ECONOMIC IMPACT ESTIMATION OF NEW DESIGN METHODS

Roland Koppe[1], Stefan Häusler[1], Frank Poppen[1], Stephan große Austing[2], Axel Hahn[2]
(1) OFFIS Institute for Information Technology (2) University of Oldenburg, Germany

ABSTRACT
New design methods and tools often promise large benefits for specific engineering tasks or whole engineering processes to make increasingly complex and sophisticated products possible. However, estimations for the pay-off of new methods or tools are based on gut feelings or rare expert knowledge. In this paper we present our approach for well-founded quantitative estimations for the economic impact of new methods or tools. We show the beginning of our methodology with an early experiment and the impact analysis for a case study of a design flow for electronic circuits.

Keywords: Impact analysis, new methods, simulation

1 INTRODUCTION
The development of high technology products like computer chips or software is today based on well-defined engineering processes. Improvements of these processes increase productivity and reliability of the products. For example Moore's Law states that the number of transistors on circuit boards doubles approximately every two years. Besides advances in materials and manufacturing this requires an increased productivity through improvements in the engineering processes for chips.
In the 1960s software engineering was lagging far behind the new technical possibilities. The term "software crisis" described the inability of the existing processes to cope with the rapidly increasing complexity. The challenge to improve the engineering process efficiency continuously is vital for technology companies to be competitive in the long term. This requires a technology monitoring to identify relevant technologies early enough and the investments to exploit these technologies in the existing processes. The implementation of a new tool or method in this context is a strategic decision that must be based on a well-founded estimation of the impact because these investments come at a high cost and the economic benefit is visible late.
Often the promised cost savings cannot be realized as imagined and expensive tools cannot speed up the engineering process. The cost-effectiveness of a process investment is hard to assess because not only the directly affected process element but the engineering process as a whole must be considered. Pilot projects provide an indication but no statistically significant results [1] and come at high costs. In another development context with different parameters like a different product, designers or activities, the evaluation results will vary. Thus a defined methodology is needed for well-founded estimations of engineering process changes.
In this paper we present a framework for this assessment based on an extended stochastic process modeling with consideration of relevant product properties. The cause-effect relationships between product properties and process (e.g. from expert knowledge) are modeled to capture consecutive effects of local changes. The estimations provided by this framework help process designers and engineers to choose the right path for continuous improvements. It also provides means to economically evaluate investment alternatives in engineering processes.
To develop our framework we used an inductive research approach. We derived the basis for impact modeling and analyses from existing literature described in Section 2. In Section 3 we present our model based on related work and findings from expert interviews and experiments. In Section 4 we present a case study from electronic design automation domain, where we estimate the impact of a new automation tool. We close our paper with a conclusion and outlook in Section 5.

2 RELATED WORK
Engineering processes are changing more frequently and intensively than most business processes. The technical change is extremely fast-paced in some engineering domains compared to other

industries and firms must always adapt to these changes to stay competitive [2]. There are different options to change engineering processes to stay efficient. However, investments in process changes have different economic benefits that are often not obvious. For example a simulation tool can be used to automate or speed up a certain process step. The profits are lower process costs or faster time-to-market. If only costs are lowered the cost-effectiveness can be analyzed locally. A faster process step on the expense of tool costs is on the other hand not always the most cost-efficient option as other options may reduce the process step use significantly. An example is to handle the cause of quality problems and not the symptoms. Another scenario is the introduction of a new step that causes additional costs but provides benefits for the succeeding tasks.

The cost-effectiveness of new tools and methodologies is thus in most cases not assessable without the modeling of process dynamics. Thus our literature search was focused on work that deals with explicit modeling of processes in engineering and especially changes to these models which are mostly called impact or sensitivity analysis. The related work mainly focuses on applications in software engineering. In contrast, there is far less work on performance estimation and impact analysis in other development domains, e.g., [3] for chip design and [4] for mechatronics. The main problems are about the same, though. This is why we mainly refer to established work in software engineering.

A popular approach in this area is Software Process Simulation (SPS) coined by Abdel-Hamid and Madnick [5]. Applications of SPS include performance analysis, what-if-analyses and prediction of process change effects. For an overview we refer to [6] and [7]. SPS models are normally derived from empirical data to reflect real world behavior [8] which is also known as empirical software engineering. Its goal is to describe the cause-effect relationships between process elements. The detail level of this description is proportional to the impact on critical factors like time, cost and quality. In contrast to the empirical software engineering the method engineering approach uses process meta-models for the comparative review for method and tools [9]. Other approaches include application of stochastic process modeling like Markov chains to describe and analyze processes which describing the possible or more likely flow using transition probabilities between states or process steps, e.g., the work of [10]. Johnson et al. use Markov chains [11], and analytic models [12] as well as a combination of these approaches [13] for the analysis of iterative design processes. The basic idea to use stochastic modeling is to reduce the necessary effort for modeling and analysis of process models. However, existing literature does not provide a common generic model for the description of engineering processes and their inherent cause-effect relationships. The processes are only given on a very abstract level ([14], [15]). Other authors disregard some products and their properties that have a relevant effect on the process dynamics (compare [16]). Beside this there are several more or less similar procedures to perform impact analyses, e.g., [17], [18] and [19] that have to be transformed for other domains than software engineering.

