strukt—A Pattern System for Integrating Individual and Organizational Knowledge Work

Ansgar Scherp, Daniel Eißing, and Steffen Staab

University of Koblenz-Landau, Germany
{scherp, eissing, staab@uni-koblenz.de}

Abstract. Expert-driven business process management is an established means for improving efficiency of organizational knowledge work. Implicit procedural knowledge in the organization is made explicit by defining processes. This approach is not applicable to individual knowledge work due to its high complexity and variability. However, without explicitly described processes there is no analysis and efficient communication of best practices of individual knowledge work within the organization. In addition, the activities of the individual knowledge work cannot be synchronized with the activities in the organizational knowledge work. Solution to this problem is the semantic integration of individual knowledge work and organizational knowledge work by means of the pattern-based core ontology *strukt*. The ontology allows for defining and managing the dynamic tasks of individual knowledge work in a formal way and to synchronize them with organizational business processes. Using the strukt ontology, we have implemented a prototype application for knowledge workers and have evaluated it at the use case of an architectural office conducting construction projects.

1 Introduction

There is an increasing interest in investigating means for improving quality and efficiency of knowledge work [6]. An established means for improving efficiency of organizational knowledge work is expert-driven business process management [1]. The implicit procedural knowledge found within the organization is made explicit by defining and orchestrating corresponding business processes (cf. [15]). By this, procedural knowledge of the organization is explicitly captured and made accessible for analysis, planning, and optimization. Important supplement to organizational knowledge work is individual knowledge work. It is present in domains in which acquiring and applying new knowledge plays a central role such as research, finance, and design [14]. Due to its complexity and variability, individual knowledge work is typically not amenable to planning. In addition, activities of individual knowledge work that occur only rarely do not justify the effort of business process modeling. Nevertheless, it seems to be worthwhile to consider individual knowledge work from the perspective of business process optimization. Even if the activities of individual knowledge work are not entirely accessible to planning, they are often embedded in organizational business processes defining,

e.g., some constraints on the activities, deadlines, communication partners, and others [20]. Individual knowledge work often contains some sub-activities that actually provide a fixed structure and thus can be explicitly planned, e.g., to obtain approval for some activities in a large construction project [20]. These activities of individual knowledge work need to be synchronized with the organizational knowledge work. However, today's models for business processes and weakly structured workflows do not allow for representing such an integration of the activities.

Solution to this problem is the semantic integration of individual knowledge work and organizational knowledge work based on the pattern-based core ontology *strukt*[1]. The *strukt* ontology allows for modeling weakly structured workflows of individual knowledge work in a formal and precise way. It allows for decomposing the tasks of the individual knowledge work into sub-tasks, which again can be structured along a specific order of execution and dependencies between the tasks. The tasks can be semantically connected with any kinds of documents, information, and tools of a particular domain. In addition, the strukt ontology provides for modeling structured workflows of organizational knowledge work and combing the weakly structured workflows with the structured ones. The ontology is used in the strukt application that allows knowledge workers to collaboratively create, modify, and execute the dynamic tasks of individual knowledge work and to synchronize them with organizational business processes.

The need for integrating individual and organizational knowledge work is motivated by a scenario of an architectural office in Section 2. Based on the scenario, the requirements on the strukt ontology are derived in Section 3. In Section 4, existing models and languages for business process modeling and weakly structured workflows are compared to the requirements. The pattern-based design of the core ontology strukt is described in Section 5. An example application of the ontology design patterns defined in strukt is provided in Section 6. The strukt application for collaboratively executing tasks of individual knowledge work and synchronizing it with business processes is presented in Section 7, before we conclude the paper.

2   Scenario

The scenario is based on a real architectural office. The work in the architectural office is highly knowledge oriented as the acquisition and application of knowledge plays a crucial role in planning and conducting construction projects. One finds some organizational business processes in the architectural office that are repeated with each project. Figure 1 depicts an excerpt of typical steps in the process of planing an apartment construction in Business Process Modeling Notation (BPMN) [17]. Subsequent to the activity Initiate construction project (a) are the activities Prepare building application (b) and File building application (c). The activities are strictly separated from each other and executed in

---

[1] strukt comes from the German word Struktur and means structure in English.

a determined, sequential order. The resource (d) defines the input and output documents of an activity. In the case of the activity Prepare building application these documents are, e.g., the building application form and all required attachments. The activity Prepare building application is associated with the role Construction draftsman (e), whereas the other activities are conducted by roles like Construction manager, Structural engineer, or Planner. Branches are used to represent parallel activities (f) and conditions (g). Besides the processes within the company also the communication with external project partners is explicitly captured (h).
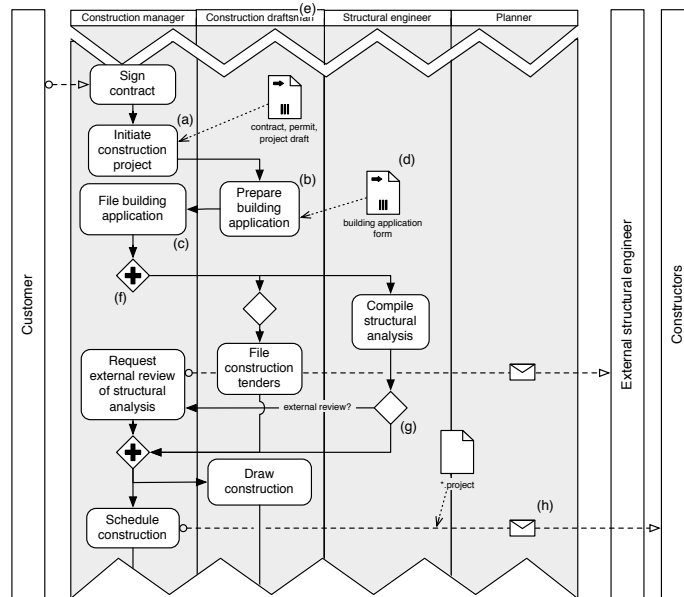

