfaces [4]. And, Contextual Design offers a process model that has a strong focus on understanding users activities and requirements in the context where the users are using the system [11].

2.3 Process Models that Combine SE and HCI

Out of the approaches that combine process models from SE and HCI the Usage-Centred Design and the Human Activity Design have been most influential to our approach.

The Usage-Centred Design (UCD) draws from the Unified Process and combines it with principles from HCI. Like the Unified Process it is based on models; it uses 'abstract models to solve concrete problems' [8, p. 26]. Whereas the Unified Process suggests models that roughly correspond to its core workflows (i.e., a use-case model; an analysis model; a design model; a deployment model; an implementation model; and a test model), the UCD has three simple models at its core: the role model representing the relationships between users and the system; the task model showing the structure of the tasks that users need to perform; and the content model laying out the functionality of the user interface. Through the focus on these three principal models UCD aims to move away from an early focus on concrete users and concrete user interface designs that often prevail in HCI.

In the later Human Activity Modelling (HAM) [6] Constantine extended his UCD with Activity Theory. The cornerstones are activities, which are basically seen as a collective endeavour in which a community of participants transforms a material into an object. This community of participants uses tools and applies rules and division of labour to organise itself. HAM has three principal models: the activity context model that did not exist in UCD represents human activities; the participation model is an adaptation of the role model and describes user roles, yet now including the context of the activities in which they occur; and the performance model is based on the previous task model and contains user actions targeted at either other users or artefacts.

2.4 Summary of Background and Related Work

Overall the gap between both fields has not been fully bridged. As we have seen—despite the great progress in process models in SE and in HCI as well as stimulating combinations of SE and HCI approaches in the UCD and HAM—an integrated approach that leverages on the expressive power of both SE and HCI and can be flexibly applied by designers and developers of software with any knowledge and experience is still missing.

The related work also shows that some terms are not used consistently, which can be misleading—especially with respect to clearly distinguishing users and developers. For instance, as we have seen in the quotes above, the term *activity* has been used in the literature to refer to both, the things that developers are doing to develop concepts and systems and the things that users are doing with the system. In order to disambiguate terms this paper uses the following: a process refers to the whole endeavour of developing a system from the beginning to the end and independently of the path that is taken. A phase refers to a distinct and significant part of the process. Iteration refers to one cycle of steps that can be repeated eventually. The terms task, activity, and action are only used for user interaction with the system.

3 The *UCProMo* User-Centred Process Model

In the this section I present the *UCProMo* User-Centred Process Model with its generic method-agnostic phases. The related work above provides great stimuli for our process model. It leads to the following requirements for our process model that can be seen as an aggregated summary of the different advantages and strengths:

- *Phases* should be clearly defined and have definite beginnings and endings while at the same time allow flexible coupling, feedback and feedforward to other phases for an iterative as well as incremental process.
- *Abstract* modelling that allows keeping the complete system in focus should be combined with *concrete* users, user requirements and needs, and designs.
- *Heterogeneous* approaches and results throughout analysis and exploration, specification, design and development, and testing should be supported.
- There should be a clear paradigm of *analysis* (i.e., modelling the status quo) on the one hand and *design* (i.e., modelling the future system) on the other. At the same time analysis and design should go hand in hand; and appropriate redesign should always be possible (i.e., this is in contrast to UCD and HAM where tasks are primarily analysed and modelled rather than (re-)designed).

Subsequently I introduce the core phases of the *UCProMo* User-Centred Process Model.

3.1 Plan the Human-Centred Design Process

Before the actual phases of the human-centred design process (HCD) can start, all parts of the project need to be planned and time and resources need to be allocated. This can be seen as phase zero of the process. At the beginning it should be clarified how usability is addressed throughout the whole process. The ISO 9241-210 recommends: to analyse 'how usability relates to the purpose and use of the product, system or service (e.g., size, number of users, relationship with other systems, safety or health issues, accessibility, specialist application, extreme environments); and to estimate how bad usability might negatively influence the project by analysing 'the levels of the various types of risk that might result from poor usability (e.g., financial, poor product differentiation, safety, required level of usability, acceptance)'; and finally, to be clear about the general conditions of the project in the sense of the 'nature of the development environment (e.g., size of project, time to market, range of technologies, internal or external project, type of contract)' [13, p. 8].