In the systems engineering community effort is spend on the evaluation of systems engineering efficiency, its return on investment (SE-ROI) and on the question how much “Systems Engineering effort” is enough. The International Council on Systems Engineering (INCOSE) has made the assessment of the return on investment in systems engineering a high-priority research topic in its Vision 2020 document [20]. Honour summarizes in [21] different studies on systems engineering value, which all address specific questions and goals. But their results are not conclusive and hard to apply to other system engineering programs [22].

Our literature survey shows that there are several approaches to describe and analyze the impact of process changes in general or with an economic view. Even if the depicted approaches are about the same problem there is no standard-conform and easily manageable method to describe processes and new methods for impact analyses. So we decide to study the basic mechanisms of engineering processes and derive a standard-conform model to draw founded conclusions about processes, described in the following section.

3 A MODEL FOR ENGINEERING PROCESSES AND BEHAVIOR

The first step in building a model to analyze engineering process changes was to understand the basic mechanisms in such processes. For this purpose, we selected the domain of hardware design and performed interviews with in house engineers as well as experiments. We compared our findings to related work, e.g., [11], [14], [17] and [23], and derived needed process elements and process behavior to be represented in an impact analysis model. In the following, we describe our results along one

conducted experiment about the customization of an already existing chip design for new environmental constraints. The engineering process was performed once.
Three characteristic properties of work products and goals were described which have to be fulfilled for the resulting product. These quality properties are:

- the area of the resulting design must not exceed 1 mm²,
- timing has a hard constraint which is set to a 200 MHz clock frequency and 5 ns clock period and
- power has a softer constraint which is fulfilled if the power estimation is maximal 20 mW with a tolerance of 5% (+1 mW).

The result of the experiment was the documentation of the process and its process behavior for a specific product performed by one specific role (engineer). During execution of the development process the engineer reports the current iteration of each task, the necessary tool and human effort for each task, the current value states of relevant properties of work products as well as causes of re-iterations.

3.1 Description of the engineering process

The design process consists of four sequential tasks. During the *Design* task the engineer modifies the Very High Speed Integrated Circuit Hardware Description Language (VHDL) code of the design to fix functional bugs or optimize the design against the defined goals. The *Synthesis* task contains the automatic synthesis of the VHDL code into a gate level netlist with the tool Design Compiler (DC) from Synopsys. The tool was constrained to achieve the defined area and timing goals. During Place & Route (P&R) the engineer uses the tool First Encounter from Cadence for the alignment and routing of the gate cells. This task delivers accurate estimates for area and timing. The engineer was instructed to analyze the current property estimates in the Analysis task, even if there are estimates in previous tasks. The aim was to keep the decision point for the following tasks as simple as possible. In the Analysis tasks the engineer simulates the design with the ModelSim tool from Mentor to analyze the functional correctness. If the design was correct the engineer used the Power Tool from Synopsys to estimate the dynamic power of the design. Also the engineer decides how to proceed in the design process based on the current property values.

3.2 Reported effort, property values and behavior

Table 1 shows the reported human effort and tool runtime for each iteration and task in hours. Since the 0th iteration mainly contains the effort to setup the environment and estimate the initial property values and does not contain the effort to implement the design, the design effort is not representative (shaded cell). In the 2nd iteration a simulation problem occurs as a result of too fast clock frequency parameter for simulation. The engineer fixes this issue and proceeds with the analysis. In the 6th iteration the simulation was aborted since the design's behavior worked not like expected. After failure analysis the engineer fixes the bug in the Design task. But also in the 7th iteration the analysis shows a failure generated during P&R, so the analysis was aborted. The engineer fixes this bug in the 8th iteration and we received a design which fulfills all defined goals.

Table 1. Reported human effort and tool runtime in hours (h) for each task and iteration (left). Current property values, green for fulfilled goals, red for violations (right).

Iter-	Design (h)		Synthesis (h)		P&R (h)		Analysis (h)		∑ (h)		area	tim.	pow.
ation	human	tool	human	tool	human	tool	human	tool	human	tool	(mm²)	(ns)	(mW)
0	0.10	0.01	0.60	0.02	2.60	0.02	0.40	2.76	3.70	2.81	0.76	6.66	50.35
1	3.30	0.02	0.20	0.04	1.00	0.06	2.10	3.66	6.60	3.78	0.76	11.00	17.50
2			0.50	0.05	0.70	0.05	2.00	13.28	3.20	13.38	0.94	6.76	25.08
3	2.50	0.05	0.25	0.04	0.70	0.05	1.00	3.73	4.45	3.86	0.67	6.76	19.84
4			0.17	0.04	0.25	0.05	0.60	3.49	1.02	3.58	0.68	8.60	19.40
5			0.23	0.12	0.50	0.09	1.10	3.97	1.83	4.18	0.68	6.80	20.60
6					2.20	0.82	2.30	0.34	4.50	1.16	0.68	n.a.	n.a.
7	0.40	0.02	0.10	0.03	0.40	0.10	1.10	0.08	2.00	0.22	0.63	n.a.	n.a.
8					1.00	0.47	0.90	3.76	1.90	4.23	0.63	5.00	20.90

