TOWARDS AN EXECUTABLE SOCIOTECHNICAL MODEL FOR PRODUCT DEVELOPMENT AND ENGINEERING SYSTEMS

Axel Hahn and Jürgen Geuter
Business Engineering
University of Oldenburg
26169, Oldenburg, Germany
Email: {hahn,geuter}@wi-ol.de

KEYWORDS

CSM, Systems theory, hierarchical systems, engineering, social systems

ABSTRACT

Individual skills and properties of the human beings involved have made analyzing product development systems difficult. A flexible approach to modeling and simulating these social systems is still missing. Due to their high degree of structure product development processes do lend themselves to simulation and analysis though. In this paper we outline the specific challenges and requirements for creating a complete model of social product development systems. We also propose a new approach for modeling and simulating real-world development systems.

INTRODUCTION

The modeling and simulation of processes involving human actors grows more complex the more freedom the respective actors have. Simple economical models which reduce human being to rational agents optimizing a simple number such as profit or cost are very simple to build and analyze. But while these simple models do help us understand certain social behaviors or systems better, many of the social contexts we want to model and analyze are inherently more complex, not only in the sheer amount of involved objects and properties but also in the way these objects, properties and goals need to be modeled: Human beings can quite comfortably operate based on a significant number of partially contradicting goals and pinning down a person's properties to a number is not always a suitable approach.

Looking at real-world social processes and systems the domain of product development and engineering provides us with a good opportunity to take a big step towards better understanding social behavior and systems: While engineering processes are characterized by standardized methods and processes, the individual developers' skills and properties as well as the way the whole group communicates and collaborates have a huge impact on the probability of a given project's success[Song et al., 1997].

Standardized processes, qualifications, methods, tools and metrics have lead to a growth in productivity and quality within the engineering domain over the last decades. But even with a rising degree of maturity considering processes and methods, a large number of engineering and design projects fail to either meet their required deadlines or stay within the given resource limitations. Many projects fail to reach their main goal at all. [Project Management Solutions, 2011] found that about 37% of ICT projects were in serious danger of failure, [Heeks, 2003] found that in eGovernment projects "35% are total" and "50% are partial failures". The goal of innovation competes with the more traditional project targets (lead time, cost, etc.): Where Innovation "consists in diverging processes that explore new alternatives, values and performance criteria"[Aggeri and Segrestin, 2007] Project development "consists in converging processes built around predefined targets and deadlines"[Aggeri and Segrestin, 2007]; innovation enforces a certain level of uncertainty.

The problem of project failure has mostly been dealt with from a process- or artifact-based angle: Traditional project management has developed methods to estimate effort and time[Boehm et al., 1995]. More traditional, static process models for engineering such as the Waterfall Model[Royce, 1970] or the V-Model[Broy and Rausch, 2005] have been extended (and in some areas even replaced) by more agile models (i.e. SCRUM) to help management making better estimations about project progress and to allow a quicker recovery from design mistakes. Also the design process itself is being rethought and improved, for example by compartmentalizing and formalizing the different modules of the product as for example Contract-Based Design [Sangiovanni-Vincentelli et al., 2012] suggests.

But studies such as [GPM e.V. und PA Consulting Group, 2006] and [Terry, 2002] show that "For the overwhelming majority of the bankrupt projects we studied, there was not a single technological issue to ex-

Proceedings 28th European Conference on Modelling and Simulation ©ECMS Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, Marco Castellani (Editors)
ISBN: 978-0-9564944-8-1 / ISBN: 978-0-9564944-9-8 (CD)

plain the failure" [DeMarco and Lister, 1999]. The way we model engineering projects lacks an integration of so-called "soft factors" or what DeMarco and Lister [DeMarco and Lister, 1999] call a project's "sociology": The way people interact, exchange ideas and build consensus. Building consensus also clashes with mismatching goals: Where the project as a whole has clear goals regarding quality, effort and cost, each individual might – often for organizational reasons – have significantly different goals working against the overall project's goals: Where the project targets a very high level of quality to serve as the basis for follow-up projects, the individual worker might be forced to do "just enough" to pass the tests in order to have enough time to fulfill his or her other project's obligations.