3.2 Understand and Define Users, Tasks, and Contexts

After the project planning the first real phase aims at understanding and specifying users, tasks, and contexts. The best way to do that is to go through the following steps: produce an inventory of all items; describe a profile of the most central characteristics for each item; and chart a map of the structure and relationships among all items.

The *user model* consists of the user inventory; user profiles; and a user map.

The *user inventory* contains the essential roles (e.g., author of a book), and role characteristics (e.g., expectations, responsibilities), as well as the essential user characteristics that have an influence on how they play their role (e.g., knowledge, skills, experience). For the *user profiles* it is advisable to identify permutations of common essential user roles and user characteristics and generate profile descriptions for them (e.g., author of a book with limited technical knowledge). The *user map* is a chart consisting of a node as a standardised labelled icon for each individual user profile and links as lines representing connections between them. In the basic form simple links are used, if needed, links can have types and directions to represent specific relationships among users (e.g., a hierarchy). If more semantics are preferable, further details can be added to the nodes representing central characteristics visually (i.e., an active role which actively participates vs. a focal role which is mandatory vs. a passive role of audience who passively participates). Fig. 1 shows a simple example of a user map.

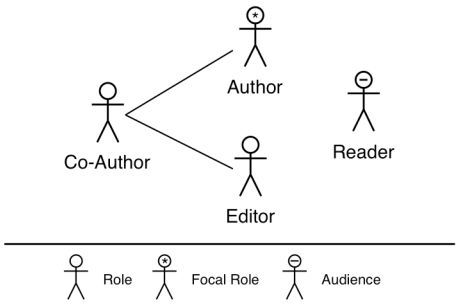

Fig. 1. *UCProMo* User Map example.

The *task model* consists of a task inventory; task profiles; and a task map. The *task inventory* is a collection of all essential tasks, where each task consists of events and processes that are clustered together and have a logical sequence (e.g., invite co-author for writing book together). Very often tasks are nested and a hierarchical decomposition helps for gaining a better understanding. Tasks are comparable to use-cases in SE, and to scenarios in HCI in that they also represent and structure the users' activities. Each *task profile* contains a structured description of a sequence of user activities that is free of technical details. The *task map* is—like the user map—a chart that puts the essential individual tasks into perspective and in relation to each other. Since for large systems task maps can get quite complex, it is very important from the beginning to focus on essential tasks that are of vital interest to the users as well as the project team. In analogy to user maps, in the simplest form, the task map provides a simple, yet informative, overview containing a node as a standardised labelled box for

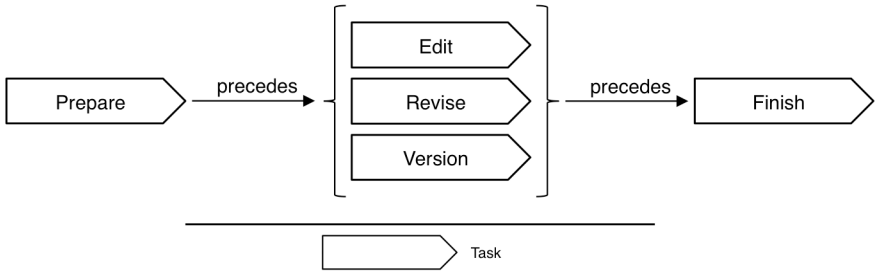

Fig. 2. *UCProMo* Task Map example.

each task and links as lines showing connections between them. To add more semantics links can be typed (e.g., showing temporality, specialisation, extension, or composition). Fig. 2 shows a simple task map.