3.3 Findings and conceptualization

From the description of the engineering process (3.1) we derive a conceptual process model where a *Task* (e.g. Synthesis) describes a piece of work which modifies or produces *Work Products* (e.g. VHDL design). Also a Task is connected to other Tasks as predecessor (e.g. Design) or successor (e.g. P&R). Further, a Task is performed by a *Role* (e.g. Engineer) which can use Task associated *Tools* (e.g. DC). Each process element is described by a set of *Properties*. For example the VHDL design has the three characteristic Properties area, timing and power. These Properties are verified against defined *Goals* during the design process.

To model the behavior of an engineering process we define three *Behavior Models* described in the following. From the documented property values in Table 1 and the reported causes of iterations (3.2) we derive that iterations to previous tasks depend clearly on Properties. We model such relationships as *Decision Models* and give three examples for such models:

- in the 0^{th} iteration the timing and power goals are violated and require obviously a re-design,
- in the 1^{st} iteration the timing goal is not fulfilled, but the engineer assumes that it can be approached through synthesis parameters and
- in the 2^{nd} iteration neither the timing goal nor the power goal can be fulfilled, so the engineer iterates to the Design task.

For economic impact estimations we have to consider the necessary effort of a design process. Figure 1 shows the aggregated human effort and tool runtime over iterations in hours according to Table 1. The diagram visualizes that the human effort mainly decreases from iteration to iteration. In combination with Table 1 we see that it is less likely that the engineer iterates from Analysis to Design to fix bugs or optimize the design for the defined Goals. Also the engineer focuses in the later iterations on P&R. On the other hand the tool effort lies almost about 4 hours dominated by the simulation effort which is also influenced by unforeseen events in the iterations 2, 6 and 7.

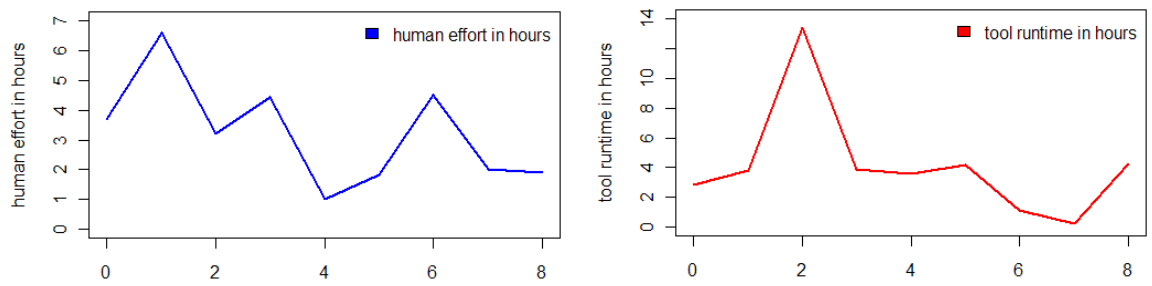

Figure 1. Reported human effort and tool runtime in hours for each iteration

Therefore, we describe the necessary effort for individual tasks as an *Effort Model* and distinguish between human effort and tool effort as their behavior is different. In Table 1 we see that the effort is not a constant value but changes dynamically over iterations. Obviously the effort of a Task is impacted by Properties of Roles (e.g. experience) who perform a piece of work, used Tools (e.g. accuracy) and Work Products (e.g. size) which are inputs to a Task. Also we learn from Table 1 that a Property's value is not constant during the design process but changes from iteration to iteration with a trend to achieve a defined Goal. We describe such behavior as *Added Value Models*.

The described experiment was performed by an engineer who has several years of experience and is familiar with the selected design. We assume that the experience as well as psychological properties (e.g. experience) have also an influence on a Task's effort, the decision for following Tasks and on Work Product Properties as the engineer modifies the design and constraints Tools.

Since not all Properties and their impacts are fully known or quantifiable and extensive pilot projects or empirical studies, which would allow "accurate" estimations of the possible impact of a new method or a new tool, consume enormous amounts of time and money and limited pilot projects often do not lead to certain statements about benefits and drawbacks of a new method in similar as well as different process contexts [1], we want to keep the factor of uncertainty and probability in our model.

3.4 Foundation of the model

Approaches which use static transition probabilities to describe a process' progress, e.g., with Markov Chains [24], [15] do not reflect the process behavior we identified. As consequence, we derived a model which allows modeling, simulation and estimation of a process' behavior.