In order to understand the effects described above better and to help engineering projects circumvent or mitigate them we need executable social models for product development that allow estimating the impact of certain properties of any given social engineering system. Some typical characteristics of engineering systems, which we will outline in the next section, support building executable probabilistic models of these specialized social systems.

In the following sections we will analyze the characteristics of modern engineering systems illustrating their aptness for modeling and simulation. We will also propose a strategy to model complex, goal-oriented social systems suitable for simulation.

ENGINEERING SYSTEMS

Engineering or designing a product is, on a very abstract level, the exploration of the design space spanned by the given requirements or objectives.

The space of alternatives (which are possible artifacts fulfilling the given objectives) shrinks while adding or substantiating requirements. Partial product models like design documents or partial implementations can further shrink the design space by creating implicit requirements or dependencies. Apart from purely functional requirements, other objectives (such as the expected time until completion for example) are usually added to the System of objectives as well.

In order to separate the social part of the engineering system from the process- or artifact-focussed part, we adopt the ideas proposed in [Ehrlenspiel, 2009]: The driving force of the development process is the "System of Objectives" (SoO) which is constituted by all given explicit requirements and objectives as well as their logical, formal or technical dependencies and connections.

The SoO is complimented by the "Object system" (OS) containing the partial product models and development artifacts as well as their connections, dependencies and metadata as well as the "Action system" (AS) which consists of the people, tools, resources, processes as well as the actions these entities make. Negele [Negele, 1998] later split the AS into the "Action System" containing the people and resources and the "Process system" (PS) aggregating the different processes within the project.

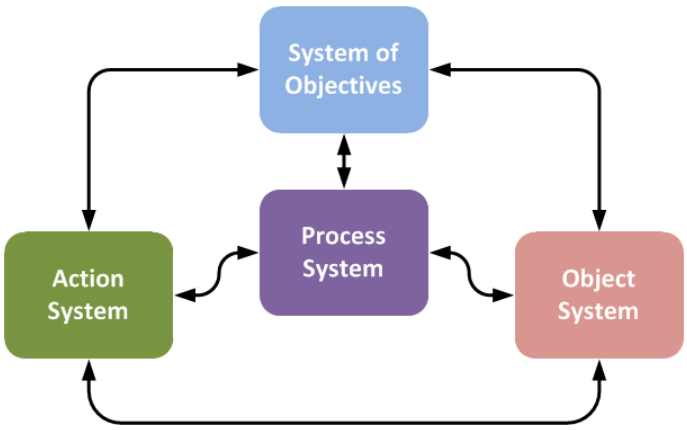

Fig. 1. Breakdown of the product development system into abstract partial systems

All those systems are separate but influence each other as figure 1 illustrates. The different ways that the systems influence each other though is not clearly defined in a general way. The SoO and the OS are tightly coupled with the entities from the OS each fulfilling one or more objectives. Only certain parts of the Action System (people, tools, etc.) can act in any of the PS's given processes or phases contributing to the completion of one or more goals.

While these very theoretical analyses of the abstract engineering system help structuring further research in the domain, they themselves were not suitable (and intended) to be used to actually model, explain or simulate existing engineering projects.

This systemic approach ignores the sociotechnical aspects of engineering: It is a task heavily based on human skills and usually teamwork. While tools help humans work more effectively and with fewer mistakes, [Song et al., 1997] has shown that "process skills, project management skills, and alignment of skills and needs" have a "strong, positive influence on [...] product performance". With mastery in a certain area taking many hours of practice and only mastery allowing innovation [Ericsson, 1998] a significant level of specialization has been characteristic of engineering systems: The process of development is often distributed amongst collaborating experts in the area of requirements engineering, design, implementation and testing.

