Managing Concurrent Engineering Design Processes and Associated Knowledge

Richard SOHNIUS[a], Vadim ERMOLAYEV[b], Eyck JENTZSCH[a], Natalya KEBERLE[b], Wolf-Ekkehard MATZKE[a], Vladimir SAMOYLOV[c]
[a] *Cadence Design Systems, GmbH, Mozart str., 2, 85622, Feldkirchen, Germany*
[b] *Zaporozhye National Univ., 66, Zhukovskogo st., 69063, Zaporozhye, Ukraine*
[c] SPIIRAS, 39, 14-th Liniya, St. Petersburg, 199178, Russia

Abstract. This paper presents our work in collecting and formalizing data about dynamic engineering design processes in microelectronics design projects. The paper reports on our experience in evaluating the design process knowledge acquisition routine by applying it to the real industrial project of Cadence Design Systems, GmbH. The methodology is based on the usage of the ontologies developed in the PSI project. Beside the actual process of formalizing the processes it especially focuses on capturing information about the concurrency among activities.

Keywords. Concurrent activities, Dynamic Engineering Design Process, ontology, agent, multi-agent system, knowledge acquisition, microelectronics

1. Introduction

It is well known that the design of microelectronic devices gets more and more complex [1] and to keep the design time and cost of such devices in reasonable boundaries, the productivity must be increased. Cadence Design Systems GmbH started the *Performance Simulation Initiative* (PSI) to help customers in finding and improving the weak spots in their design processes [1] and thereby increasing productivity. The approach is to examine *dynamic engineering design processes* (DEDPs) [2] by acquiring knowledge about them using a family of OWL[2] ontologies [2][3], and to employ the *multi-agent system* approach to simulate them [4]. The simulation results can then be used to assess past projects and to analyze how to optimize a given design system for future projects.

In this paper we focus on the methodology of acquiring and formalizing the information about DEDPs required for simulation using the PSI Ontologies. The paper is structured as follows: Section 2 surveys the related work in this field; Section 3 describes the general properties of design processes; Section 4 gives an overview of the ontologies used; Section 5 presents the different aspects of concurrency encountered; Section 6 explains the formalization process based on an example test case; Section 7 contains the conclusions and the plans for future work.

[1] Moore's law: every 18 months the processing power (of the product) doubles while cost holds constant.
[2] Web Ontology Language (OWL): http://www.w3.org/TR/owl-ref/

2. Related Work

The mainstream of formal business process modeling and engineering today is using PSL [5], PDL [6] or their extensions. Unfortunately, these formal flow modeling frameworks do not fully allow breaking down the diversity of the processes encountered in real life. This diversity may be characterized for example by Sandewall's taxonomy [7] providing the basic features of the processes. This classification embraces highly predictable, normal, manufacturing processes at one side and stochastic ("*surprising*" [3]), structurally ramified, time-bound processes characteristic for design domain, on the other side of the spectrum.

In our search for the proper upper-level ontological paradigm for DEDP modeling we examined the SUMO and DOLCE [8] ontologies as candidates for the formal description framework. In DOLCE a process is the sub-category of a perdurant. Perdurants also comprise events, phenomena, activities, and states. The semantics of a DEDP in our framework is aligned with DOLCE in this sense. However, we consider DEDPs to be the aggregations of the simpler occurrences: sub-processes, events, states, activities. As in DOLCE we assume that subjects and objects are not parts of processes, but rather participate in them or are transformed by them.

The conceptual framework and the PSI family of ontologies [3] used to model DEDPs are the follow-up of our results published in [2]. The DEDP modeling framework in its part of organizational and actor-related knowledge representation is based on the frameworks [9][10][11], extends them by incorporation of roles, abilities, actors, dynamic teams of actors, and its binds to the engineering design domain. In the part of process modeling PSI bases its approach on [12], [13], [14]. One advancement of the PSI ontologies family is the classification of the concurrency types as the co-execution types [3]: *sequential, partially overlapping at start (end), inner overlapping, parallel.*

3. Concurrent Engineering Design Processes in Microelectronics

The processes of designing microelectronic devices tend to be executed as parallely as possible. The design tasks are too time-consuming to be executed in sequence since time to market is one of the critical aspects of the process. Especially in the consumer electronic space, certain deadlines (e.g. Christmas) must be met to survive in business. In order to be able to exploit concurrency where possible, it is essential to model the dependencies between tasks as detailed as possible.

Another important aspect of these design processes is the lack of predictability. No two devices are designed exactly the same. Therefore, the simulation can only give an approximate estimation of a process flow and the time it requires. As a result, it has to be able to adapt as the dynamic system develops. We use DEDP-MAS [4] to simulate human-like behaviour and to provide the required flexibility.