Fig. 1: Example Business Process of the Architectural Office

The organizational knowledge work is already well described on the level of business processes. However, the core area of the architectural office's activities is insufficiently captured. For example, activities such as Prepare building application or Draw construction consist of a large number of sub-activities and are usually collaboratively executed by multiple persons. These activities of individual knowledge work are characterized by high complexity and variability when executing the tasks. As an example, we consider the business process Prepare building application of Figure 1 in more detail: For preparing a building application one has to fill a corresponding application form. This form requires some attachments such as ground plan, site plan, and others that are used to prepare the administrative permit for the construction project. Depending on the type of building construction, however, different attachments are needed. In addition, the construction projects may have specific requirements to be considered like

terrestrial heat, timber construction, accessibility, and others. In some cases a complete structural engineering calculation has to be conducted at application time whereas this is not required in other cases.

3 Requirements to strukt Ontology

We have derived the requirements to the strukt ontology from the scenario in Section 2 as well as from related work in information systems research such as [12, 11, 20, 15]. We briefly discuss each requirement and provide a reference number *REQ-<number>*.

Weakly Structured Workflows (*REQ-1*): Individual knowledge work is characterized by a high complexity and variability [12]. Resources and activities for conducting tasks are often not known a priori (see Section 2). A support for representing weakly structured workflows is needed that can be adapted during execution time without violating the consistency of other running processes.

Support for Structured Workflows (*REQ-2*): Despite the high flexibility of indidivual knowledge work, there are also some organizational requirements and framework directives that need to be strictly followed (see scenario in Section 2). Thus, support is needed to represent structured workflows in the sense of traditional business process management [11].

Integrating Weakly Structured and Structured Workflows (*REQ-3*): Within an organization there is typically a need to represent both weakly structured workflows and structured workflows (see Section 2). Today's models and systems, however, lack in formally integrating weakly structured workflows and structured workflows and thus cannot benefit from this integration. In order to leverage the strength of both weakly structured and structured workflows, an appropriate model must be able to formally integrate and synchronize them into a common workflow.

Workflow Models and Instances (*REQ-4*): Distinguishing workflow models and workflow instances is a common feature of traditional business process models [20]. In individual knowledge work, however, such a distinction is often not made as the individual knowledge work is high in complexity and variability. However, also from the execution of weakly structured workflows one can learn some generic procedural knowledge. Thus, also for weakly structured workflows the distinction between instance and model should be made. In addition, it shall be possible to modify a workflow instance without affecting its workflow model or other workflow instances. In addition, it shall be possible to derive workflow models from executed workflow instances.

Descriptive Workflow Information (*REQ-5*): Structured workflows and weakly structured workflows are characterized by the resources involved. A core ontology for integrating individual and organizational knowledge work should therefore support describing the necessary information for the workflow execution, like resources used, processed, or created (which is a central aspect in particular for individual knowledge work [15]), the tools applied, the status of the workflow execution, as well as scheduling information (cf. Section 2).

4 Comparing Models for Knowledge Work

We analyze and evaluate existing models for structured workflows and weakly structured workflows with respect to the requirements introduced in Section 3. The traditional business process models like BPMN [17] and extended Event-driven Process Chain (EPC) [18] are available as semantic models in form of the sBPMN [13] and sEPC [13] ontologies. However, they still lack support for representing weakly structured workflows and thus are less applicable to our problem. Also OWL-S [22] shares these characteristics of traditional business process models. Ad-hoc and weakly structured models like the Process Meta-Model (PMM) [2] and the Task-Concept-Ontology (TCO) [19] do not require a strictly determined process flow like the traditional business process models and may be automatically extracted from natural language descriptions [10]. Such models are suitable to represent individual knowledge work. However, the lack of formal precision and missing integration with traditional business processes hinder their reuse.

The DOLCE+DnS Plan Ontology (DDPO) [8] provides a rich axiomatization and formal precision. It obtains its high level of formal precision from the foundational ontology DOLCE [3] and specializes the ontology design pattern Descriptions and Situations (DnS). The central concepts defined in the DDPO are Plan, Goal, Task, and PlanExecution [8]. A Plan is a description of at least one Task and one agentive role participating in the task. In addition, a Plan has at least one Goal as a part. A Goal is a desire that shall be achieved. Tasks are activities within plans. They are used to organize the order of courses. Finally, PlanExecutions are actual executions of a plan, i.e., they are real-world situations that satisfy a Plan. It is in principle possible to represent both traditional workflows as well as weakly structured workflows using the DDPO. However, DDPO does not distinguish structured and weakly structured workflows (*REQ-3*) and does not support descriptive workflow information (*REQ-5*). *REQ-4* is present in DDPO but not explicitly specified. Nevertheless, due to its high formality and using the foundational ontology DUL as basis, the DDPO is well suited for extensions and serves as basis for our work.