The *context model* consists of a context inventory; context profiles; and a context map. Here a context is defined as: 'the interrelated (i.e. some kind of continuity in the broadest sense) conditions (i.e. circumstances such as time and location) in which something (e.g. a user, a group, an artefact) exists (e.g. presence of a user) or occurs (e.g. an action performed by a human or machine)' [9, p. 286]. The *context inventory* brings together all contexts in which users perform their tasks. Furthermore, mobile use needs to be considered when analysing the context. The *context profiles* should for each context or trajectory identify all information relevant to the user performing the respective task. A profile should include the technical (e.g., hardware, software, network connectivity), the physical (e.g., noise, thermal conditions, vibration, space and furniture), the organisational (e.g., work practices, assistance, interruptions), and the social environment (e.g., other persons in the same room). The *context map*—analogous to the user map and the task map—provides a visual overview of all contexts and their relations. It shows individual contexts as nodes in labelled boxes and links between contexts as simple lines. Again, in the basic form the context map includes all contexts and their connections; in more detailed versions the links represent the relationships between contexts—contexts can have temporal relations (e.g., followed-by) and can be nested (e.g., contains vs. part-of).

3.3 Specify System Requirements

This phase also defines a core model—the integration model. Despite the similarities and overlaps with the models that define users, tasks, and contexts there is one essential difference regarding the attitude with which the model is created in this phase: whereas in the previous phase the models have pure analytical purposes and document the state-of-the art, the model of this phase is design-oriented and anticipates, specifies, and defines future aspects of the system and related issues.

The *integration model* provides a hierarchical description of the task that users can perform with the future system, where activities are interactions with the system towards solving specific problems and with a purpose. Activities are composed of actions, and actions are composed of operations. For instance, an activity could be to write an email, where a specific action could be to add a recipient, which is done operationally by selecting an entry from the address-book and adding it to the 'To:' field of the email program. The integration model consists of integration profiles and an integration map; and it is complemented by the performance map. The *integration profiles* specify the design of future activities the system should support and aim to inform interaction design. They consist of four parts: *purpose* describes the motive and objective of the respective activity; *place and time* describe the context of the activity in terms of time and location it takes place; *participation* describes the user roles (and characteristics) involved in the activity; and *performance* provides details how the activity is performed. The *integration map* is a complex chart that not only builds on and integrates the user map, task map, and context map from the previous phase, but also moves from a presentation of the state-of-the-art to an anticipation and

specification of the future system. It consists of different categories of nodes representing users with activity levels, roles, centrality; and tasks that are clustered into contexts. Fig. 3 shows an example of an integration map (please note that the symbols for boundary, control, and entity class resemble to the extensions of the graphical notation of UML by [14, p. 439]).

The *performance map* goes beyond the task map and is also design-oriented rather than analysis-oriented. In the simple version the performance map includes nodes as standardised labelled boxes representing activities and links as untyped connections between the nodes. The basic model can be extended by tasks—so for each decomposable activity all contained tasks are drawn into the model. This provides more

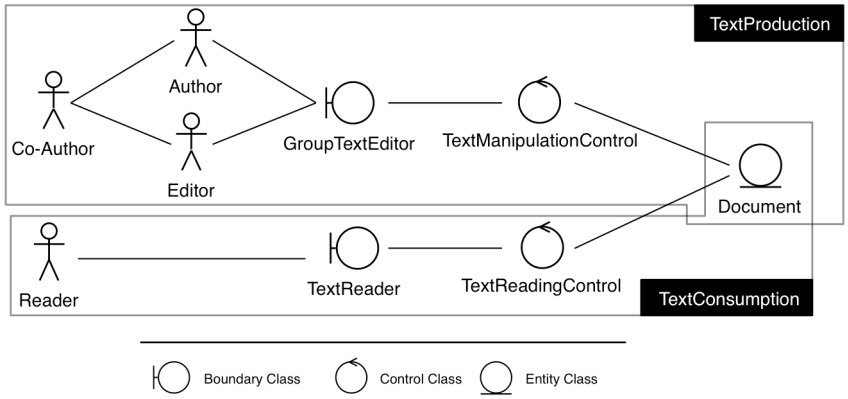

Fig. 3. *UCProMo* Integration Map example with a text production context (top) and a text consumption context (bottom).