As basis for our model we selected the Software & Systems Process Engineering Meta-model (SPEM 2.0) a standard of the Object Management Group (OMG) [25]. SPEM 2.0 provides the description of reusable process elements and therefore the management of process and method knowledge. This is very relevant for company contexts, as reusable formalized knowledge avoids reinventing the wheel. Figure 2 shows core concepts to describe method knowledge (definition) and concrete engineering processes. For a process instantiation the definition concepts pass their semantic and associations to specialized process elements. In a concrete *Process*, *Activities* and *Task Use* build work breakdown structures which are associated with concrete *Work Product Uses* and *Role Uses*. We extend this description mainly by Properties, Goals and Behavior Models.

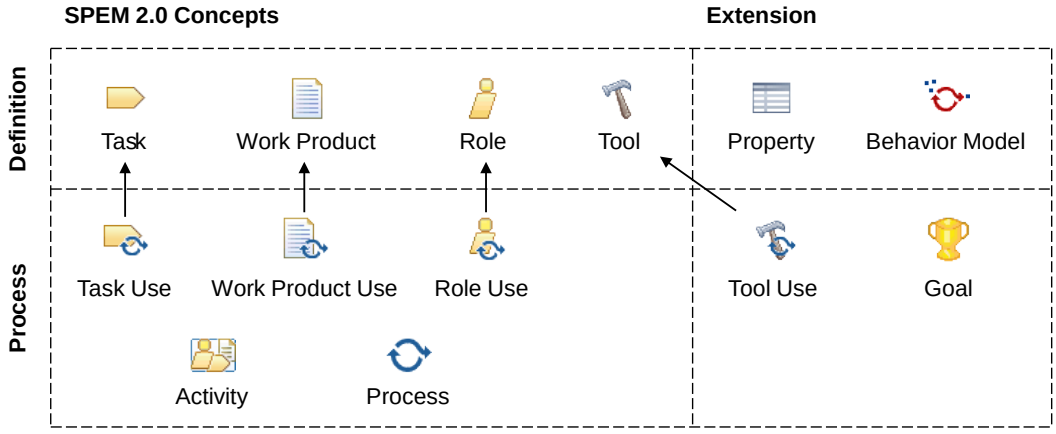

Figure 2. Core concepts for method definition and process description, with extension for the description of process behavior.

For further background information to our model we refer to [26]. After we have a formal and standard-conform model, we performed a case study to evaluate our model. We describe the case study and the instantiation of our model in the following section.

4 AN ELECTRONIC DESIGN AUTOMATION CASE STUDY

The case study originates from the automotive domain and is about the development of a hardware design for a field programmable gate array (FPGA). The objective of the design process is the integration of an edge detection filter (EDF) in an existing video processing algorithm as a part of a night vision system for cars. With our impact analysis we would like to answer if a high level synthesis tool would pay-off for the chip design process. The next subsections follow our methodology we defined in [27] for impact analyzes. So the case study allows the evaluation of our model as well as our methodology.

4.1 Definition of impact analysis goals

To keep the case study as simple as possible, the only goal of the impact analysis is to reduce the overall process effort in person hours to develop the mentioned hardware design. According to this goal we have to ask what the current most likely effort is and what factors influence the process' effort. Thus we derive a simple metric that observes a task's necessary effort and therefore as sum the necessary effort for the development process.

4.2 Description of the process model

As described in Section 3 the model of our as-is design process consist of a number of Tasks, Work Products Roles and Tools as well as their characteristic Properties. Figure 3 depicts the process model with process elements relevant to the specified goal and modeled with our impact tool [28]. The current development process starts with an existing C/C++ model of the system under development which implements the complete functional behavior of the night vision system. The system is also implemented on a FPGA with exception of the EDF.

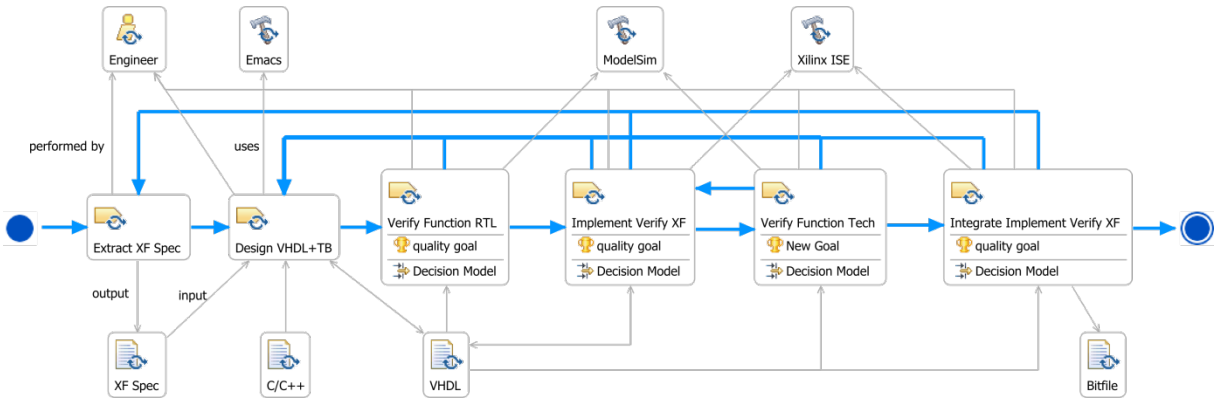

Figure 3. Modeled as-is process with tasks, their relations, roles, tools and work products