Actual engineering in the wild has not been untouched by the trends of globalization and outsourcing in the last decades. Engineering projects, even those within one company, are increasingly global endeavors, distributing the participants within one project over many different countries. Globalization allows a company not only to build highly specialized development groups while still profiting from potentially lower wages or other environmental factors. But this approach is not without its own, often social, problems. Not only does it require a higher amount of explicit communication due to the lack of the so-called "Watercooler effect" (meaning the kind of communication happening in a company around points of non-work related social interaction, such as a Watercooler) but also opens the

whole process up for cultural mismatches and tensions as exemplified in [Nicholson and Sahay, 2001].

The orientation to project-oriented company organization like the Matrix Organization has put some pressure on the design team as well: With each individual potentially working on not only more than one project but also under more than one leadership, the individual project goals can easily come into conflict with the goals of the different organizational entities or the goals of the individual product designer.

The recent focus on international standards and certifications such as CMMI[Cmmi Product Team, 2010] and SPICE (ISO/IEC 15504) have forced globally active enterprises to further solidify and document their methods and processes providing a good foundation for modeling their real-world processes for simulations.

Given the existing research within the area of engineering and the trends we outlined, the domain of engineering and product development provides a very suitable testbed to model and simulate social and sociotechnical processes for the following reasons:

- **Standardized procedures and processes** (Waterfall, Agile development, SCRUM, V-Model, etc)
- **Well-defined process steps** (Component-based development, separation of requirements, design, implementation and testing)
- **Use of standardized metrics** (CMMI/SPICE for degree of maturity, Code Quality Metrics, Error per Line of Code, etc)
- **Division of Labor** (collaborating specialists within each domain such as design, implementation and testing)
- **Distributed development** (globally distributed teams, global collaboration between companies, global component suppliers)
- **Knowledge and skill driven** (importance of individual skills, training of engineers)
- **Social system** (Communication, collaboration)

REQUIREMENTS FOR CREATING EXECUTABLE MODELS OF ENGINEERING SYSTEMS

From sections 1 and 2 we derive the following requirements for modeling engineering systems in a way that allows their execution/simulation corresponding to engineering projects in the wild. We have categorized the requirements in the following subsections: General Requirements for the model, Social Requirements for the modeling of the social (inter)actions and Domain requirements which follow from the domain (engineering/product design) itself.

GENERAL REQUIREMENTS

In general, the model of a sociotechnical model of engineering systems has to consider the following requirements. These are not directly tied to the domain of engineering and can be applied to other similarly structured processes such as for example law-making:

- **Hierarchies of communicating systems**: In order to model the different organizational layers within complex engineering systems and companies, hierarchies need to be implemented. A system creating a product needs to be able to include the engineering systems building subcomponents. The model needs to be able to implement traditional monolithic design system (one independent R&D department) as well as modern enterprise structures such as Matrix-Organization or development processes spanning many different entities/companies in order to be able to represent actual modern development systems.
- **opaque Systems**: In order to simulate the actions of systems that cannot be made fully transparent (such suppliers of subcomponents or resources) the model needs to be able to deal with opaque, loosely coupled systems. This requirement follows directly from the trend towards globalization as we outlined in section 2. Integrating systems which do not present their internal state or objective allows modeling global distributed engineering without enforcing insurmountable levels of transparency.
- **heterogenous depth**: When modeling a complex engineering system different branches of the hierarchical tree of entities and sub-entities can have - due to the organizational structure of the project or due to a lack of deeper structured information about a part of the project - different depths of sublevels.
- **heterogenous systemic structure**: Each subsystem of a given development project can have its own process and method just as well as its own goals - possibly conflicting with the goals of a higher-ranking system. Each subsystem needs to potentially be observable as its own autonomous unit without sacrificing the opportunity of interconnectedness.