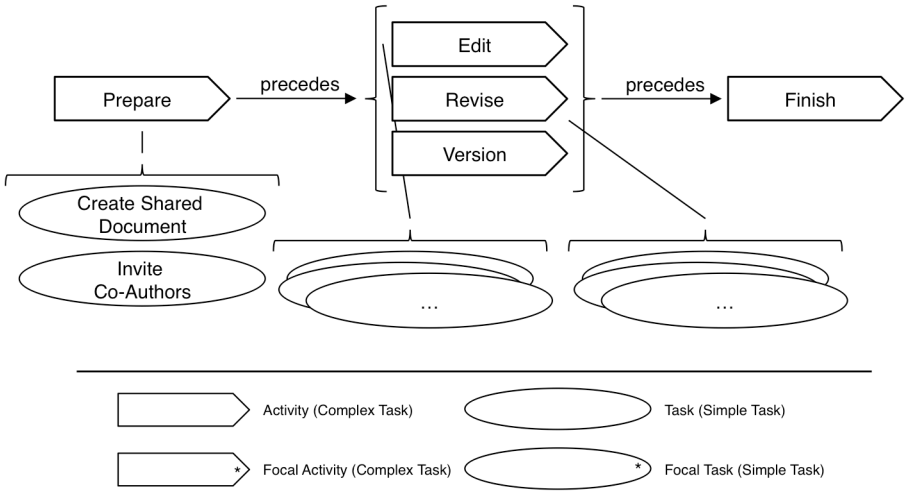

Fig. 4. *UCProMo* Performance Map example.

information on the users' interaction with the system. Fig. 4 shows a generic example of a performance map.

3.4 Design User Tasks, and User Interactions

The tasks designs and interactions designs should—given they were carefully specified—logically follow from the previous models. Theoretically task designs describe how the users will accomplish their tasks with the system, whereas interaction designs illustrate how the tasks will exactly be performed with the future system. With the aim of remaining generic in the *UCProMo* process model (i.e., not diving into concrete screen designs, etc.) the two perspectives are combined into one unified *interaction space model*. This model describes the interaction between the users and the system in the form of summaries of the abstract path the users can take through the system. It consists of interaction space profiles, and an interaction space map.

The *interaction space profiles* contain abstract, yet detailed, information on individual interfaces in terms of its information contents and interaction components for user input. It is important to note that the interaction space profiles initially do not need to have any visual representation (e.g., showing the proportions of the different parts of the user interface). Interaction space profiles resemble essential use cases of Usage-Centred Design [7]. However, approaches such as Usage-Centred Design [7] and Contextual Design [11] often proceed in a bottom-up manner—that is, depart from individual cases and aggregate them. The *UCProMo* suggests a hybrid approach, where the interaction space profiles and map are developed in sync having the user journey or customer journey in mind. This is important for many reasons—such as for consistency in similar interaction types among individual profiles.

The *interaction space map* has nodes as standardised labelled boxes for each interaction space as well as links as lines representing connections of interaction spaces. The connections between the interaction spaces are navigation paths that the users can follow when using the system. This map provides a general overview of the interaction space landscape, and additionally serves as a tool to judge and optimise the breadth and depth of the user interaction. In fact, when designing the interaction space model there is a trade-off between having simple interaction spaces with few elements (and consequently a high number of interaction spaces to cover the whole functionality) and having complex interaction spaces with many elements (and consequently fewer interaction spaces and less navigation effort for the users). Fig. 5 shows an excerpt of an interaction space map.

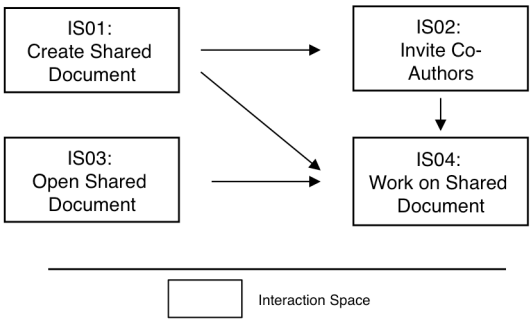

Fig. 5. *UCProMo* Interaction Space Map excerpt.

