Modeling Dynamic Engineering Design Processes in PSI

Vadim Ermolayev[3], Eyck Jentzsch[1], Oleg Karsayev[2], Natalya Keberle[3], Wolf-Ekkehard Matzke[1], Vladimir Samoylov[2]

[1]Cadence Design Systems, GmbH, Feldkirchen, Germany
{wolf, jentzsch}@cadence.com
[2]SPII RAS, Saint Petersburg, Russia
{ok, samovl}@iias.spb.su
[3]Zaporozhye National Univ., Zaporozhye, Ukraine
{eva, kenga}@zsu.zp.ua

Abstract. One way to make engineering design effective and efficient is to make its processes flexible – i.e. self-adjusting, self-configuring, and self-optimizing at run time. The paper presents the descriptive part of the Dynamic Engineering Design Process (DEDP) modeling framework developed in PSI[1] project. The project aims to build a software tool assisting managers to analyze and enhance the productivity of the DEDPs through process simulations. The framework incorporates the models of teams and actors, tasks and activities, design artifacts as the major interrelated parts. DEDPs are modeled as weakly defined flows of tasks and atomic activities which may only "become apparent" at run time because of several presented dynamic factors. The processes are self-formed through the mechanisms of collaboration in the dynamic team of actors. These mechanisms are based on several types of contracting negotiations. DEDP productivity is assessed by the Units of Welfare collected by the multi-agent system which models the design team. The models of the framework are formalized in the family of DEDP ontologies.

1 Introduction

It is widely accepted that the processes of engineering design differ from manufacturing processes by the fact that they "… are frequently chaotic and non-linear, and have not been well served by project management or workflow tools" (cf. [NSB01]). The primary reason is that the ability to design is one of the signatures of human intelligence which can hardly be framed by the rigid and static bounds of pre-defined business processes. Therefore one of the promising ways to make engineering design effective and efficient is to manage its processes in a flexible manner – i.e. make them self-adjusting, self-configuring, and self-optimizing at run time. By doing so we may enhance the degree of coherence between the interrelated activities and make them better coordinated and therefore more productive. Hence, the model of a

[1] Productivity Simulation Initiative (PSI) is the R&D project of Cadence Design Systems GmbH.

DEDP should be at least capable to account for the constellation of the factors which make a DEDP "chaotic and non-linear" and, at most, to eliminate them as much as possible. Provided that we have built such a fine-grained DEDP modeling framework, we may expect implementing software tools allowing to assess a process and, ultimately, to optimize DEDPs in terms of engineering design productivity.

Improving DEDPs in terms of engineering design productivity is the focus of PSI project. The project have prototyped a software tool which provides for the assessment of the accomplished DEDPs and the prediction of the characteristics of the planned DEDPs through their simulations. This simulation prototype has been implemented as a multi-agent system [Go05][2] which models designers' teams working on projects by dynamically formed teams of software agents, DEDPs performed by these teams – by tasks, and the results of these processes – by design artifacts. The knowledge on the performed processes is formalized and stored to PSI testbed in terms of DEDP family of ontologies presented in this paper. Through that we obtain the incremental collection of actors' experience which is further on used to make simulation results more reliable.

The paper is structured as follows. Section 2 discusses modeling requirements justifying the necessity to cope with the dynamic character of DEDPs. Section 3 outlines our approach to assessing the productivity of DEDPs. Then PSI Actor model, Task-Activity model and Design Artifact model are presented in Sections 4 – 6. Section 7 deals with the epistemological aspects of DEDP ontologies family and the usage of these ontologies in PSI in the form of DEDP-lite ontologies. Section 8 surveys the related work and analyses the contributions of DEDP modeling framework.

2 The Model of a Dynamic Engineering Design Process

A DEDP is the process of aiming a weakly defined engineering design workflow to achieve its goal in an optimal way in terms of result quality and gained productivity. It is therefore clear that the following entities are involved in the process: actors, who form design teams and collaboratively do the work in the flow; activities which are the atomic parts of a workflow defined by the technology used in the house; tasks which are subjective actors' representations of activities' compositions and choreography[3]; and design artifacts which are the results of engineering design activities. Hence, only engineering design activities are defined by the design technology and are well known before a DEDP starts. Other elements may only "become apparent" at run time because:

– The treatment of a task as atomic or composite is different by the actors having different capabilities. A task which is perceived as an atomic activity by one actor may be recognized as composite by another actor.