In conclusion, one can say that none of the existing models fulfill all requirements stated to strukt. Traditional business process models miss representing weakly structured workflows of individual knowledge work. On contrary, weakly structured workflows are in principle enabled to represent the activities of individual knowledge work. However, they lack the formal precision required and do not allow for an integration with traditional business process models. The DDPO model differs from the other models insofar as it in principle allows for modeling both organizational business processes and activities of individual knowledge work. In addition, it enables integration with other systems due to its formal nature. Thus, it is used as basis in our work and will be adapted and extended towards the requirements stated in Section 3.

5 Pattern-based Core Ontology strukt

The foundational ontology DOLCE+DnS Ultralight (DUL) [3] serves as basis for the core ontology strukt. Foundational ontologies like DUL provide a highly axiomatized representation of the very basic and general concepts and relations that make up the world [16]. As such, foundational ontologies are applicable to a wide variety of different fields. Foundational ontologies like DUL follow a pattern-oriented design. Ontology design patterns [9] are similar to design patterns in software engineering [7]. Adapted from software engineering, an ontology design pattern provides (i) a description of a specific, recurring modeling problem that appears in a specific modeling context and (ii) presents a proven, generic solution to it [4, 7]. The solution consists of a description of the required concepts, their relationships and responsibilities, and the possible collaboration between these concepts [4]. An ontology design pattern is independent of a concrete application domain [7] and can be used in a variety of different application contexts.

In the following, we briefly introduce the patterns of DUL that are of particular interest in this work:[2] The Descriptions and Situations Pattern provides a formal specification of context [16]. The Description concept formalizes the description of a context by using roles, parameters, and other concepts. The Situation represents an observable excerpt of the real world that satisfies the Description. By using the Descriptions and Situations Pattern, different views onto the same entities can be formally described. The patterns of the core ontology strukt are based on the Descriptions and Situations Pattern. This means that they reuse or specialize concepts or relations defined in the pattern. The foundational ontology DOLCE+DnS Ultralight provides a specialization of the DDPO [8] (see Section 4) for planning activities, called the Workflow Pattern. Central entity of the Workflow Pattern is the Workflow concept that formalizes the planning of processes. The Workflow concept is specialized from DDPO's Plan, which itself is derived from Description of the Descriptions and Situations Pattern. The WorkflowExecution concept represents the concrete execution of a workflow instance. It is derived from DDPO's PlanExecution, which is a specialization of Situation. The Task Execution Pattern formalizes the processing of tasks in activities. The Role Task Pattern enables association of roles to tasks. The Part-of Pattern represents the (de-)composition of entities into wholes and parts [21]. The Sequence Pattern describes the order of entities through the relations precedes, follows, directlyPrecedes, and directlyFollows.

A core ontology refines a foundational ontology towards a particular field by adding detailed concepts and relations [16]. However, core ontologies are still applicable in a large variety of different domains. The core ontology strukt reuses and specializes different ontology design patterns that DUL offers. Central patterns of the core ontology strukt are the Weakly Structured Workflow Pattern (*REQ-1*), the Structured Workflow Pattern in combination with the Transition Pattern (*REQ-2*), the Workflow Integration Pattern to integrate weakly structured workflows and structured workflows (*REQ-3*), and the Workflow Model

---

[2] For a detailed description we refer to `http://ontologydesignpatterns.org/`

Pattern for differentiating workflow models and workflow instances (*REQ-4*). Weakly structured workflows and structured workflows can be further described by applying strukt's Condition Pattern, Resource Pattern, Status Pattern, and Scheduling Pattern (*REQ-5*). Each pattern of the core ontology strukt solves a specific modeling problem that distinguishes it from the other patterns. However, strukt is not just a collection of some otherwise independent ontology design patterns. Rather, the set of ontology design patterns strukt defines relate to each other and are designed to be applied together. Such a set of related patterns is called a pattern system [4]. The core ontology strukt can be applied in various domains that need to represent knowledge work and workflows, respectively. Finally, strukt can be extended by domain ontologies such as an architectural ontology or financial administration ontology. In the following, we describe the patterns of the strukt ontology.

5.1 Weakly Structured Workflow Pattern

The Weakly Structured Workflow Pattern depicted in Figure 2 refines the generic Workflow Pattern of DUL. The concept WeaklyStructuredWorkflow specializes the Workflow concept of DUL's Workflow Pattern. Using the defines relation, different Roles and Tasks are defined. Roles abstract from the characteristics, skills, or procedures relevant for the execution of a specific task and allows for differentiating Agents and Objects participating in activities (see Role Task Pattern of DUL). The classifies relation determines the Role of an Object in the context of a specific workflow. The concept Agent is a specialization of the Object concept and describes the entity acting such as a person. Objects and Agents are defined as participants of an Action by using the hasParticipant relation. Tasks are used to sequence activities [8]. They structure a workflow into different sub-tasks relevant for the workflow execution and can be hierarchically ordered (see Task Execution Pattern of DUL). Tasks are associated to Actions using the relation isExecutedIn. Action is a specialization of DUL's Event and describes the actual processing of a task. Tasks can be ordered using the precedes relation. The order of tasks may be underspecified, i.e., the actual sequence of processing may only be determined on a short-term basis and day-to-day requirements when executing the workflow. Thus, a strict order of processing the tasks is not enforced and the order may even change during execution time.