An *Engineer* has to implement the EDF according to the C/C++ model and integrate it into the rest of the night vision system. The aim of the first Task *Extract XF Spec* is to extract the extra functional requirements (e.g. bit width, latency, throughput, clock frequency) from the environment of the night vision system, the automobile. This Task is followed by the manual *Design VHDL* Task to re-implement the functional model into the accordant hardware description for the FPGA resulting in a *VHDL* Work Product on register transfer level (RTL). Even though this is apparently tedious and error prone, it is state of the art in industry. The implementation is followed by *Verify Function RTL* to test if the VHDL has the expected behavior. It is very likely that the VHDL is not fully functionally correct in the first iteration as our studies in Section 3 have shown. So we have to iterate to previous Tasks. After the goal functional correctness is fulfilled we proceed with *Implement Verify XF* where we generate a technology specific netlist of Boolean functions, place and route the netlist and verify the non-functional and functional requirements. If the analysis fails we iterate to the corresponding Task. As the previous analysis does not consider technology specific problems, e.g., timing violations and race-conditions, the Engineer has to verify the functional correctness. The last Task describes the integration of the VHDL of the EDF into the remaining system. If the Task succeeds we get a bitfile that can be uploaded to the FPGA.

4.3 Identification of cause-effect relationships

Exemplarily we have a closer look at the Verify Function RTL Task that verifies the VHDL and returns to the Design VHDL Task if the necessary functional correctness goal, a quality goal, is not fulfilled. The effort of this Task is influenced by the complexity of the input *VHDL* work product as more complex VHDL requires more effort to mature. Furthermore, the effort of the Tasks depends on the experience of the Engineer who performs the verification Task. With each iteration of the design Task the Engineer becomes more familiar with the VHDL and thus the necessary effort of the Task decreases. According to that we define three behavior models: an Effort Model for the necessary effort of the Task, an Added Value Model for changes of the functional correctness and a Decision Model to select the following Task. We explain these Behavior Models in Section 4.5 in more detail.

4.4 Collection of empirical data

For describing the mentioned cause-effect relationships in terms of Behavior Models, we performed an interview with an expert who has several years of experience in FPGA design. For example we asked for the minimal, maximal and likeliest effort of an activity during each iteration, the probability of rework and added value for specific Properties of artifacts with each iteration. Other interviews of industry experts or automatic data collection and analysis might lead to different understandings and relationships. Therefore our model supports flexible definition and reuse of already known Behavior Models. For example an Effort Model can be based on a company specific calibrated COCOMO model, the function point approach, regression models or dynamic models, e.g., [29], [30], [31]. For each approach frequent design data needs to be collected to increase the accuracy of the used models when expert knowledge is not sufficient.

4.5 Quantification of cause-effect relationships

The quantification of the cause-effect relationships leads to the following three models. From our expert interview and our studies in Section 3 we learned that the necessary effort for an iteration of a

Task is impacted by unforeseen random events and decreases from iteration to iteration in general. Also the effort is impacted by the complexity of the artifact in terms of how hard is it to create (difficulty of creation). Therefore our Effort Model is a probabilistic function which gives an effort value (e) for the current iteration (starting with 0) between the lowest (l) and highest (h) assumed effort for a specific complexity (c). The function *rnd* gives a uniform distributed random value between 0 and 1. This relationship is illustrated in Figure 4 where the areas describe the possible effort for two designs of different complexities c=1 and c=5.

$$e = l \cdot c + \frac{h \cdot c \cdot rnd()}{iteration + 1} \tag{1}$$

During the Verify Function RTL Task the functional correctness (fc) of the VHDL artifact is verified against a specific Goal we set to 1. This is similar to our observation in Section 3. We define a Decision Model which describes the following Task based on the Goal. So we get the following Boolean model.

$$\begin{aligned} &\textit{if } (fc \geq goal) \textit{ then select Implement Verify XF} \\ &\textit{else select Design VHDL} \end{aligned} \tag{2}$$

An Added Value Model describes the value of the VHDL's functional correctness over iterations. At the beginning of the VHDL Design it is very unlikely that we get a fully functionally correct design. It is rather the case that the functional correctness increases from iteration to iteration. Also it is common that more complex VHDL requires more iterations to mature. So we derive the following probabilistic model from our interview which approaches the defined goal over several iterations. The probability for functional correct VHDL is depicted in Figure 5 for different assumptions of complexity.

$$min\left(rnd() + \left(1 - \frac{1}{exp\left(\frac{iteration}{2c}\right)}\right), 1\right) \tag{3}$$