SOCIAL MODEL REQUIREMENTS

When modeling engineering processes it makes sense to make social interactions and factors very explicit. There already are established standards such as SPEM[Object Management Group, 2008] for modeling engineering processes which we need to augment with social influence factors for which we devised the following requirements:

- **Model of communication**: Communication between people and systems does not happen flawlessly. Potential defects or degradation of trust happen due to cultural or language barriers as shown by [Nicholson and Sahay, 2001] or due to the sheer perception of distance [Moon, 1999].
- **Model of knowledge and skills** (as a special form of knowledge): In a domain as dependent on human actions as engineering, a precise model of the acquisition, degradation and application of skills such as [Rasmussen, 1983] and a concept of human memory such as [Shiffrin, 2003] is central for an appropriate simulation.
- **Personal traits**: The model needs to be able to assign different characteristics to human entities within the development system with a focus on modeling biases when it comes to decision making. An example for this is the "perfectionist bias" which highly influences the decision whether to continue to improve a given

artifact or consider its quality high enough. Research [Costa and McCrae, 1992], [Bartle, 2004] shows that reducing the complexity of human behavior for specific tasks to a set of limited archetypes produces good results while providing a simple abstraction for classifying individual human beings.

- **Social interaction and organizational roles**: Product development and engineering happen within social systems and hierarchies. The model needs to take these into account and should be able to model the power structure of a given social system on a structural/organizational level as well as on a social level. This is especially relevant for the propagation of objectives and goals from higher to lower levels withing the organizational hierarchy.
- **Other flexible "weak factors"**: Studies [Ernst, 2002] have shown that many non-tangible so-called "weak factors" heavily influence product development success. An executable model of product development needs to be able to potentially include these kinds of influences.

DOMAIN REQUIREMENTS

Finally we developed the following requirements emerging from the domain of engineering itself. Most of these requirements can be fulfilled by adapting or integrating existing standards from the domain itself:

- **Model of artifacts, objectives, goals and their interdependences**: Using an integrated model of product development (as proposed in [Ehrlenspiel, 2009]) which connects the objectives and their fulfillment to the development artifacts is necessary to trace how well a given system follows its set of goals. The model of artifacts also needs to support keeping track of the efforts having gone into the different parts of the product.
- **Model of process**: The structure of processes , their sequence and the potential points where backtracking to earlier steps is necessary needs to be specified. Standards such as for example [Object Management Group, 2008] can provide a solid foundation.
- **Change of requirements**: The executable model needs to be flexible enough to deal with changing requirements or objectives. Legal requirements or a stakeholder's expectations might change during the run of the project just as a project might need to drop its targeted level of quality in order to meet deadlines. The model needs to keep track of these changing objectives and their consequences for the project itself.

HIERARCHICAL OPAQUE SOCIOTECHNICAL SYSTEMS

Our approach for modeling and finally simulating engineering systems is based on the conceptual work done in systems theory [Luhmann, 2010] that [Ehrlenspiel, 2009] and [Negele, 1998] adapted in an abstract fashion to the domain of product development: The whole engineering system is separated into partial systems (objectives, artifacts, actions, processes, etc.) (see figure 1) which interact through messages and semantic connections. This allows us to integrate already existing standards and implementations to model parts of the full executable model (like the SPEM standard [Object Management Group, 2008] for process modeling for example).