In knowledge-intensive activities, it may not be possible to define *a priori* all details of a complex task. Thus, the Weakly Structured Workflow Pattern allows for defining additional (sub-)tasks during the execution of the workflow using the hasPart relation. A Task is associated with a Role using the isTaskOf relation (see Part-of Pattern of DUL). Also Actions can be decomposed using the relation hasPart. Typically, the decomposition of an Action is bound with the decomposition of the corresponding Task.

The goal that is to be reached by executing a workflow is represented using the Goal concept and associated to WeaklyStructuredWorkflow using the hasComponent relation. It can be further decomposed into sub-goals using the hasPart

Fig. 2: Weakly Structured Workflow Pattern

relation. Goals are explicitly associated to corresponding sub-tasks using the relatesTo relation. The Goal concept is central to the weakly structured workflow pattern and is used by the Workflow Integration Pattern described in Section 5.4 to link the Weakly Structured Workflow Pattern with the Structured Workflow Pattern.

5.2 Structured Workflow Pattern and Transition Pattern

The Structured Workflow Pattern provides a formal specification of traditional business processes (see Section 4). It is applied in combination with the Transition Pattern that defines the transitions between processes, i.e., the Events. Thus, the Structured Workflow Pattern is an abstraction from the concepts of traditional business process models.

Figure 3 depicts the Structured Workflow Pattern. It specifies the concepts StructuredWorkflow and StructuredWorkflowExecution as specialization of DUL's Workflow and WorkflowExecution concepts. This eases the integration with the Weakly Structured Workflow Pattern that specializes the same concepts and thereby supports the integration of individual and organizational knowledge work. The distinction between StructuredWorkflow and StructuredWorkflowExecution reflects the two phases of traditional business process management, namely the definition phase and execution phase [23]. In the definition phase, existing business processes are captured and orchestrated into a (semi-)formal business process model. In the execution phase, the previously created process model is implemented.

Using the defines relation, the StructuredWorkflow specifies the Roles, Tasks, EventTypes, and TransitionTypes of the workflow as in the definition phase. Roles determine the roles played by Objects such as Agents participating in processes of the workflow. The roles are associated with some concrete Tasks using the isTaskOf relation. The TransitionType is part of the Transition Pattern and allows for formally defining the transition between two concepts, which classify
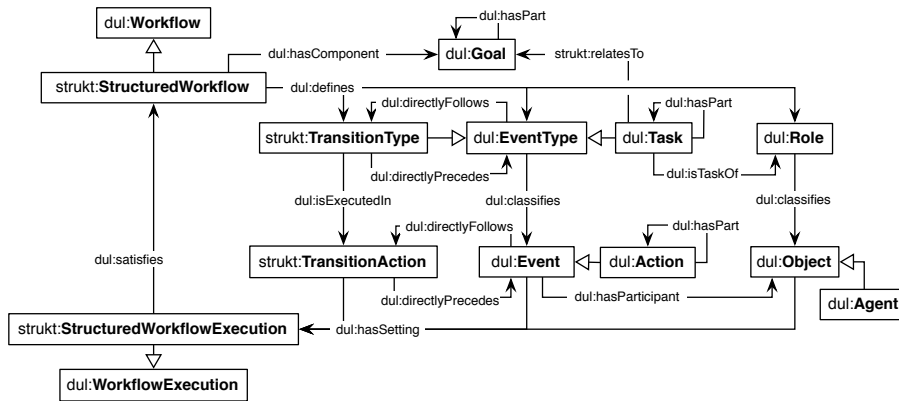
Fig. 3: Structured Workflow Pattern

processes represented by DUL's Event concept. The concepts TransitionAction, Event, and Object constitute the entities of the workflow execution phase. Like the WeaklyStructuredWorkflow concept, also the StructuredWorkflow concept defines a Goal concept, which captures the goal of the workflow.

The transitions between business processes are defined using the Transition Pattern. It provides the four basic transition types [18, 17, 5] *sequence*, *condition*, *fork*, and *join*, defined as specializations of the generic Transition Pattern. The Sequence Transition Pattern specifies a strict sequence of process execution as depicted in Figure 4(a). The corresponding operator in BPMN is shown in Figure 4(b). The Sequence Transition Pattern defines a SequenceTransitionType as specialization of the generic TransitionType. It determines a strict sequential order of execution of two EventTypes. Thus, the SequenceTransitionAction connects exactly two concrete business processes represented as Events. The Condition-based Transition Pattern models process executions that are bound to some process conditions. The Fork Transition Pattern is used to model fork/join transitions.
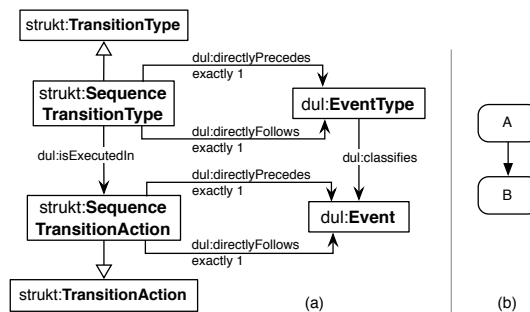