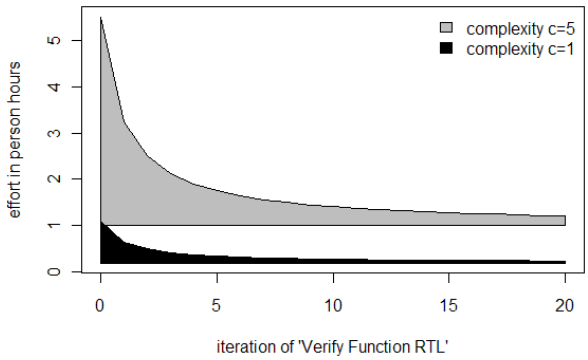

Figure 4: Verification effort depends on iteration and artifact complexity

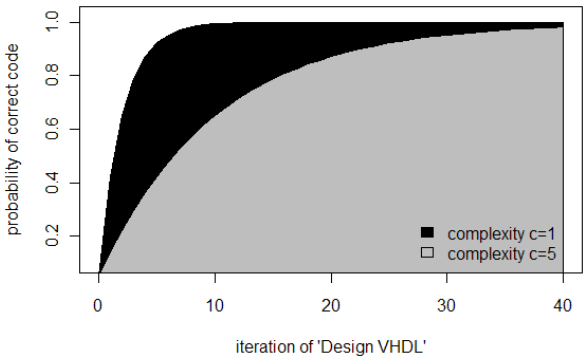

Figure 5: The probability of functional correct VHDL code increases

4.6 Description of the new method

Our case study continues with a modification of the existing process. We introduced a new method "high level synthesis" (HLS). In our case study we replace the manual Design VHDL Task by HLS which transforms a SystemC artifact automatically into a VHDL artifact. To get a SystemC artifact we have to enhance the existing C/C++ model. Now we want to study if the effort for the changed development process outperforms the effort of the existing development process.

With introduction of HLS the process is enhanced by further Tasks in front of the *Design VHDL* Task to design the EDF in SystemC, verify the functional behavior and transform the SystemC artifact into the accordant VHDL artifact. In contrast to the as-is process we have to spent more effort at the beginning of the process in describing the EDF in SystemC, verify and transform it. This is modeled as done before by Task specific Effort Models.

For the impact analysis we assume that the functional correctness of the resulting VHDL artifact has a high initial functional correctness in contrast to the as-is process since the HLS transforms valid SystemC into valid VHDL and this reduces the effort for following Tasks.

4.7 Simulation and impact analysis

In this step, we compare the as-is process against the alternative process including the new HLS method using simulation to assess the process change impact. Also for simulation and impact analyses we used our impact tool. First, we run 10,000 simulations for each process to analyze both more likely as well as extreme process behavior. As stated in the goal definition step 5.1, we want to compare the effort of the processes in person hours. Figure 6 shows the difference between the as-is process (grey) and the changed process including high level synthesis (black) for an artifact complexity of 1 which is a sensitive parameter to the described Behavior Models. The density plot displays the necessary effort on the x-axis and the density on the y-axis. By observation one can describe the obvious differences between the simulated processes. The changed process with the new method introduced is located in front of the as-is processes. Also the as-is process has a higher variance. For a simple artifact like our EDF the mean manual effort is around 23.03 hours and the HLS effort lies around 13.84 hours. But this is not a reliable comparison for strategic decision making. So we have to study the difference in more detail. For example we are interested in the mean difference of the two observations. This is defined as $Z_i = X_{1i} - X_{2i}$ with i = 1, …, n=10,000, where X_{1i} are the observations of the process effort of the as-is process and X_{2i} are the observations of the process effort for the changed process. For the difference Z we define the mean difference process effort as (4) and the unbiased variance of Z as (5).

$$\bar{Z}(n) = \frac{\sum_{i=1}^{n} Z_i}{n} \approx 9.19 \tag{4}$$

$$Var = \frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \bar{Z}(n))^2 \approx 5.21 \tag{5}$$

Thus we can calculate the confidence interval (6) of the mean process effort difference approximately (cause the observations are not ideally normal distributed with a skewness to the right) as

$$\bar{Z}(n) \pm t_{n-1,1-\alpha/2}\sqrt{\frac{Var}{n}} \tag{6}$$

where t describes the students t-distribution with n-1 degrees of freedom and 1-α is the probability that the real mean of the difference is in the resulting interval. For our observation we are interested in the mean difference covered by 99.8% (α = 0.002) of all cases. The result is an interval which states that the changed process will be in mean 9.12 to 9.26 hours faster than the current as-is process in 99.8% of all cases. Similar calculations can be made for other points of interest, e.g., for comparison of percentiles.

To evaluate the possible influence of the changed process we have to consider not only the effort difference but also the necessary investments for the introduction of the new method. Assume that we have specific introduction costs C for purchasing, training and introduction. Assume further the changed process will be performed multiple times n under similar conditions so that the behaviors of the processes are comparable. Then the break-even point is given by $C \leq n \cdot c(\Delta)$ where $c(\Delta)$ is the mean cost difference for an effort value from the interval. For an optimistic estimation of the benefit of the HLS under the given conditions we assume investments C about 100,000 Euro and $c(\Delta) = 9.12$ h · 100 Euro/h = 912 Euro as the costs for the selected effort. We see that the new method would pay-off after 100,000 Euro / 912 Euro ≈ 110 performed HLS processes for the EDF.