[2] [Go05] is the parallel paper which reports on the implementation and the evaluation experiments with PSI simulation prototype. In this paper we omit the description of this important part of our research due to space limitations.

[3] Choreography in the mentioned context is understood similarly to Web Services choreography and means the way of arranging material input – output communication among the dependent activities.

- The composition [Er04] of the activities is defined only subjectively and partially. Tasks in our model may be composed of the activities and other tasks in different ways by different actors having different knowledge. It implies that the sequence of activities and sub-tasks in a task may be understood differently in the partial local plans of different actors.
- The number of activity loops is not defined in advance. It depends on the quality checks at intermediate steps. Changing the number of activity loops may cause the changes in its duration. In turn, it may cause the delays of the dependent tasks and activities with associated penalties for, e.g., deadline violation.
- The duration of activity execution is not defined in advance. Different actors possess different capacities to be spent for the activity at a certain time. They may perform the same activity with different efficiency (productivity – Section 3). An activity may remain idle while waiting until the pre-conditions have been triggered. Idle state duration can't be computed in advance because the preconditions may be formed by the other activities executed by other actors.
- The actors are not assigned in advance to perform certain activities. An actor is chosen by the Task Manager when s/he decides to assign the activity. In PSI framework contracting negotiations are the means to optimally choose the actor to perform the task. DEDP model should therefore incorporate the actor model and the means to arrange actors' collaboration through peers' assessment and negotiations.

Mentioned factors provide certain degrees of freedom[4] in DEDP planning, re-planning, scheduling, re-scheduling, and execution. In PSI a DEDP is not rigidly planned before it starts. The decisions on how to continue its execution are taken each time it reaches a certain state in the state space. These decisions are taken by the design team members (Section 4) who manage the tasks which continue the process. According to the aforementioned properties of a DEDP different paths through the state space may be more or less productive.

As shown in Fig. 1 a DEDP has components which differ along the dimensions of their changeability. The first dimension is the dynamic character ranging from static, i.e. pre-defined for all possible DEDPs, to dynamic, i.e., subjected to changes in a DEDP. Another dimension is the sphere of visibility or commitment. It ranges from shared, i.e., having the same meaning and instances for all DEDP participants, to subjective, i.e. having specific instances for different actors (though in the terms of a common ontology). Static shared DEDP components are atomic activities, associated software tools, and resources. It is assumed that the processes are assembled (ultimately) of atomic activities which are the pieces of the design technology used by the company. The technology normally provided by a design support unit often suggests the usage of a specific software tool to perform the activity. The execution of a given activity consumes certain resource instances in given quantities. The model of a design process is based on the following assumptions. A DEDP is initiated by an external influence providing a goal to a certain actor. This goal is subjectively transformed to a task according to the knowledge of this actor. The actor uses his or her subjective knowledge about the composition of the task, i.e. about the sub-tasks

[4] It should be noted here that this freedom implies more complications in planning, scheduling and the necessity to deal with finer grained DEDP model.

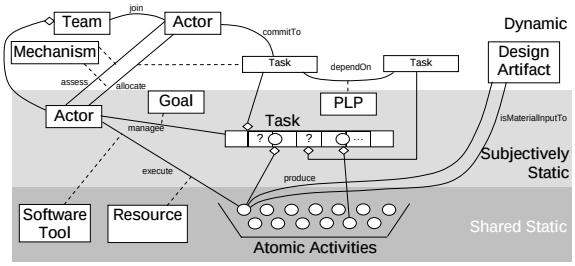

Figure 1. Static and dynamic components of DEDP modeling framework.