Fig. 4: Sequence Transition Pattern

5.3 Condition Pattern, Resource Pattern, Status Pattern, and Scheduling Pattern

The workflows specified using the Weakly Structured Workflow Pattern and Structured Workflow Pattern can be further described with information about the conditions, resources, status, and scheduling of activities. Information about conditions for executing an activity are added by combining the Weakly Structured Workflow Pattern or Structured Workflow Pattern with the Condition Pattern. The Condition Pattern allows for defining some preconditions and postconditions such as that a document needs to be signed. Using the Resource Pattern, one can define if an activity produces a resource (create), uses a resource (without exactly knowing if the resource is modified or not), views a resource (without modifying it, i.e., read), edits a resource (update), consumes a resource (delete), or locks a resource. The status of activities and processes can be set to active, inactive, or finished using the Status Pattern. The pattern can be extended to domain specific requirements such as initiated, suspended, and failed. Activities may have to be executed at a specific time and/or place. This can be represented using the Scheduling Pattern.

5.4 Workflow Integration Pattern

The integration of individual knowledge work and organizational knowledge work is conducted using the Workflow Integration Pattern specialized from DUL's Workflow Pattern and is depicted in Figure 5. The alignment of the concepts defined in the Weakly Structured Workflow Pattern and the Structured Workflow Pattern is supported by using the Workflow Pattern of DUL as common modeling basis. As described in Section 5.1 and Section 5.2, it is possible to associate a Goal to each Task using the relatesTo relation. The Goal concept is connected to the workflow via the hasComponent relation. Using the Goal concept, a formal mapping of weakly structured workflows and structured workflows can be conducted. It is based on the assumption that if some individual knowledge work is carried out in the context of an organizational business process or vice versa, they share a common Goal. Finally, the association between concrete activities carried out in the individual knowledge work and organizational knowledge work is established through the relatesTo relation that connects the Goals with Tasks in the Weakly Structured Workflow Pattern and the Structured Workflow Pattern.

5.5 Workflow Model Pattern

The Workflow Model Pattern allows for explicitly distinguishing workflow models and workflow instances for both, weakly structured workflows and structured workflows. To create a workflow model, the Workflow Model Pattern is able to represent on a *generic*, i.e., conceptual level, the flow of Tasks, their dependencies, and the resources required. In contrast to the traditional business process modeling (see Section 4), however, the workflow instances created from a workflow model do not need to be strictly in accordance with the model. This is in
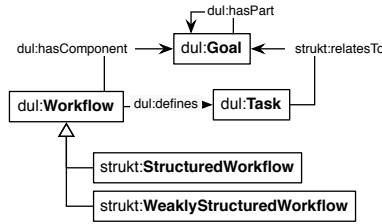
Fig. 5: Workflow Integration Pattern

particular important for weakly structured workflows that can be adapted to the requirements of a concrete execution situation.

Figure 6 depicts the Workflow Model Pattern. It consists of two parts, one for the WeaklyStructuredWorkflowModel and one for the StructuredWorkflowModel. In the case of the StructuredWorkflowModel, subclasses of Role, Task, and TransitionType are defined as valid components of the workflow model definition. For weakly structured workflow models, only Roles and Tasks can be defined.
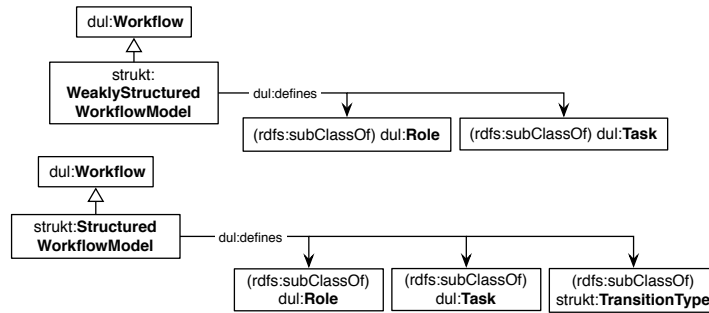
Fig. 6: Workflow Model Pattern