According to Sandewall's taxonomy [7] a DEDP may be considered a *discrete, branching, structurally ramified, concurrent, surprising, equinormal, hierarchical, time-* and *memory-bound* process with *delayed* effects. This combination of the features makes DEDP modeling a challenging task. The starting point for our modeling

[3] A process is considered "surprising" if it is allowed that a *surprising* or *exogenous* event may cause a change that is not anticipated by the process script [7].

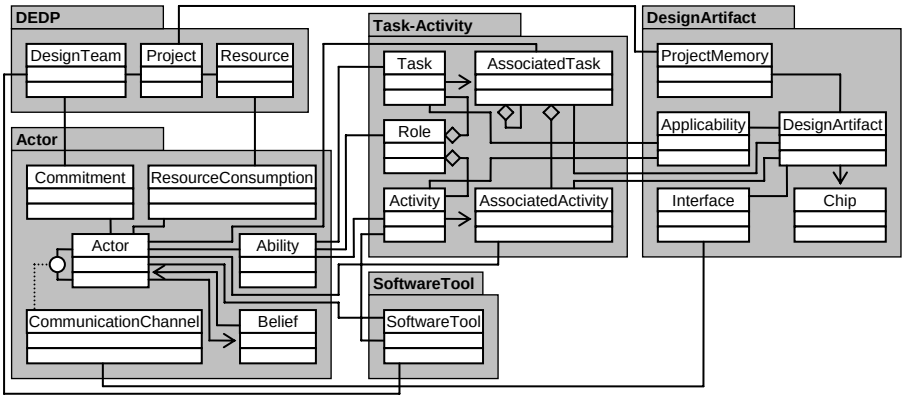

Figure 1. PSI Ontology overview

framework was the observation that an arbitrary engineering design process is the configuration of the following basic building blocks: a goal (the state of affairs to be reached); an action; an object to apply actions to; a subject who (or which, if unanimated) applies actions to objects; an instrument to be used by a subject to execute actions; and an environment in which the process occurs. This structure at the first glance resembles the one of PDI (Process Definition Interface [6]).

4. Expressing DEDP Knowledge Using PSI Ontologies

The PSI family of ontologies comprises five tightly linked ontologies which in UML representation are grouped in separate packages: the Actor Ontology (a subject), the DEDP Ontology (an environment), the Task-Activity Ontology (an action), the Software Tool Ontology (an instrument), and the Design Artifact Ontology (a goal and an object). The classes shown within the packages in Figure 1 identify the major concepts of the respective ontology [3].

The instances created using the ontologies are grouped by their level of abstraction. The most abstract level is the knowledge base. It contains the information on the types of design artifacts that can exist and the activities that can be executed on these design artifacts. This level is also considered common knowledge and is the same for all projects examined. The next less abstract level is the project definition. It comprises all the knowledge and data required to simulate the project: the structure of the design artifact, the design team, its members and their skills, the available resources, the desired project output and the time frame. Finally, the last level is the project plan. It contains all the activities executed (or to be executed), their schedule and the assignment of the designers and the resources to them.

One of the facets of activity descriptions used in PSI (unlike PSL [5] or TOVE [9]) is the description using the design artifact input and output representations. An activity transforms a design artifact from one representation to another. For example *Initial RTL*[4] *Development* transforms a functional block in the representation *Specification* to

[4] RTL (Register Transfer Level), functional description of a digital block in terms of logical and sequential equations.