and the atomic activities to be performed within the given task. The dependencies between different tasks are also the subjective knowledge of an actor and are formalized in his or her Partial Local Plans (PLP). The actor may decide to perform a sub-task or to execute an activity of a decomposed task himself or to hire (for the price) another actor through the available collaboration mechanism (contract net negotiations in PSI). In the latter case the sub-task becomes the goal of another peer-actor who commits himself to perform the corresponding task by striking the contract deal. Hence the appearance of actor-task combinations in a DEDP is subjectively dynamic. The mechanism of incorporating new actors to the process and the model of the design team are subjectively dynamic as well since they depend on the decisions and choices taken at run time by the actors which states change in the process. The rules of encounter of the mentioned mechanism are shared static and provide the horizontal laws for the system [Er04, EJM04]. A design artifact is a subjectively dynamic outcome of the process since it is formed out by subjectively dynamic collaborative team of actors. However, the proposed layering allows reaching this effect through applying shared static atomic activities, though in subjectively dynamic combinations. For an activity a design artifact is both the material input and the result of its execution.

The actors who perform a task and initiate collaboration are Task Managers. Their rational goal with respect to the performed task is to choose the next step on the process path as productive as possible. Of course an actor needs a sort of productivity assessment model for that (Section 3).

3 Assessing Productivity by the Earned Units of Welfare

Productivity by its very nature is one of the most important economic metrics and stands for the ratio of the produced output (value) to the consumed input (value). As such it is an integral characteristic of any transformation process, e.g. a DEDP. This neo-classical definition of productivity imposes rigid requirements on the process under consideration. The homogeneity of inputs and outputs is the most severe one with respect to engineering design. Known productivity measurement methodologies in engineering design ground themselves on the assessment of design complexity characteristics in the creation of homogeneous input- and output-measures. They pretend to do it by applying heuristic weights to compared parameters (e.g., the normalized transistor count[5] in Semiconductor and Electronic Systems (SES) design,

[5] Measuring IC and ASIC Design Productivity. White Paper. Numetrics Management Systems, 5201 Great America Parkway, Suite 320 Santa Clara, CA 95054, 2000

FP, KSLOC counts[6] in software design, etc.). The fundamental problem of this approach is that the complexity characteristics need to be invariant both to the type of a process and to the transformed design artifact. If they aren't, measurement scales tend to lack well-defined units. Consequently the properties of the measurement scale, the labeling of the units, and the interpretation of the values derived are of very limited practical use. Furthermore in non-deterministic environments such measures are not very reliable even if proposed. It is therefore important to build a measure which addresses the homogeneity requirement with respect to inputs and outputs and which is invariant to the dynamic characteristics of a process (Section 2). Such a measure may be based on the integral process success indicators like for example the ratio of the Earned Value to the Planned Value or to the Actual Cost at a Sign-off Stage of the process. This implies that productivity of a DEDP may be assessed by the value produced and accumulated by designers in a team. The more value produced by a designer – the more relatively productive s/he is. It is also true in a longer run if several DEDPs are taken into consideration. Hence more productive designers are characterized by the higher volume of accumulated Units of Welfare (UoW) if designers are incentivized adequately to their produced value (assumed in PSI). This characteristic is invariant to all aforementioned dynamic features of an engineering design process. UoW is a normalized scalar measure which by its semantics is similar to the notion of a Utility which is used in Distributed Rational Decision Making. UoW earning mechanisms in PSI are based on contracting deals stricken through several types of negotiations [EJM04].

4 Actors, Teams, Beliefs, and Negotiations

Actors and related concepts are denoted by DEDP Actor ontology which is outlined in Fig. 2. An Actor is the abstraction of a person who performs Tasks and executes atomic Activities which result in the transformation of Design Artifacts. An Actor as the part of an organization plays Organizational Roles which are regulated by organizational Policies. An Organizational Role is the subclass of an abstract Role. A Role specifies the set of requirements to an Actor with respect to his or her capability to execute Activities. Thus Organizational Roles and Policies constitute the organizational framework of DEDP model. A Collaboration Role is another subclass of a Role specifying the Roles of Actors with respect to their encounters governed by PSI Negotiation Mechanisms defined by interaction protocols, negotiation sets, and negotiation strategies. Therefore another important aspect covered by DEDP Actor ontology is Collaboration and Team Formation framework. Chosen collaboration mechanisms based on contracting negotiations (full details are in [EJM04]) imply the appearance of the following subclasses of an Actor: a Task Manager and a Believed Performer. A Task Manager intends to out-source a Task to one of his or her peers. The following two aspects constrain the set of peers to the sub-set of the Believed Performers: a Task Manager believes that the Believed Performers are (1) Capable to perform the Task and (2) Credible enough to trust the performance of the Task to them. These Beliefs in PSI are (1) formalized by Belief sub-ontology and (2) adjusted

[6] FP stands for Functional Point, KSLOC – for kilo lines of source code.