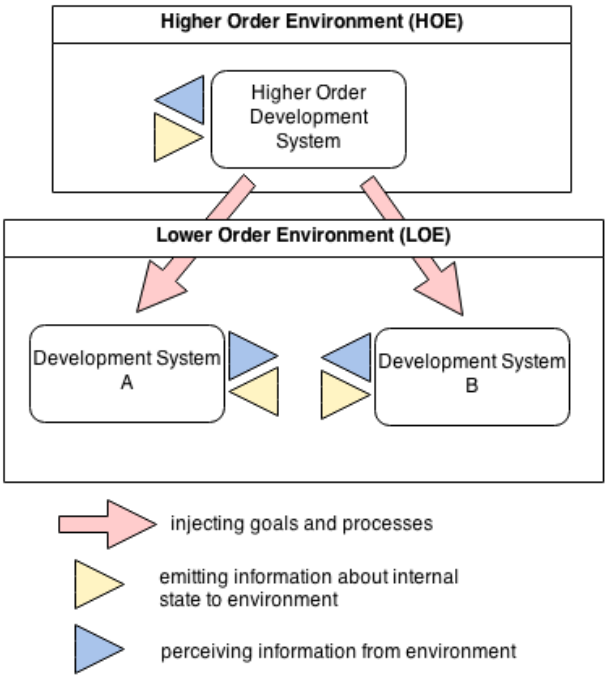

Fig. 2. Illustration of a simple hierarchical model

Using hierarchies of systems and subsystems as an architectural principle is a well established approach, especially for analysing engineering problems. [Varaiya, 1972] developed a theory for hierarchical, multilevel systems which can be applied to cooperating social systems. Finding out how to identify and separate subsystems from each other was further described by [Guinzy, 1973]. Hierarchical systems are a very well-established method for handling systemic complexity within very different systems (Software, Hardware, Social systems, etc).

In our approach we follow the definition of "system" proposed by Luhmann [Luhmann, 2010] in which systems allow no direct access to their innards but only interact with the rest of the world by "irritating" it and each other. This means that each system provides information about itself to the environment and can use other system's provided data without knowing how another system came to that state. Luhmann's model of social systems fits in perfectly with Ehrlenspiel's and Negele's model while also providing the necessary facilities to model supply chains and distributed development. There are two different ways in which different systems influence each other: Indirectly and directly.

Hierarchical opaque sociotechnical systems (HOSS) (as illustrated in figure 2) consequently connect fully developed singular models to each other either by *direct semantic connections* (such as "is higher-order system to" or "produces sub-part for") or by emitting information into an environment that is potentially shared with

other development systems. Direct connections allow the propagation of organizational goals and products of the development process whereas shared environments of systems allow the representation of so-called "weak factors" as described by [Terry, 2002]. The systems are opaque meaning that they cannot perceive each other's internal state or structure but rely on each system communicating its own state through information emitted into the environment or via explicit artifacts (such as reports of achieved level of completion or quality).

The direct way of social systems influencing each other (whether they are individuals communicating or groups of individuals) is based on the idea of social networks: The social systems (either people or groups of people) interact based on a network not only of paths of communication but also based on social and formal hierarchies. Each direct connection between social systems can transport information (potentially based on an artifact as for example a design document being exchanged via email) and each means of transportation of information can have distinctive characteristics such as percentage of misunderstanding or the organizational impact (i.e. a direct order by a superior has a different impact than the suggestion of a coworker).

The hierarchical, systemic approach allows the modeling researcher to model a complex development system in any granularity: For the big picture view only one level of few communicating systems might suffice, for in-depth analyses the hierarchy can be drilled down until the individual human beings are integrated. This also allows combining smaller, well-tested models into bigger, more complex models.

A hierarchical model allows each system to consist of subsystems of a similar type: The Action System "Company" can include multiple Action Systems "Developer" each having their own specific goals and properties next to the ones given to them by the organization. Figure 3 shows an illustration of our approach. Each individual person therefore is their own opaque system with all connected properties and degrees of freedom. Communication between people is modeled by a direct exchange of objects, either artifacts such as a written document or by an abstract "Communication Object" encapsulating a speaking act. The process of a system adapting an object to its own internal state allows the easy integration of models of communication such as [Nicholson and Sahay, 2001] or [Moon, 1999]. Skills and properties are attached to each Action System with each Action System's environment also containing descriptions of potential changes of properties (as for example a rate of learning/skill acquirement).

Each human within the system is assigned a prototype personality based on typical behavior within product development systems. These prototypes model biases of developers within the development process and are constructed analogously to the way [Costa and McCrae, 1992] and [Bartle, 2004] build their archetypes for other domains by separating individuals based on continuums of different behavioral extremes. In combination with decision heuristics and typical human non-development-related decision biases as described in [Tversky and Kahneman, 1974] this forms a usable, flexible as well as extensible model of the social and individual aspects of product development.

"Weak factors" that cannot easily be attributed to one entity can either be capsulated into the system environment or into abstract social systems that emit certain information or state into the social network of communicating and interacting developers. This approach allows the integration of non-tangible effects such as "our company if only 80% effective on Fridays" allowing the analyst of the product development system to explicate certain facts about the system that are known but hard to associate to one source.