3.5 Develop the System

The actual implementation and test of the system are core activities in each process model—both in terms of their importance for the overall success of the project, and in terms of the money, time, and other resources spent in this phase in comparison to the other phases.

Still, the actual implementation is in many process models only briefly covered. This probably has several reasons, one of which being that it is a rather practical endeavour and a completely different terrain. As Jacobson et al. write in their book on 'The Unified Software Development Process' in the introduction to their chapter on 'Implementation': 'Fortunately, most of the system's architecture is captured during design. The primary purpose of implementation is to flesh out the architecture and the system as a whole.' [14, p. 267].

3.6 Evaluate the System

The evaluation of the system from a HCD perspective in general (besides expert evaluations and simulations) involves direct contact with users—typically presenting them some results and getting feedback. These results do not only refer to the final product, but also to any result that is generated throughout the process—particularly including the different models that can and should be verified with users.

ISO has clear recommendations on how evaluations should be done. They should include adequate allocation of resources to evaluation; early planning of evaluation; enough testing and analysing of the results and eventually prioritising the reactions triggered by the results; and appropriate communication with all stakeholders involved [13].

As a matter of fact user evaluation is also a vital part of the overall software testing that is very important for any kind of software (and hardware) project. From this perspective the software test has two goals [19]: to show that the software successfully fulfils all requirements; and to eventually find problems which can then be solved. It is important to note what Sommerville—quoting Dykstra—points out: 'Testing can only show the presence of errors, not their absence' [19, p. 206]. Likewise user evaluation can only proof the effectiveness (degree to which the users reach their goal), efficiency (effort that is required to reach the goal), and satisfaction (comfort and pleasure when using the system) of the current users, and only assume that the same holds true for future user populations.

3.7 Deploy the System

The final phase after a successful evaluation is Deploy the System. A successful evaluation can happen already in the first iteration, or in later iterations, and at least theoretically it could also be possible that it never happens but that the system is still rolled out. This phase is beyond the scope of this paper.

4 Discussion and Conclusions

In this paper I motivated the need for an integrated process model leveraging on both SE and HCI processes. I introduced the generic *UCProMo* User-Centred Process Model with its phases and models that can be easily followed and produced by designers and developers without an SE or HCI background.

UCProMo supports clearly defined phases and iterative and incremental feedforward and feedback cycles. It combines abstract modelling from SE with concrete user experience design from HCI. And it supports the whole range of activities from analysis and exploration to specification to design and development and testing. Finally, it is lean and lightweight but at the same time has built-in redundancy between analysis and design—that is, it documents the state-of-the-art in user, task, and context models for analysis; and it generates an integration model (i.e., integration map and performance map) as abstract representation of the statics and dynamics of the future system and the interaction space map as concrete design of the interaction with the future system.

It is on purpose that the interaction spaces and the interaction space map in the design phase resemble use-cases that are in many process models very early in the analysis phase. Indeed, human-centred analysis and design should not take for granted and analyse the activities as they are and build a system around them, but rather it should creatively reflect current practice and—together with the users—eventually redesign activities where appropriate. An example of theory-based creative modelling is [2], where the authors depart from a framework of social interaction from social science as input for their models.

The fact that *UCProMo* aims at rapid modelling should not be confused with other approaches with similar goals. For instance, agile modelling by Ambler has great suggestions on how to apply existing UML models and notations in a lean way [1]. The *UCProMo*, however, suggests generic models that complement existing UML models and notations.

Finally, I did not have the space to address basic principles that apply to many areas of design likewise. For instance, Cockton has suggested 'meta-principles for any design process: receptiveness, expressivity, committedness, credibility, inclusiveness, and improvability' [5, p. 2223].