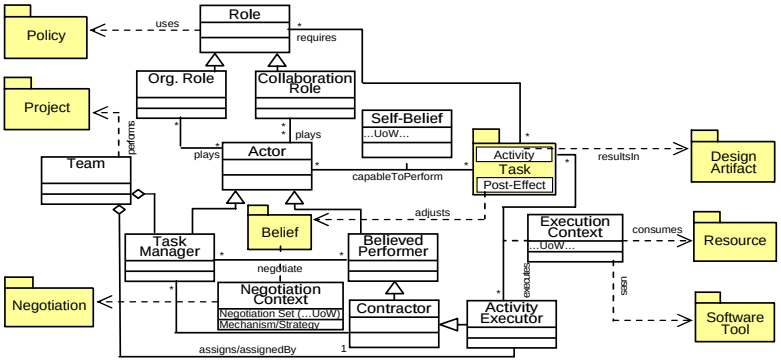

Figure 2. The outline of DEDP **Actor** ontology.

by the Post-Effects of Activities (Section 5) through capability and credibility assessment mechanisms adopted from RACING [Er04]. A Contractor is the sub-class of a Believed Performer as s/he is the only one of Believed Performers who receives the negotiated Task and commits him- or herself to perform it according to the commitment-convention framework [EJM04]. If a Contractor according to his subjective knowledge decides that the received Task comprises only one atomic Activity[7] then s/he becomes an Activity Executor. S/he also becomes the member of the design Team by committing him- or herself to the Activity execution. A Team is therefore formed of Task Managers and Activity Executors through contracting negotiations. Conceptually a Team is the bridge providing the relationship of a DEDP to the Project which is implemented through this DEDP.

It is assumed in PSI that collaboration mechanisms are based on three types of negotiations which use one basic protocol (extended FIPA CNP) but differ by negotiation sets and strategies [EJM04]: (1) on Task allocation; (2) on Design Artifact re-use; (3) on Software Tool provision. PSI Negotiation ontology based on [EKT02] is used as the namespace for formalizing Negotiation Contexts in all negotiation types.

The central property of an Actor is the capability to perform Tasks. An Actor is capable to perform Tasks in frame of the Organizational Role he plays in the sense that s/he has the subjective knowledge on the following: (1) if the certain Task is a *composite* one or it contains only one an *atomic* Activity; (2) if s/he can perform this Task *by himself* or he should *allocate* it to another Actor paying a price in UoW. This knowledge constitutes Actor's Self-Beliefs. Another portion of subjective Task-related knowledge is formalized by the DEDP Task ontology (Section 5). However the Actor ontology provides for the clear separation between the notions of a Task and an Activity. A Task is *performed* – i.e. arranged and managed by Task Managers. An Activity is *executed* by Activity Executors – physically: using Design Artifacts as material Inputs and Software Tools as instruments, consuming Resources, producing material Outputs in the form of Design Artifacts. These aspects are captured by Execution Context concept of the Actor ontology. UoW are spent by Activity Executors for lending Software Tools and using Resources.

[7] As the Contractor believes.

5 Tasks, Activities, and Partial Local Plans

DEDP Task-Activity model provides formal shared static description framework (Fig. 3) used in the knowledge models of Actors[8] to form their subjective static knowledge on Task compositions, Activity choreography, and Task dependencies. This knowledge according to the Task-Activity model is tightly linked to the Belief and Self-Belief parts of DEDP Actor ontology.