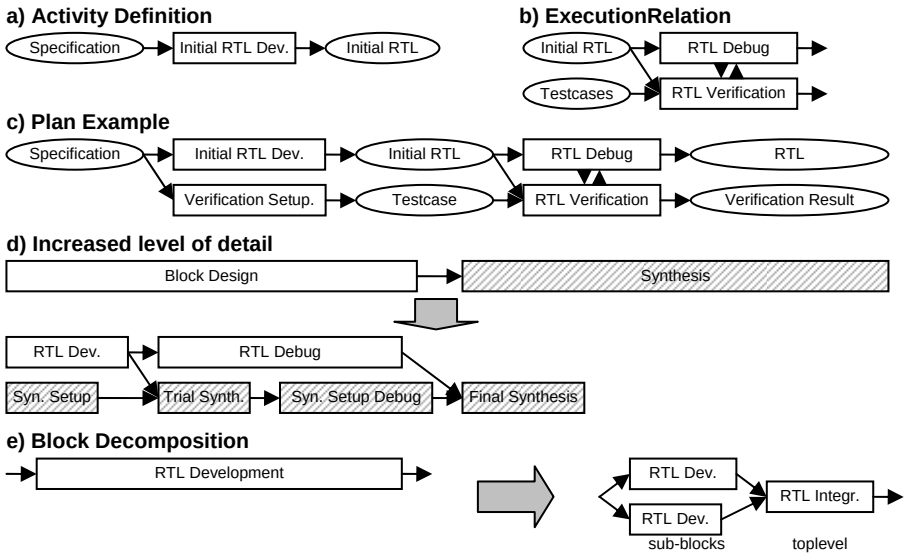

Figure 2. Activity representation and concurrency

the representation *Initial RTL* (see Figure 2a). Therefore, the simulation does not rely on predefined plans but can link activities by the representations of their inputs and outputs (see Figure 2c) to build a graph which shows every possible way from the initial input representations to the target output representations and can then choose the optimal one. Activities have a property *difficulty* which is used to estimate the required design time. The result of the linking of input representations to output representations is called the IO-chain.

Additionally, the appearance of an activity is constrained by its applicability which describes to which type of a design artifact the activity can be applied. You cannot for example apply *Initial RTL Development* to an analog block.

Finally, we model co-activities by introducing an *ExecutionRelation* which allows restricting the co-execution type between two activities. The problem is that some activities exchange information and therefore must be executed (partially) in parallel. For example *RTL Debug* and *RTL Verification* need to be executed in parallel as the verification requires the debugged RTL and the debugging requires the verification results (see Figure 2b).

5. Concurrency

In a DEDP the activities which do not depend on each other are executed concurrently in order to reduce the total development time. Additional concurrency can be gained by accounting more process details (Section 5.1) or by decomposition of blocks (Section 5.2).

5.1. Improving concurrency using finer-grained process model (knowledge base)

One of the methods to allow for increasing the process concurrency is by increasing the level of detail in the knowledge base. If, at the knowledge base development stage, it is found out that a meaningful action viewed as an activity can be represented as the composition of several other activities, the knowledge base should be adjusted accordingly. Some of these activities may be executed concurrently and their introduction may therefore add to the concurrency factor. For example, at an early modeling stage *Block Design* and *Synthesis* tasks could be seen as coarse activities which have to be executed in sequence due to their input/output representations. When the knowledge base is refined both are modeled by the compositions of several activities (Figure 2d). After that both tasks can be executed partially in parallel.

Since the majority of design tasks are typical in the industry branch the knowledge base appears to be a valuable knowledge asset which may be efficiently reused and transferred.

5.2. Concurrency by block decomposition

Another way to gain improved concurrency is to perform an activity on each sub-block and integrate the results instead of performing the activity on the whole design. This also reduces the difficulty of each activity compared to the original one as only the complexity of the associated sub-block impacts each design activity and the integration activity depends only on the interconnection of the sub-blocks. Figure 2e shows this method using the example of *RTL Development*.

6. The Test Cases: Knowledge Acquisition Using PSI Ontologies

The ontologies were initially evaluated with different test cases. At the beginning simplified artificial data was used (e.g. [15]). The simulation of DEDPs based on these simplified initial test cases has been reported in [4]. In this paper we shall report on the follow-up work with one of the real projects run at Cadence Design Systems, GmbH.

The knowledge acquisition and DEDP modeling scheme for real projects is as outlined in Figure 3. The project knowledge is first homogenized and formalized. Then the project definition is extracted. After that the acquired knowledge is used to re-plan and re-schedule the project by simulations. These results are then compared to the original plan and schedule developed by the project manager manually. In this paper we describe the left branch of the routine: starting from the raw sources and ending up by the project definition (steps 1-3 in Figure 3).

6.1. The Test Case

The test case described here is a real project which created an MMC/SD[5] host controller Soft IP[6]. The available input consists of the project plan in the form of a

[5] MMC (Multi Media Card) / SD (Secure Digital Card), flash memory card formats.

[6] Soft IP (intellectual property) is a predefined functional block which is incorporated in other designs. In contrast to Hard IP which comes in the form of layout data, Soft IP is delivered as a functional description which still needs to be mapped to a specific silicon technology.