While the process model leverages on fantastic input from great existing work in HCI and SE, it still would benefit from a proper validation. In the future it should be applied to human-centred software engineering projects to get feedback of designers and developers.

Acknowledgements

I would like to thank all members of the Cooperative Media Lab in Bamberg as well as the colleagues from the Madeira Interactive Technologies Institute for inspiring discussions. Thanks to the anonymous reviewers for great feedback.

References

1. Ambler, S. Agile Modelling: Effective Practices for eXtreme Programming and the Unified Process. Wiley, N.Y., 2002.
2. Beckmann, C. and Gross, T. Social Computing - Bridging the Gap Between the Social and the Technical. In Proceedings of the 6th International Conference on Social Computing and Social Media - SCSM 2014. Springer-Verlag, Heidelberg, 2014. pp. 25-36.
3. Boehm, B.W. A Spiral Model of Software Development and Enhancement. IEEE Computer 21, 5 (May 1988). pp. 61-72.
4. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3 (May 2003). pp. 291-315.
5. Cockton, G. Getting There: Six Meta-Principles and Interaction Design. In Proceedings of the Conference on Human Factors in Computing Systems - CHI 2009. ACM, N.Y., 2009. pp. 2223-2232.
6. Constantine, L.L. Human Activity Modelling: Towards a Pragmatic Integration of Activity Theory and Usage-Centred Design. In Seffah, A., Vanderdonckt, J. and Desmarais, M.C., eds. Human-Centred Software Engineering: Software Engineering Models, Patterns, and Architectures for HCI. Springer-Verlag, Heidelberg, 2009. pp. 27-51.
7. Constantine, L.L. and Lockwood, L.A.D. Software For Use: A Practical Guide to the Models and Methods of Usage-Centred Design. Addison-Wesley, Reading, MA, 1999.
8. Constantine, L.L. and Lockwood, L.A.D. Usage-Centred Engineering for Web Applications. IEEE Software 19, 2 (Mar./Apr. 2002). pp. 42-50.
9. Gross, T. and Prinz, W. Modelling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. Computer Supported Cooperative Work: The Journal of Collaborative Computing 13, 3-4 (Aug. 2004). pp. 283-303.
10. Hartson, H.R. and Hix, D. Human-Computer Interaction Development: Concepts and Systems for Its Management. ACM Computing Surveys 21, 1 (Mar. 1989). pp. 5-92.
11. Holtzblatt, K. and Beyer, H.R. Contextual Design. In Soegaard, M. and Dam, R.F., eds. The Encyclopedia of Human-Computer Interaction (2nd ed.). The Interaction Design Foundation, Aarhus, Denmark, 2016.
12. ISO. ISO 13407: 1999 - Human-Centred Design Processes for Interactive Systems. ISO - International Organisation for Standardisation/
13. ISO/IEC. ISO 9241-210:2010: Ergonomics of Human-System Interaction - Part 210: Human-Centred Design for Interactive Systems. International Organization for Standardization.
14. Jacobson, I., Booch, G. and Rumbaugh, J. The Unified Software Development Process. Addison-Wesley, Reading, MA, 1998.
15. Jacobson, I., Spence, I. and Kerr, B. Use-Case 2.0. Communications of the ACM 59, 5 (May 2016). pp. 61-69.
16. Kruchten, P.B. The Rational Unified Process: An Introduction. Addison-Wesley, N.Y., 2003.
17. Nunes, N.J. What Drives Software Development: Bridging the Gap Between Software and Usability Engineering. In Seffah, A., Vanderdonckt, J. and Desmarais, M.C., eds. Human-Centred Software Engineering: Software Engineering Models, Patterns, and Architectures for HCI. Springer-Verlag, Heidelberg, 2009. pp. 9-25.
18. Royce, W.W. Managing the Development of Large Software Systems. In Proceedings of the Ninth International Conference on Software Engineering - ICSE'87. IEEE Computer Society Press, Los Alamitos, 1987 (reprint from 1970). pp. 328-338.
19. Sommerville, I. Software Engineering 9. Pearson Education Limited, Harlow, England, 2011.