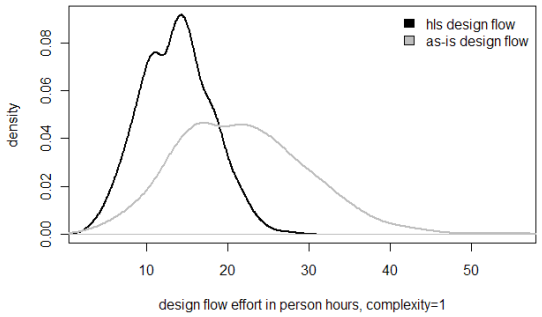

Figure 6. Density of process effort

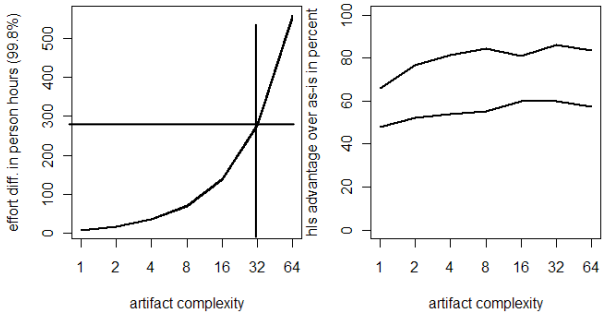

Figure 7. Effort difference and advantage of HLS

Sensitivity analyses about the complexity of the design artifacts show that HLS provides higher benefit for more complex artifacts but its advantage about the as-is process is almost constant as Figure 7 illustrates. We study the effect of HLS for a simple intellectual property (IP) component, e.g., a digital signal processor, where the interviewee assumed a complexity of 32 with a mean effort of 480 hours in comparison to the EDF. The HLS process gives a mean effort of 202 hours. The lower bound of the 99.8% confidence interval lies about 275.72 hours. So the HLS for a more complex design pays-off after 100,000 Euro / (275.72 h · 100 Euro/h) ≈ 4 performed processes.
The impact analysis and sensitivity analysis aids the estimation of the possible benefit of a process change through a new method. For our case study the main manual effort was moved from the VHDL design to the higher abstraction level of HLS. So the difficulty of creation of complex designs is influenced by HLS. The automated transformation of SystemC into VHDL reduces the overall design effort significantly.

4.8 Validate and improve

The results depicted in the previous section have to be validated against real world experience and/or collected data. As the derived results for possible benefits and drawbacks are only as good as the underlying model of the process and accordant cause-effect relationships, these have to be improved if necessary.

5 CONCLUSION AND OUTLOOK

In this paper we presented an approach for the economical assessment of strategic investments. From an experiment we derived a formal model which extends the standard SPEM 2.0 for process description. The model includes behavior models for tasks that describe the added value, the necessary effort and the decisions for following steps depending on process element properties. Of course, there is a trade-off between necessary modeling effort and certain process knowledge that have to be considered. To encounter this abstraction problem we detailed stochastic processes into descriptions of cause-effect relationships between existing process elements. A defined methodology was used to perform an impact analysis for a case study from the automotive industry. The benefit of a newly introduced automation method was quantified and thus an economical assessment has been enabled to support strategic decisions.
Further research is conducted on the validation and improvement of the methodology using additional case studies from different domains. Therefore we currently design a standardized interview guide. It has to be noted that certain types of investments are still hard to evaluate. For example the introduction of a knowledge management system requires a lot of initial effort and pays off after several developments. Also the benefit may depend on external factors that are difficult to express in the behavioral models. Further work has to be done to evaluate whether the modeling approach is suitable for such investments and how the behavior models have to be designed. Also accompanying research is done on specific work properties like quality and complexity to improve the understanding of the impact on the behavioral models.

REFERENCES

[1] J. Münch, D. Pfahl, and I. Rus, “Virtual Software Engineering Laboratories in Support of Trade-off Analyses,” Software Quality Journal, vol. 13, Dezember. 2005, pp. 407-428.

[2] N. Hinrichs, M. Olbrich, and E. Barke, “Performance Management and Optimization of Semiconductor Design Projects,” AIP Conference Proceedings, vol. 1247, no. 1, pp. 413-427, 2010.

[3] A. Hassine and E. Barke, “On Modeling and Simulating Chip Design Processes: The RS Model,” presented at the IEEE International Engineering Management Conference, pp. 81-85, 2008.

[4] M. Balazova, “Methode zur Leistungsbewertung und Leistungssteigerung der Mechatronikentwicklung,” Paderborn, 2004.