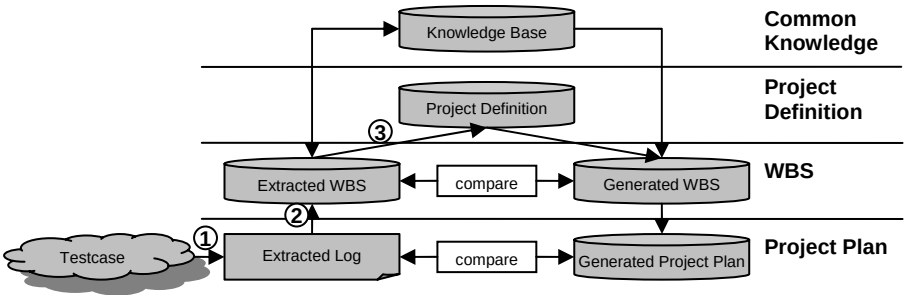

Figure 3. Test case knowledge acquisition and usage

GANTT-diagram with resource assignments, a block-diagram and a verbal description of the designers by the project leader. The project was chosen because it is small enough to be formalized relatively quickly and still contains the typical pitfalls (see Sections 6.2, 6.3) which were not present in the initial simplified test cases. In detail, it contains 4 digital blocks and 34 tasks executed by 3 designers. The resulting WBS had 30 activities.

6.2. Formalizing the Design Process Log

The first step in formalizing the project knowledge was the extraction of the required information from the sources and bringing it to a homogenous representation – a spreadsheet was used to hold the data. This is the step 1 shown in Figure 3. The main problem at this step is to associate each task with one particular block of the design.

The resulting process log was incomplete and ambiguous. In the available knowledge sources, several smaller tasks had been omitted or grouped together in the project plan to keep the diagram clearer to the manager whereas the bigger ones were split to show the different facets of a task. There was for example no explicit verification of one of the functional blocks (*register file*) as it was always verified in conjunction with the other blocks. On the other hand, there were several test case development tasks for the command path - one per command mode. This was mainly because the project was structured differently than intended by the ontology and the model. The project leader had grouped the tasks only by their similarity without respect to the block structure while the model organizes tasks by both aspects and requires a strict association of tasks to blocks.

6.3. WBS Extraction

The next step was the extraction of a coherent WBS from the log (Figure 3 step 2). This required the logged tasks to be mapped to activities described in the knowledge base (Figure 4). If the project contains tasks which cannot be represented with the activities in the knowledge base, the knowledge base must be extended. In the given case, activities for FPGA-emulation and coverage analysis were added.

This step required some tasks to be split and others to be merged to remove simplifications in the original project plan. Consequently, the tasks which had been omitted in the project plan were introduced at this step to complete the IO-chain. Finally, the missing data had to be supplemented. This work normally requires the

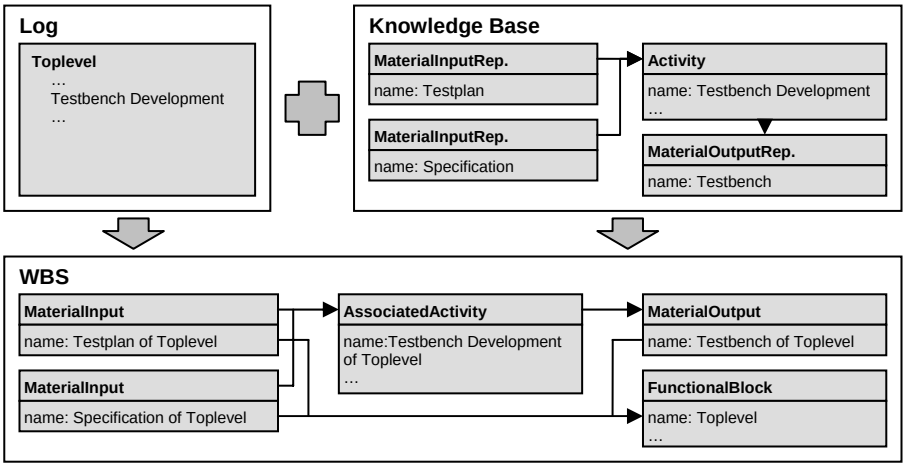

Figure 4. Task mapping example for Testbench Development

profound knowledge of the domain and the project itself and is best done with the help of the project leader or the designer working on the particular part of the project. In this particular case the project leader was available as domain expert.

To reuse the previous example, the test case development tasks for the command path were merged into one activity..The example of splitting a task was the synthesis. Due to the simplicity of the design the source project plan did not further refine the synthesis task. But the knowledge base suggested the composition of the four activities instead - '*Synthesis Setup*', '*Trial Synthesis*', '*Synthesis Setup Debug*' and '*Final Synthesis*'. Thus, the original task was split in four activities. Finally, small verification activities applied to the register file were introduced to complete the IO-chain.