6 Example Application of the strukt Core Ontology

The application of the strukt core ontology is shown at the example of an apartment construction by the architectural office introduced in Section 2. Figure 7 (bottom part) depicts the application of the Weakly Structured Workflow Pattern wsw-prepare-building-application-1 for preparing a building application. It defines the Tasks t-compute-statics-1 and t-create-groundplan-1. The relation isExecutedIn classifies the individuals a-compute-statics-1 and a-create-groundplan-1 as Actions, executing the tasks. The isTaskOf relation specifies that the task t-compute-statics-1 has to be conducted by an agent playing the role of a StructuralEngineer r-structural-engineer-1, here the NaturalPerson tmueller-1. The Nat-
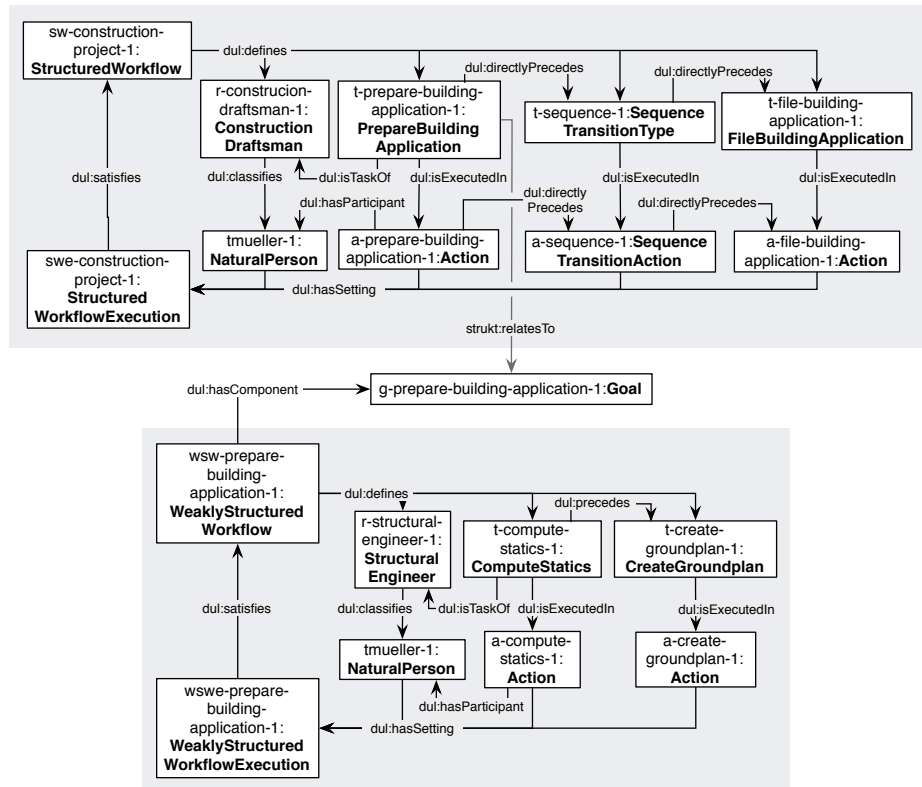
Fig. 7: Example integration of weakly structured and structured workflow

uralPerson tmueller-1 is specified as participant of the Action a-compute-statics-1. The participant of the Action a-create-groundplan-1 is not specified.

The weakly structured workflow belongs to the organizational business process depicted in Figure 7 (top part) using the Structured Workflow Pattern. It models an excerpt of the business process shown in Figure 1 of the scenario in Section 2. The StructuredWorkflow sw-residential-object-1 defines the Tasks t-prepare-building-application-1 and t-file-building-application-1, the SequenceTransitionType tt-sequence-1, and the role r-draftsman-1. The tasks t-prepare-building-application-1 and t-submit-application are connected in a sequence using the relations directlyPrecedes and directlyFollows of t-sequence-1. The Role r-draftsman-1 is connected using the isTaskOf relation with the Task t-prepare-building-application-1. In the context of this workflow, the NaturalPerson tmueller-1 acts as r-draftsman-1. The Actions a-prepare-building-application-1, a-sequence-1, and a-submit-application-1 constitute the execution of the Tasks and SequenceTransitionType, respectively. The integration of the weakly structured workflow wsw-prepare-building-application-1 and structured workflow sw-construction-project-1 is conducted by defining g-prepare-building-application-1 as Goal of the t-prepare-building-applica-
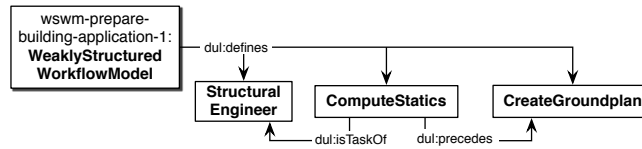
Fig. 8: Application of the Workflow Model Pattern

tion-1 task. The Goal g-prepare-building-application-1 is then connected with the Weakly Structured Workflow wsw-prepare-building-application-1 using the has-Component relation. As described above, the wsw-prepare-building-application-1 captures the individual activities, concrete sub-tasks, and roles involved in actually writing the building application.

An instance of a workflow such as the example of the weakly structured workflow wsw-prepare-building-application-1 in Figure 7 (bottom part) can be abstracted to a workflow model using the Workflow Model Pattern. As shown in Figure 8, the abstraction from a workflow instance to a model is basically the upper part of the Descriptions and Situations Pattern of DUL. In our example, the WeaklyStructuredWorkflowModel wswm-prepare-building-application-1 consists of the domain-specific concepts of the role StructuralEngineer, the two tasks ComputeStatics and CreateGroundplan, and the relations.

As shown in Figure 7, using the patterns of struct allows for modeling and integrating structured workflows and weakly structured workflows. Using the ontology design pattern Descriptions and Situations as design principle for representing workflows in strukt allows for modifying workflow instances without affecting the original workflow model. This is achieved by contextualizing the workflow instances using the individuals sw-construction-project-1 and wsw-prepare-building-application-1. Other instances of the same WeaklyStructuredWorkflow like a wsw-prepare-building-application-2 can have different roles and tasks defined for the actual execution and the tasks can be executed in different order.