The Action system also integrates the use of product development tools: Access to a given artifact within the model can go through a specific tool augmenting a developer's skills and performance as well as having other properties such as reduced number of errors.

The Systems of Objectives, Artifacts and Processes are not fully opaque but work similar to containers. Each social system knows its own goals and objectives as well as the ones of the social system it is part of. Similarly each social system can have its own artifacts (design document not shared with other groups) and its own processes. Artifacts can be transferred to a different social system through means communication (with the potential of misunderstanding as outlined above).

Development systems with a high degree of maturity will have more information about their inner workings than less mature systems. The approach for modeling development systems as it is described here is extensible enough to include any sort of communication or influence factor without enforcing any.

Because of its structural separation of different systems, the import of existing information from the development tool landscape is possible: Artifact and objective information from a Product Data Management (PDM) system or a source code management system, processes from a process modeling tool or even very simple process models such as those project management tools (MS Project etc.) can provide.

SIMULATION OF HIERARCHICAL OPAQUE SOCIOTECHNICAL SYSTEMS

Right now the HOSS approach does offer a new way to think and structure models of engineering and product development systems. The next step is to create simulators from those descriptive models.

We have implemented simple, prototypical simulators of HOSS models for simple engineering systems with a limited amount of participants, goals, skills and artifacts. Given our experience we are currently in the process of redesigning a more complete and well-defined framework of building blocks for creating HOSS models.

Simulations of HOSS are driven by the hierarchical Action Systems (see figure 3) making decisions according to their skills and their goals concerning the process itself. Applied to the domain the action systems encap-