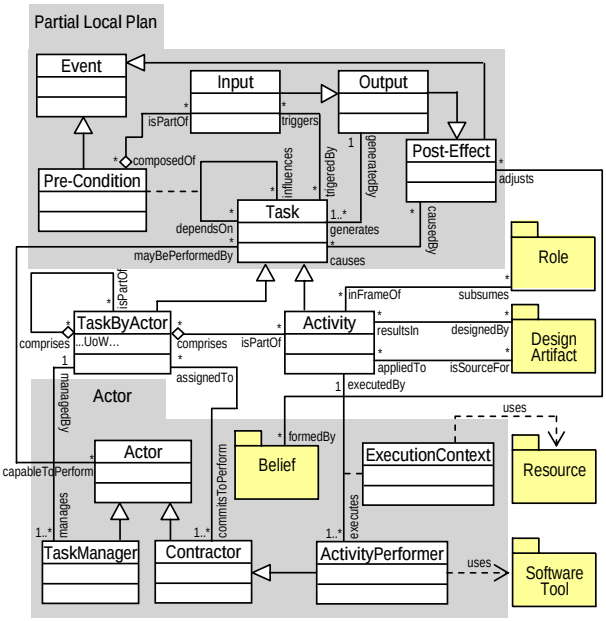

Figure 3. Outline of DEDP **Task-Activity** ontology.

An Activity is the basic process building element which is shared static (defined by the design technology) and is treated as objectively atomic by all DEDP participants. Material Inputs and Outputs of Activities are also fixed by the technology and are Design Artifacts. Task-Activity model provides corresponding relationships. An Activity as the basic building element is the sub-class of a Task concept. In difference to a Task an Activity is the only piece of a DEDP which is executed and produces material Outputs. A Task is subjectively static as the representations of the compositions of the same Task may differ from Actor to Actor. This is one of the explicit reasons to introduce a TaskByActor concept as the sub-class of an abstract Task. A Task is linked to an Actor by the capability relationship with the associated Self-Belief context. In difference to an abstract Task a TaskByActor is associated with a Task Manager. Thus its semantics becomes even more subjective in the sense that it is the Task which is managed and, therefore, can not comprise only one atomic Activity. A TaskByActor is the Task to which the Task Manager has committed him- or herself by striking the deal in the contracting negotiations. Hence, a TaskByActor (but not a Task) has UoW property associated with it. UoW property of a TaskByActor reflects the result of the negotiations on this very task providing the Contractor with the budget figure.

A Task in difference to an Activity is managed. Task management comprises the proper scheduling of its sub-Tasks which requires the knowledge on the dependencies among these sub-Tasks. In the Task-Activity model Tasks may be *independent* or

[8] Actors are modeled by economically rational software agents in PSI DEDP Simulation Prototype [Go05].

strongly dependent on other Tasks. The model also indirectly allows coping with *facilitations* (or *weak dependencies*). Task t_1 is said to be independent of Task t_2 if the performance of t_2 does not depend of t_1 performance or of the results of t_1 and vice versa. The task t_1 is said to be *strongly dependent* of task t_2 if the results of t_2 are essential to start the performance of t_1. Finally, task t_1 is said to be *facilitated* by task t_2 if the performance of t_2 or the results of t_2 may help to execute t_1 in less time, with less resources consumed or obtaining better quality with the same resource consumption. From productivity viewpoint facilitation means UoW savings. For example getting a proper Design Artifact from a fellow for re-use may facilitate to the design of the similar Design Artifact resulting in effort, resource (and therefore UoW) savings. Hence, fine-grained knowledge on Task dependencies allows making the process properly coordinated and therefore more productive.

One more important aspect captured by the discussed model is the subjectivism of dependencies' representations in the PLPs of different Actors. Dependency plans are denoted as partial local because different Actors: (1) have different knowledge on Task dependencies – these pieces of knowledge are the subjective parts of the whole picture possibly leading to alternative paths in DEDP state space; (2) do not use the knowledge of other Actors in task planning and scheduling – i.e. take their decisions locally or autonomously.