7 Prototype Application

The prototype application supports individual and organizational knowledge work and their combination. It instantiates the pattern of the strukt ontology. A domain-specific construction ontology aligned to DUL is used to describe the roles such as manager, draftsman, and engineer. The user interface for the individual knowledge worker is depicted in Figure 9. It consists of a *task space* for managing the weakly structured workflows with their tasks and sub-tasks. The task space allows for receiving details of a task, create new tasks, modify tasks, save a workflow instance as workflow model, instantiating a workflow model, and deleting tasks and workflows, respectively.

The left hand side of the screenshot depicted in Figure 9 shows example weakly structured workflows from the architecture scenario presented in Section 2. The tasks and subtasks of a weakly structured workflow can be shown by
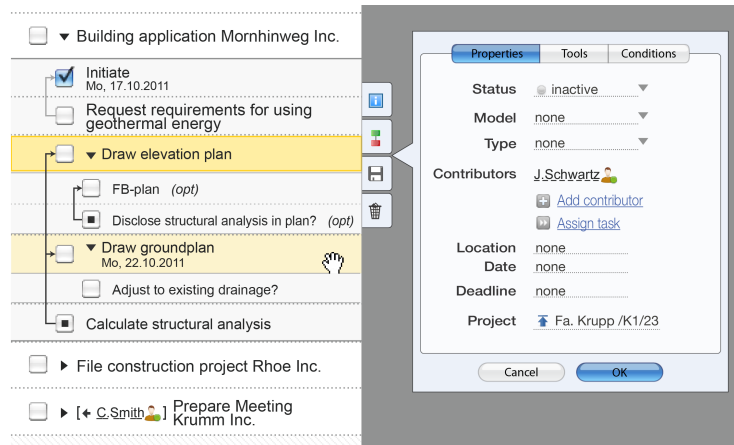
Fig. 9: Task Space for the Individual Knowledge Worker

clicking on the small triangle symbol next to the task like the Building application Mornhinweg Inc example. Important details of a task are shown on the right hand side of the screenshot such as deadlines, appointments, and others. Tasks can be marked as finished by clicking on the checkbox on the left to the task name. When there is a lock symbol in the checkbox (indicated as small box), the task cannot be accomplished due to unfulfilled dependencies (indicated by the arrows). For example, the task Calculate structural analysis cannot be processed as the tasks Draw elevation plan and Draw ground plan are not completed. Optional tasks are indicated with the keyword *(opt)*. The order of tasks in a weakly structured workflow can be changed by the knowledge worker using simple drag and drop interaction. The right hand side of the screenshot in Figure 9 provides details of a task such as its status and the responsible agent. Additional agents can be added as well as the responsibility of tasks can be forwarded. Thus, the strukt prototype enables a collaborative execution of a weakly structured workflow by multiple knowledge workers. Further details can be investigated using the tab Tools showing the tools used to process a task and the tab Conditions showing detailed information about the conditions associated with the task, e.g., when a specific role needs to sign a specific document.

In order to abstract a workflow model from a workflow instance, the strukt application provides the *workflow transformation menu* depicted in Figure 10. It allows for defining the components of the workflow model. To this end, all components of the workflow instance to be transformed are depicted in a table. Each row of the table represents a task of the weakly structured workflow. Subtasks are indicated by indentions. The columns Task, Conditions, Optional, Role conditions, Documents, and Tools show the details of the tasks relevant for creating a workflow model. Tasks can be removed from the workflow at this point from the transformation process. In addition, the order of tasks can be changed by drag and drop interaction and new tasks can be added.
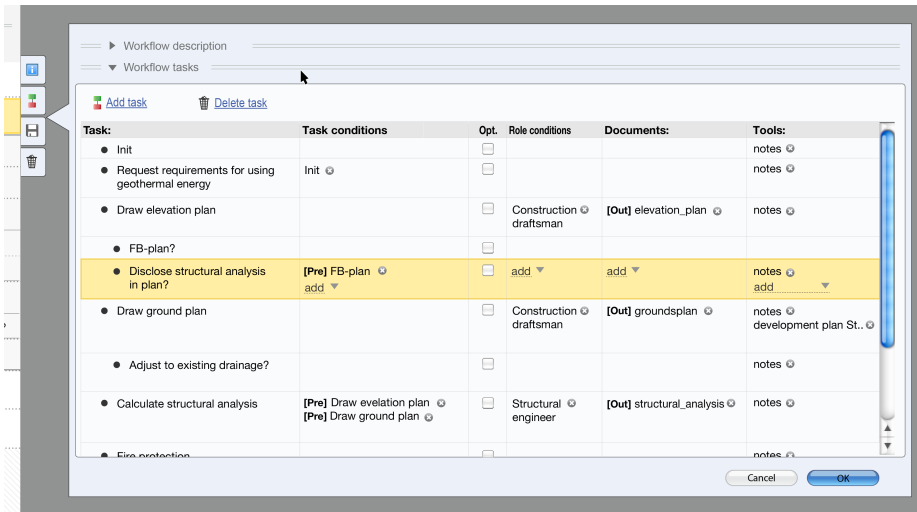
Workflow description