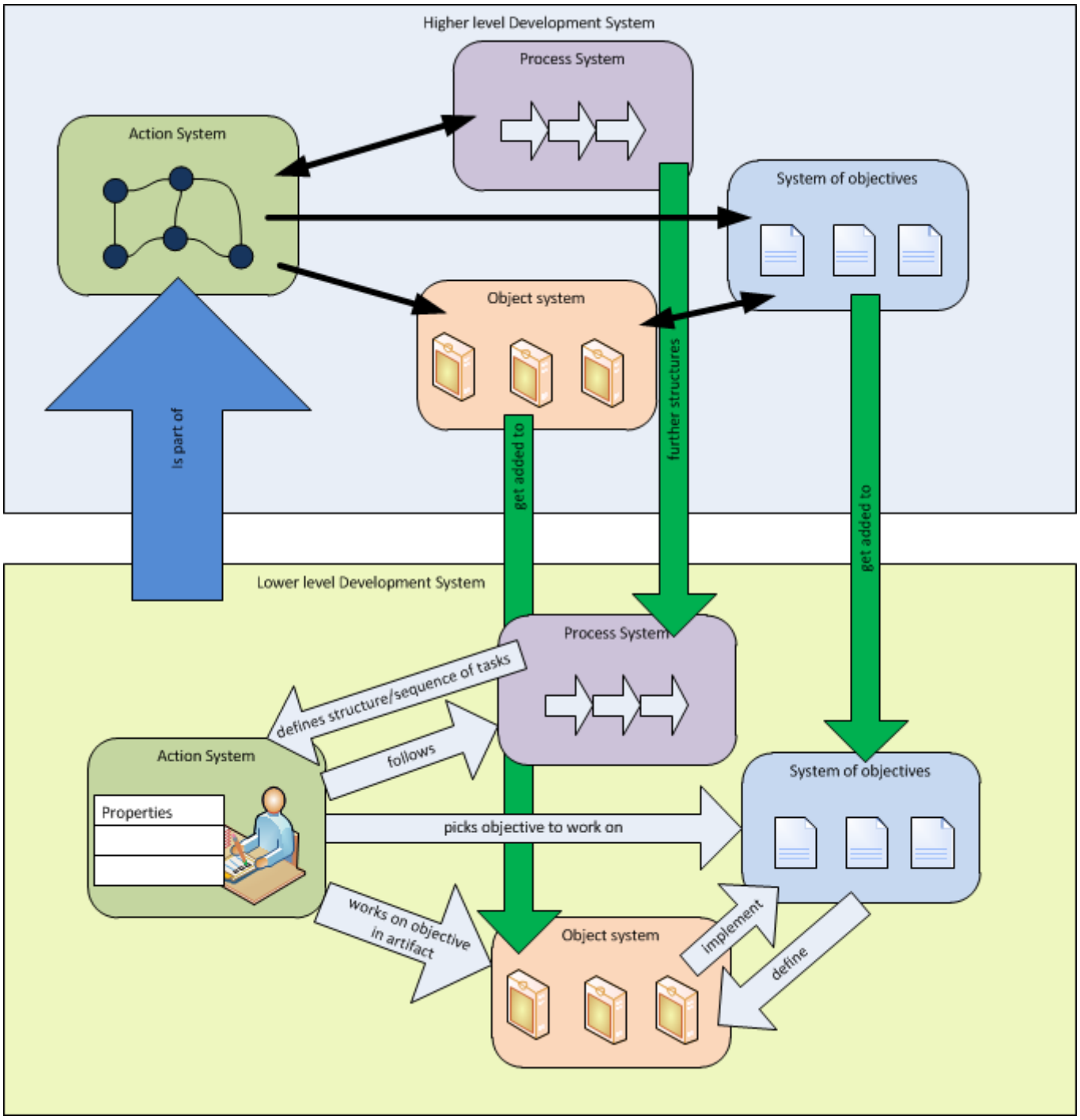

Fig. 3. Illustration of a simple hierarchical model

sulate individual human beings within the development process, groups of people participating in the development process or complete (external) entities such as suppliers, partners, the regulating government or other stakeholders.

The model is based on a discrete understanding of time, with each Action System deciding for each step how to invest their energy in each step of time. While actual work on the implementation of a goal costs a lot of energy, communication is relatively cheap but can create a certain level of stress affecting other properties negatively. Actions can last for more than one step of time depending in the scaling of each step's duration, this allows integrating systems with a very fine-grained resolution of time with systems who only perform long-lasting actions (such as integrating a local development team whose actions are known within a 10 Minute granularity and an external supplier providing feedback every week).

In order to implement a HOSS model for a real-world development system, it needs to be calibrated to that system. Individual developers or units of developers need to be characterized which can be done either manually or by using techniques like standardized questionnaires. Certain factors of human (inter-)actions can be implemented based on existing psychological and soci-

ological studies and research but have to be evaluated for any given development system. A baseline for how big one unit of work is for any given system has to be derived based on previous projects data as for example outlined in [große Austing and Hahn, 2009].

CONCLUSIONS

The domain of product development and engineering is especially suited for developing a deeper understanding of complex social systems. Its specific characteristics which we outlined in the second section of this paper provide a good compromise of a high degree of individual freedom in decision making as well as a strongly structured and well-defined environment.

Based on existing standards and abstract models hierarchical opaque sociotechnical systems (HOSS) do allow to very precisely model given complex product development systems, thereby helping in analyzing and communicating the structures and dependencies within complex sociotechnical systems.

The structure of HOSS allows the simple implementation of simulators for these complex systems by integrating existing simulations, standard models and statistical models. The next step in our research is to create a set of well-defined building blocks for implementing these simulators as well as a framework to execute these which is the current focus of our work.

The concept of HOSS is not tied to product development but can also easily be adapted to other domains with a similar hierarchical structure such as law-making or other processes of structured decision-making.

HOSS can provide a flexible and simple way to describe and explain social interactions within highly structured environments such as the product development domain.

In order to evaluate the concept more than just theoretically, defining, describing and implementing a simple basic frameworks to implement HOSS is necessary which we already started work on.

REFERENCES

[Aggeri and Segrestin, 2007] Aggeri, F. and Segrestin, B. (2007). Innovation and project development: an impossible equation? Lessons from an innovative automobile project development. *R&D Management*, 37(1):37–47.

[Bartle, 2004] Bartle, R. (2004). *Designing Virtual Worlds*, volume p.

[Boehm et al., 1995] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1(1):57–94.

[Broy and Rausch, 2005] Broy, M. and Rausch, A. (2005). Das neue V-Modell® XT.