Task-Activity model handles dependencies among Tasks based on the assumption that the existence of a strong dependency between t_1 and t_2 implies that the material Outputs of t_2 are required as material Inputs to t_1 before t_1 starts. Therefore the Pre-condition of t_1 is that the events of the appearance of all the necessary Inputs (eventual Inputs to be shorter) have all took place thus triggering t_1. A weak dependency is based on the same triggering mechanism through the eventual Inputs. However in the latter case the trigger just lowers the amount of UoW required for managing the dependent task reflecting that the facilitation has occurred. PLP part of the Task-Activity ontology frames out the sets of eventual Inputs as Pre-conditions. It is stated that an eventual Input is the sub-class of an eventual Output because only some outputs may become inputs. An eventual Output is in turn the sub-class of a Post-effect. A Post-effect is the abstraction of the changes implied by the Task onto its environment. With respect to a DEDP Post-effects are not only the eventual Outputs but also the events caused by Task re-planning and re-scheduling like deadline violations. Consequently, Post-effects cause the changes in Actors' Beliefs (Fig. 3). Eventual Inputs, Outputs, and Post-effects are ultimately the sub-classes of an abstract Event concept.

6 Design Artifacts and Project Memory

The purpose of PSI Design Artifact model is twofold: (1) it provides the grounding to SES design domain and (2) it reflects the project-oriented nature of engineering design. DEDP Design Artifact ontology is outlined in Fig. 4. From the point of view of domain grounding the model specifies that a Design Artifact comprises the hierarchy of Functional Blocks as the structural elements of designed functionality. Functional Blocks are generally viewed as "gray boxes" with functional subdivision to digital, analog and mixed-signal blocks according to the function and components

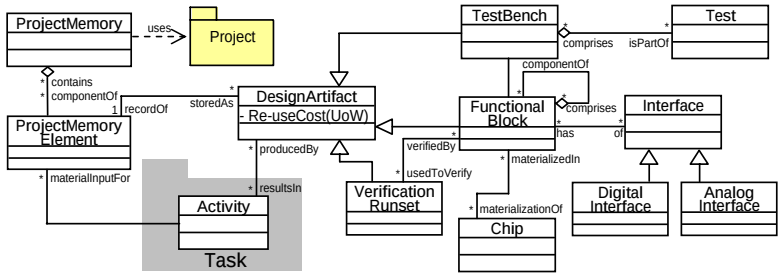

Figure 4. Outline of DEDP **Design Artifact** ontology.

used in their design. Therefore the Interfaces of Functional Blocks are of type Digital or Analog. A Functional Block of mixed functionality may have Interfaces of both types. The Functional Block of the topmost level is finally materialized in the corresponding Chip. The description of a Chip ready for production is considered the terminal output of a DEDP. Functional Blocks are complemented by TestBenches and Verification Runsets – the means to test and verify designs according to the provided engineering design technology.

The Design Artifact model provides the formal frame for handling material Inputs and Outputs of DEDP Activities. It is considered that a Design Artifact is the material Output of an Activity (through resultsIn – producedBy relationship) and is stored to the design Project Memory as a Project Memory Element. A Project Memory Element (but not a Design Artifact) is therefore the material Input to an Activity. Hence a Design Artifact may be rightfully used as the material Input for an Activity only after properly stored to the Project Memory. PSI mechanisms assume that a Project Memory is a shared tuple space used for activity run-time coordination based on blackboard principles.

7 DEDP Ontologies: Epistemology and Usage

The descriptive part of DEDP modeling framework has been initially designed as the family of DEDP ontologies and coded in the set of UML class diagrams (further on referenced as DEDP-full). Further formalization and implementation work has been performed in the way aligned with scenarios of ontology usage identified by Uschold and Jasper [UJ99]. DEDP ontologies are used [Go05] for authoring DEDP logs recorded to PSI testbed (neutral authoring), for specifying the designs of DEDP-MAS simulator software (ontology as software specification), and as shared ontologies for agent communication at run time (common access to information). Ontology usage aspects influenced the choice of the formal languages for coding the ontologies. The ontologies were first coded in OWL-lite[9] (further on referenced as DEDP-lite). This language was chosen because it is one of the de-facto ontology specification standards. Second reason for choosing OWL-lite was that its expressive power is similar to that of the internal mental model specification language (MMSL) of MASDK [Go04] which has been used for specifying the design and prototyping of

[9] OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/

PSI prototype – DEDP-MAS. From epistemological viewpoint the transformation of DEDP-full ontologies to DEDP-lite required the changes of UML associations to the constructs with binary relationships with restrictions. This transformation has been performed manually with the help of Protégé 3.0[10] ontology editor.