[5] T. K. Abdel-Hamid and S. E. Madnick, Software project dynamics : an integrated approach. Englewood Cliffs: Prentice Hall, 1991.

[6] M. Müller and D. Pfahl, “Simulation Methods,” in Guide to Advanced Empirical Software Engineering, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. Springer London, 2008, pp. 117-152.

[7] H. Zhang, B. Kitchenham, and D. Pfahl, “Reflections on 10 Years of Software Process Simulation Modeling: A Systematic Review,” in Making Globally Distributed Software

Development a Success Story, 2008, pp. 345-356.
[8] D. M. Raffo, “Getting the Benefits from Software Process Simulation,” in Proceedings of the International Conference on Software Engineering and Knowledge Engineering, 1999.
[9] S. Brinkkemper, “Method engineering: engineering of information systems development methods and tools,” Information and Software Technology, vol. 38, no. 4, pp. 275-280, 1996.
[10] F. Padberg, “Linking software process modeling with markov decision theory,” in Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International, vol. 2, 2004.
[11] E. W. Johnson, L. A. Castillo, and J. B. Brockman, “Application of a Markov model to the measurement, simulation, and diagnosis of an iterative design process,” in Proceedings of the 33rd annual Design Automation Conference, pp. 185-188, 1996.
[12] E. W. Johnson and J. B. Brockman, “Sensitivity analysis of iterative design processes,” in Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design, pp. 142-145, 1996.
[13] E. W. Johnson and J. B. Brockman, “Measurement and analysis of sequential design processes,” ACM Trans. Des. Autom. Electron. Syst., vol. 3, no. 1, pp. 1-20, 1998.
[14] S. Cho and S. D. Eppinger, “Product Development Process Modeling Using Advanced Simulation,” in Proceedings of DETC'01 ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2001.
[15] F. Deissenboeck and M. Pizka, “The Economic Impact of Software Process Variations,” in Software Process Dynamics and Agility, 2007, pp. 259-271.
[16] R. J. Madachy, “System dynamics modeling of an inspection-based process,” in Proceedings of the 18th international conference on Software engineering, pp. 376-386, 1996.
[17] I. Rus, S. Biffl, and M. Halling, “Systematically combining process simulation and empirical data in support of decision analysis in software development,” in Proceedings of the 14th international conference on Software engineering and knowledge engineering, pp. 827-833, 2002.
[18] D. Pfahl, “An integrated approach to simulation-based learning in support of strategic and project management in software organisations,” Kaiserslautern, 2001.
[19] I. Rus, H. Neu, and J. Münch, “A Systematic Methodology for Developing Discrete Event Simulation Models of Software Development Processes,” in ProSim Workshop 2003, 2003.
[20] B. Blanchard, E. Honour, J. Lake, and S. Weiss, “Engineering Vision 2020 – Ver. 1.5,” 2005.
[21] E. C. Honour, “Understanding the Value of Systems Engineering,” in Proceedings of the INCOSE International Symposium, 2004.
[22] E. Honour, “A practical program of research to measure systems engineering return on investment (SEROI),” in Proceedings of the Sixteenth Annual Symposium of the International Council on Systems Engineering, 2006.
[23] H. Zhang, R. Jeffery, and L. Zhu, “Investigating test-and-fix processes of incremental development using hybrid process simulation,” in Proceedings of the 6th international workshop on Software quality, pp. 23-28, 2008.
[24] E. W. Johnson and J. B. Brockman, “Measurement and analysis of sequential design processes,” ACM Trans. Des. Autom. Electron. Syst., vol. 3, no. 1, pp. 1-20, 1998.
[25] OMG, “Software & Systems Process Engineering Meta-Model Specification,” 2008.
[26] R. Koppe, S. Häusler, and A. Hahn, “Ein Modell zur Einflussanalyse von Änderungen in Entwicklungsprozessen,” in Informatik 2010, vol. 2, pp. 639-644, 2010.
[27] R. Koppe, S. Häusler, F. Poppen, and A. Hahn, “Process Model Based Methodology for Impact Analysis of new Design Methods,” in Modelling and Management of Engineering Processes, 1. ed., P. Heisig, S. Vajna, and P. J. Clarkson, Eds. Springer, London, 2010, pp. 53-64.
[28] R. Koppe, S. Häusler, R. Buschermöhle, and A. Hahn, “Process Change Impact Analysis Tool,” in Proceedings of the 1st International Conference on Modelling and Management of Engineering Processes, 2010.
[29] B. Boehm, C. Abts, and S. Chulani, “Software development cost estimation approaches — A survey,” Annals of Software Engineering, vol. 10, no. 1, pp. 177-205, 2000.
[30] S. große Austing and A. Hahn, “Complexity Measurement of Product Models,” in Proceedings of the International Conference on Knowledge Engineering and Ontology Development, pp. 404 – 407, 2010.
[31] R. Buschermöhle and J. Oelerink, “Rich meta object facility formal integration platform: syntax,

semantics and implementation," Innovations in Systems and Software Engineering, vol. 4, no. 3, pp. 249-257, 2008.