[Cmmi Product Team, 2010] Cmmi Product Team (2010). CMMI® for Services, Version 1.3 CMMI-SVC, V1.3 Improving processes for providing better services. Technical Report November, Carnegie Mellon University.

[Costa and McCrae, 1992] Costa, P. T. J. and McCrae, R. R. (1992). *NEO-PI-R professional manual: Revised NEO personality and NEO Five- Factor Inventory (NEO-FFI).*, volume 4.

[DeMarco and Lister, 1999] DeMarco, T. and Lister, T. (1999). *Peopleware: Productive Projects and Teams 2nd Ed.* Dorset House Publishing Co., Inc.

[Ehrlenspiel, 2009] Ehrlenspiel, K. (2009). *Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit.* Hanser Verlag, 3., aktual edition.

[Ericsson, 1998] Ericsson, K. A. (1998). The Scientific Study of Expert Levels of Performance: general implications for optimal learning and creativity. *High Ability Studies*, 9:75–100.

[Ernst, 2002] Ernst, H. (2002). Success Factors of New Product Development: A Review of the Empirical Literature. *International Journal of Management Reviews*, 4:1–40.

[GPM e.V. und PA Consulting Group, 2006] GPM e.V. und PA Consulting Group (2006). Ergebnisse der projektmanagement studie konsequente berücksichtigung weicher faktoren.

[große Austing and Hahn, 2009] große Austing, S. and Hahn, A. (2009). Measurement of product model complexity based on the integrated PLM model. In *Proceedings of The 6th International Product Lifecycle Management Conference*, page 100. University of Bath.

[Guinzy, 1973] Guinzy, N. (1973). System identification in large scale systems with hierarchical structures. *Computers & Electrical Engineering*, 1:23–42.

[Heeks, 2003] Heeks, R. (2003). Most e-government-for-development projects fail how can risks be reduced?

[Luhmann, 2010] Luhmann, N. (2010). *Introduction to systems theory.*

[Moon, 1999] Moon, Y. (1999). The effects of physical distance and response latency on persuasion in computer-mediated communication and human–computer communication.

[Negele, 1998] Negele, H. (1998). *Systemtechnische Methodik zur ganzheitlichen Modellierung am Beispiel der integrierten Produktentwicklung.* Utz.

[Nicholson and Sahay, 2001] Nicholson, B. and Sahay, S. (2001). Some political and cultural issues in the globalisation of software development: case experience from Britain and India. *Information and Organization*, 11(1):25–43.

[Object Management Group, 2008] Object Management Group (2008). Software & Systems Process Engineering Meta-Model Specification. *Process Engineering*, (April).

[Project Management Solutions, 2011] Project Management Solutions (2011). Strategies for project recovery.

[Rasmussen, 1983] Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE TRANS. SYS. MAN CYBER.*, 13(3):257–266.

[Royce, 1970] Royce, W. (1970). Managing the development of large software systems. *proceedings of IEEE WESCON*, 26:1–9.

[Sangiovanni-Vincentelli et al., 2012] Sangiovanni-Vincentelli, A., Damm, W., and Passerone, R. (2012). Taming Dr. Frankenstein: Contract-Based Design for Cyber-physical Systems. *European Journal of Control.*

[Shiffrin, 2003] Shiffrin, R. (2003). Modeling memory and perception. *Cognitive Science*, 27(3):341–378.

[Song et al., 1997] Song, X. M., Souder, W. E., and Dyer, B. (1997). A causal model of the impact of skills, synergy, and design sensitivity on new product performance. *Journal of Product Innovation Management*, 14:88–101.

[Terry, 2002] Terry, C.-D. (2002). The "real" success factors on projects. *International Journal of Project Management*, 20(3):185–190.

[Tversky and Kahneman, 1974] Tversky, A. and Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases. *Science (New York, N.Y.)*, 185:1124–1131.

[Varaiya, 1972] Varaiya, P. (1972). Theory of hierarchical, multilevel systems. *IEEE Transactions on Automatic Control*, 17.