Workflow tasks

Add task   Delete task

Task:	Task conditions	Opt.	Role conditions	Documents:	Tools:
Init		☐			notes
Request requirements for using geothermal energy	Init	☐			notes
Draw elevation plan		☐	Construction draftsman	[Out] elevation_plan	notes
FB-plan?		☐			
Disclose structural analysis in plan?	[Pre] FB-plan add ▼	☐	add ▼	add ▼	notes add ▼
Draw ground plan		☐	Construction draftsman	[Out] groundsplan	notes development plan St..
Adjust to existing drainage?		☐			notes
Calculate structural analysis	[Pre] Draw evelation plan [Pre] Draw ground plan	☐	Structural engineer	[Out] structural_analysis	notes
Fire protection		☐			notes

Cancel   OK

Fig. 10: Transformation Menu for Creating a Workflow Model from an Instance

We have implemented a simple workflow management system in our strukt prototype. It provides a test environment for synchronizing the activities in the weakly structured workflows and some pre-defined business processes of the architectural office. A user interface is not provided as it is assumed that strukt is integrated in an existing business process engine with its own interface.

8   Conclusions

We have presented an approach for integrating individual knowledge work and organizational knowledge work by means of the pattern-based core ontology *strukt*. The core ontology strukt defines several ontology design patterns for capturing weakly structured workflows of individual knowledge work and structured workflows in the context of organizational knowledge work. A formal alignment and synchronization of the activities in individual knowledge work and organizational knowledge work is conducted by basing on the DOLCE+DnS Plan Ontology [8]. Concrete instances of weakly structured workflows can be transformed into generic workflow models, enabling reuse of procedural knowledge. On basis of the strukt ontology, we have developed a prototypical software system for the collaborative planning and execution of weakly structured workflows and applied it to the use case of an architectural office. The strukt prototype connects with a simple workflow management system to synchronize the flexible, individual knowledge work with the strict execution of business processes. This work has been co-funded by the EU in FP7 in the ROBUST project (257859).

References

1. J. Becker, B. Weiss, and A. Winkelman. Developing a Business Process Modeling Language for the Banking Sector - A Design Science Approach. In *Americas Conference on Information Systems*, pages 1–12, 2009.
2. J. Bolinger, G. Horvath, J. Ramanathan, and R. Ramnath. Collaborative workflow assistant for organizational effectiveness. In *Applied Computing*. ACM, 2009.
3. S. Borgo and C. Masolo. *Handbook on Ontologies*, chapter Foundational choices in DOLCE. Springer, 2nd edition, 2009.
4. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley, 1996.
5. S. Carlsen. Action port model: A mixed paradigm conceptual workflow modeling language. In *Cooperative Information Systems*, pages 300–309. IEEE, 1998.
6. P. Drucker. Knowledge-Worker Productivity: The Biggest Challenge. *California Management Review*, 41(2):79–94, 1999.
7. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, July 2004.
8. A. Gangemi, S. Borgo, C. Catenacci, and J. Lehmann. Task Taxonomies for Knowledge Content. *METOKIS Deliverable*, D7:20–42, 2004. `http://www.loa-cnr.it/Papers/D07_v21a.pdf`.
9. A. Gangemi and V. Presutti. *Handbook on Ontologies*, chapter Ontology Design Patterns. Springer, 2009.
10. P. T. Groth and Y. Gil. A scientific workflow construction command line. In *Intelligent user interfaces*. ACM, 2009.
11. M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. Harper Paperbacks, 2003.
12. W. Hart-Davidson, C. Spinuzzi, and M. Zachry. Capturing & Visualizing Knowledge Work: Results & Implications of a Pilot Study of Proposal Writing Activity. In *Design of Communication*, pages 113–119. ACM, 2007.
13. M. Hepp, R. Belecheanu, J. Domingue, A. FilipowskaG, M. Kaczmarek, T. Kaczmarek, J. Nitzsche, B. Norton, C. Pedrinaci, D. Roman, et al. Business Process Modelling Ontology and Mapping to WSMO. Technical report, SUPER Project IST-026850, 2006.
14. A. Kidd. The marks are on the knowledge worker. In *Human Factors in Computing Systems*, pages 186–191. ACM, 1994.
15. I. Nonaka. The Knowledge-Creating Company. *Harvard Business Review*, 69(6):96–104, 1991.
16. D. Oberle. *Semantic Management of Middleware*. Springer, 2006.
17. OMG. Business process model and notation (BPMN), version 2.0 beta 2, 2010. `http://www.omg.org/cgi-bin/doc?dtc/10-06-04.pdf`.
18. A. Scheer. *Aris-Business Process Frameworks*. Springer, 1998.
19. S. Schwarz. Task-Konzepte: Struktur und Semantik für Workflows. In *Professionelles Wissesmanagement; Luzern, Switzerland*. GI e.V., 2003.
20. S. Schwarz, A. Abecker, H. Maus, and M. Sintek. Anforderungen an die Workflow-Unterstützung für Wissensintensive Geschäftsprozesse. In *Professionelles Wissensmanagement; Baden-Baden, Germany*, 2001.
21. A. Varzi. Parts, Wholes, and Part-Whole Relations: The Prospects of Mereotopology. *Data & Knowledge Engineering*, 20(3):259–286, 1996.
22. W3C. OWL-S, 2004. `http://www.w3.org/Submission/OWL-S/`.
23. WFMC. Terminology & Glossary. Technical Report WFMC-TC-1011, Version 3.0, 1999. `http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf`.