DEDP-MAS has been implemented to evaluate the modeling framework and to assess the feasibility of building a software tool for DEDP optimization through their productivity assessment. In the performed evaluation experiments [Go05] the simulator is used in two application modes: playback and "freestyle" (predictive) simulation. In playback mode the simulation is used to assess the performance of the DEDPs which have been accomplished in the past. Predictive simulation supports project managers in planning and dynamic re-planning of running design projects in cases of several kinds of events which are out of their direct control: late changes to the design objective, sudden unavailability of the team members, the changes in the workload of the designers according to the influence of the other independent projects, etc.

Simulations performed on the With DEDP records stored to PSI testbed demonstrated that the simulator develops DEDPs very closely to what happened in reality. Observed fluctuations were caused by the changes in the parameters of the availability of the team members in the course of the simulation experiments by "screwing" their available capacities. This fact confirms the adequacy of the developed framework to the industrial requirements in SES.

8 Related Work and Discussion

The projects which pioneered R&D in agent-based engineering design process modeling, support and automation appeared about a decade ago, e.g. [Cu93, DB94, BN95]. Some projects of the "second wave" [PWF99, DJ01] helped to specify the focus of PSI in automating the near-optimal arrangement of DEDPs in terms of their productivity.

DEDP modeling framework in its part of organizational and actor-related knowledge representation bases itself on the frameworks [FG98, UKM98, EKT02]. PSI contribution in this part is the incorporation of roles and actors with its specific subclasses, teams of actors, negotiation context in one coherent ontologies' family and its binding to the engineering design domain through incorporating Design Artifacts and Software Tools ontologies. The main contribution of the family of DEDP ontologies is the model of a dynamic team of designers which is formed through contracting negotiations and performs dynamically orchestrated processes. Hence DEDPs in PSI are understood as socially performed processes in the sense close to [BT04]. For example the notions of a Role or a Policy of PSI Actor ontology are semantically close to that of the normative multi-agent framework.

In the part of process modeling PSI bases its approach on [BV04, EKK04, FB02]. In the family of DEDP ontologies engineering design processes are modeled as tasks composed of sub-tasks and atomic activities. Similarly to [NL99] subtasks and activities may have weak and strong dependencies. However, in PSI the knowledge

[10] Protégé ontology editor and knowledge acquisition system http://protege.stanford.edu/

on these dependencies is local and differs from actor to actor as specified in their partial local plans. Similarly to [FB02] activities have pre-conditions and post-effects. However, DEDP Task-Activity ontology constrains the semantics of pre-conditions and post-effects by making them sub-classes of an event concept. Material inputs and outputs belong semantically and structurally to DEDP Design Artifacts ontology.

Planning and scheduling are known as possibly the oldest research areas in AI. Examples of theoretical frameworks for solving planning tasks are Decision Theoretic Planning (DTP) [Bl99] and Hierarchical Task Networks (HTN) [EHN94]. PSI framework is built upon the conceptual denotation of the planning task shared by the mentioned frameworks. Planning is understood as the process of cascade decomposition of the goal, transformation of the sub-goals to Tasks and committing Actors to Tasks. However PSI framework extends the capabilities of the classical AI approaches to planning by accounting the dynamic character of the process and by the capability to collaborative distributed planning through negotiation mechanisms. The latter feature also distinguishes our descriptive framework from the plan-task ontology of KMI [RM04]. Moreover, the family of DEDP ontologies provides conceptual means for dynamic re-scheduling based on the concepts of Self-Beliefs and Beliefs.

9 Conclusions

The paper presented the descriptive part of DEDP modeling framework developed in the PSI project. The project is aimed to build a software tool assisting in analysis and optimization of DEDPs' productivity through agent-based simulations. The framework incorporates the models of teams and actors, tasks and activities, design artifacts as the major interrelated parts. DEDPs are modeled as weakly defined flows of tasks and atomic activities which may only "become apparent" at run time because of several factors which are beyond the control of the design team members. The processes are self-formed through the mechanisms of collaboration in the dynamic team of actors. These mechanisms are based on several types of negotiations. DEDP productivity is assessed by the Units of Welfare collected by the multi-agent system which models the design team. The models of the framework are formalized in the family of DEDP ontologies. These ontologies are used in the implemented simulator software prototype. Initial evaluation experiments have been performed using PSI testbed [Go05].