6.4. Extracting the Project Definition

This is the last step in the knowledge acquisition routine (Figure 3 step 3). Acquiring project data like starting date, end date and required output is straight-forward as is the description of block structure. The assessment and modeling of designers' abilities and the complexity of blocks is a rather difficult topic and will be a major part of our future research. For now we are working with the simplified model and the acquisition is currently relying on the judgement of the designers only.

7. Conclusions and Future Work

With the current version of the PSI Ontologies we were able to formalize a real life project and we could even reuse the initial knowledge base with only minimal additions. This shows that the modeling of activities by their inputs and outputs with a few additional constraints is feasible and that it is suitable for the application domain of semiconductor design.

To be able to improve the modeling of dependencies between activities, we still need to collect a comprehensive set of test cases. With this done, we will concentrate

on new concepts and properties to create detailed models of the complexity and the quality requirements of design artifacts, the difficulty of activities, the abilities of designers and the capabilities of software tools used. In parallel, ongoing work is focusing on improving the simulation and the evaluation of the results using a growing testbed and assessing real design systems.

8. References

[1] Matzke, W.-E.: Engineering Design Performance Management – from Alchemy to Science through ISTa. In: Kaschek, R., Mayr, H.C., Liddle, S. (Eds.) Proc. 4th Int. Conf on Information Systems Technology and its Applications (ISTA'2005), 23-25 May 2005, Palmerston North, New Zealand, p 154-179

[2] Ermolayev, V., Jentzsch, E., Karsayev, O., Keberle, N., Matzke, W.-E., Samoylov, V.: Modeling Dynamic Engineering Design Processes in PSI. In: J. Akoka et al. (Eds.): ER Workshops 2005, Proc. Seventh International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2005), Klagenfurt, Austria, October 24-28, Springer LNCS 3770, pp. 119 – 130, 2005

[3] Ermolayev, V., Jentzsch, E., Keberle, N., Samoylov, V., Sohnius, R.: The Family of PSI Ontologies. Reference Specification. Technical Report. Cadence Design Systems, GmbH, 47 p., 2006

[4] Gorodetsky, V., Ermolayev, V., Matzke, W.-E., Jentzsch, E., Karsayev, O., Keberle, N., Samoylov, V.: Agent-Based Framework for Simulation and Support of Dynamic Engineering Design Processes in PSI. In: Pechouchek, M., Petta, P., Varga, L. Z. (Eds.) Proc. 4th Int. Central and Eastern European Conf. on Multi-Agent Systems (CEEMAS'05), 15-17 September 2005, Budapest, Hungary, LNAI 3690, pp. 511-520, 2005.

[5] Conrad Bock, Michael Gruninger, PSL: A semantic domain for flow models. Software and Systems Modeling Journal (2005) 4: 209–231 / Digital Object Identifier (DOI) 10.1007/s10270-004-0066-x

[6] Workflow Management Coalition. Workflow Standard. Process Definition Interface -- XML Process Definition Language. V. 2.00, Doc No WFMC-TC-1025 (Final), October 3, 2005

[7] Erik Sandewall, Features and Fluents. The Representation of Knowledge about Dynamical Systems. Oxford University Press, 1994

[8] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening Ontologies with DOLCE. In: A. Gómez-Pérez , V. Richard Benjamins (Eds.) Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web: 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002

[9] Grueninger, M., Atefi, K., Fox, M. S., Ontologies to Support Process Integration in Enterprise Engineering Computational & Mathematical Organization Theory 6, 381–394, 2000.

[10] Uschold, et al: The Enterprise Ontology. Knowledge Engineering Review, 13(1), 1998

[11] Ermolayev, V. Keberle, N. and Tolok, V.: OIL Ontologies for Collaborative Task Performance in Coalitions of Self-Interested Actors. In: H. Arisawa, Y. et al (Eds.): Conceptual Modeling for New Information Systems Technologies ER 2001 Workshops, Yokohama Japan, November 27-30, 2001. LNCS vol. 2465, 390-402, 2002.

[12] Buhler, P. and Vidal, J.M.: Enacting BPEL4WS specified workflows with multiagent systems. In Proc. of the Workshop on Web Services and Agent-Based Engineering, 2004

[13] Ermolayev, V., et al: Towards a framework for agent-enabled semantic web service composition. Int. J. of Web Services Research, 1(3): 63-87, 2004.

[14] Fensel, D. and Bussler, C.: The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications 1(2): 113-137, 2002.

[15] Jentzsch, E., Matzke, W.-E.: Case Study of a Digital Design Process. VCAD EMEA Cadence Design Systems GmbH. May 17, 2004