References

[Bl99] **Blythe, J**.: Decision-Theoretic Planning. *AI Magazine*, 20 (2), 1999.

[BN95] **Balasubramanian, S. and Norrie, D. H.**: A multi-agent intelligent design system integrating manufacturing and shop-floor control. In: Proc. First Int. Conf. on Multi-Agent Syst., San Francisco, pp. 3-9, 1995

[BT04] **Boella, G.** and **van der Torre, L**.: An Agent Oriented Ontology of Social Reality. In: Varzi, A., Vieu, L. (Eds.) Proc. 3-d Int. Conf on Formal Ontology in Information Systems (FOIS'04), Turin, Nov. 3-6, 2004, pp. 199-209

[BV04] **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*

[Cu93] **Cutkosky, M.R. et al**: PACT: An Experiment in Integrating Concurrent Engineering Systems. *IEEE Computer* 26(1), p. 28-38, 1993

[DB94] **Darr, T. P., Birmingham, W. P.**: An Attribute-Space Representation and Algorithm for Concurrent Engineering. CSE-TR-221-94, University of Michigan, Department of Electrical Engineering and Computer Science, Ann Arbor, Michigan 48109-2122, 1994.

[DJ01] **Danesh, M. R. and Jin, Y.**: An Agent-Based Decision Network for Concurrent Engineering Design. *CERA* 9(1), 37-47, 2001.

[EHN94] **Erol, K., Hendler, J. and Nau, D. S.**: Semantics for Hierarchical Task-Network Planning. Technical report CS-TR-3239, University of Maryland at College Park, 1994.

[EJM04] **Ermolayev, V. et al**: Agent-Based Dynamic Engineering Design Process Modeling Framework. Technical Report. Cadence Design Systems, GmbH, 29 p., 2004.

[EKT02] **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.

[Er04] **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.

[FB02] **Fensel, D. and Bussler, C.**: The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications* 1(2): 113-137, 2002.

[FG98] **Fox, M.C. and Gruninger, M.**: Enterprise Modelling. *AI Magazine* 19(3): 109–121, 1998.

[Go04] **Gorodetsky, V. et al.:** Multi Agent System Development Kit: MAS software tool implementing GAIA Methodology. In: Z. Shi and Q. He (eds.) Int. Conf. on Intelligent Information Processing (IIP2004), Beijing, Springer, pp. 69-78. 2004.

[Go05] **Gorodetsky, V., et al.**: Agent-based framework for simulation and support of Dynamic Engineering Design Processes in PSI. To appear In: Proc. CEEMAS'05, 15-17 September 2005, Budapest, Hungary.

[NL99] **Nagendra Prasad, M. V., and Lesser, V. R.** (1999) Learning situation-specific coordination in cooperative multi-agent systems. *Autonomous Agents and Multi-Agent Systems*. 2(2): 173-207, 1999

[NSB01] **Neal, D., Smith, H. and Butler, D**.: The evolution of business processes from description to data to smart executable code – is this the future of systems integration and collaborative commerce? *Research Services Journal*: March 2001, 39-49

[PWF99] **Parunak, H.V.D. et al**: The RAPPID Project: Symbiosis between Industrial Requirements and MAS Research. *Autonomous Agents and Multi-Agent Systems* 2: 111-140, 1999.

[RM04] **Rajpathak, D. and Motta, E.**: An Ontological Formalization of the Planning Task. In: Varzi, A., Vieu, L. (Eds.) Proc. 3-d Int. Conf. on Formal Ontology in Information Systems (FOIS'04), Turin, Nov. 3-6, 2004.

[UJ99] **Uschold, M. and Jasper, R.**: A Framework for Understanding and Classifying Ontology Applications. In: 12-th Workshop on Knowledge Acquisition, Modeling and Management (KAW'99), Banff, Alberta, CA, 16-21 Oct., 1999