# Knowledge Acquisition from EPC Models for Extraction of Process Patterns in Engineering Domains

Andreas Bögl, Maximilian Kobler, Michael Schrefl

Johannes Kepler University Linz, Austria
Altenberger Strasse 69
A-4040 Linz
boegl@dke.uni-linz.ac.at
maximilian.kobler@jku.at
schrefl@dke.uni-linz.ac.at

**Abstract:** This paper presents an approach for the automated extraction of process patterns from Event-driven Process Chain (EPC) models in engineering domains. The manually extraction of process patterns (semantically described reference building blocks) is a labor-intensive, tedious and cumbersome task. The introduced approach comprises the three stages knowledge acquisition, process pattern extraction and generic pattern construction that are conducted automated. The presented approach is characterized by exploiting the implicit semantics of natural language terms associated with EPC model elements. Semantic patterns are employed to analyze the lexical structure of these terms and to relate them to instances of a reference ontology. Thus semantically annotated EPC model elements are input for subsequent process pattern extraction. The semantic annotation allows during pattern extraction to identify process goals of EPC functions and to organize goals into hierarchies. During generic process pattern construction, common goals of different process patterns give rise to construct generic process patterns.

## 1 Introduction

Process modeling constitutes a sophisticated and cognitive task. Reference models can serve as templates for new models or as guidelines for comparing and validating specific process models. [Sc98, 237] proposes the identification of patterns (structural analogies) in existing models as a possible approach for the construction of reference models. These patterns (in the following referred to as *process patterns*) are semantically described reference building blocks including structural, behavioral and semantic information.

The automated extraction of process patterns from existing process models generates several advantages. Using a large set of specific models for derivation of reference models offers a detailed insight into the common and best practices of a domain [JS07]. The frequency of occurrence of process patterns enables an objective measure to evaluate candidates for common or best practice solutions. The dependencies between patterns can provide information on larger structures (reference models [SBK07]) or process variants (configurable and generic adaptation of reference models [BDK07]).

This paper introduces a novel approach that extracts process patterns automatically from engineering processes described by EPC (Event-driven Process Chain) models. The overall proceeding for process pattern extraction comprises the three stages (1) *Knowledge Acquisition* (2) *Extraction of Process Patterns and* (3) *Construction of generic Process Patterns.* The purpose of the knowledge acquisition step is the elicitation of process knowledge captured in process models describing engineering processes and to populate a process knowledge base (reference ontology) automatically. The reference ontology provides the grounding for the extraction of semantically described process patterns. Stage 3 addresses the construction of generic process patterns. They represent generalized process solutions retrieved from extracted patterns.

The automated knowledge acquisition from EPC models faces the problem that an essential part of the semantics is bound to natural language. Syntactical similarities can be misleading; even though the difference is slight, the semantic can vary substantially. For example, "*Define Software Requirements with Customer*" has a different meaning than *"Define Software Requirements for Customer".* The former expression means that software requirements are defined in a cooperative manner with a customer; the latter one indicates that software requirements are the output of the activity for the target group customer. The semantic annotation of EPC models provides an adequate solution to tackle this problem but cause the problem to determine the semantics of used lexical terms in natural language expressions and to populate them in a knowledge base. This can be achieved manually, but a human-driven semantic annotation of process models remains a tedious, cumbersome task that may result in a knowledge acquisition bottleneck. To cope with this problem, we adopt the idea of semantic patterns that is inspired by [RS98]. They employ semantic patterns to extract a use case model from ambiguous textual use case descriptions. Semantic patterns allow the specification of lexical structures for EPC functions/events and its associated semantics that bridges the gap between informal and formal representation. Further, lexical structures assist in tackling the problem of expressing same or similar knowledge in a different syntactical manner.

The paper is structured as follows. The upcoming section presents knowledge acquisition and addresses the semantic annotation of EPC functions/events. Section 3 discusses the extraction of process patterns and highlights the relationship between semantically annotated EPC functions/events and extracted process patterns. Section 4 summarizes the construction of generic process patterns. Section 5 deals with related work. The paper concludes with a summary of the experiences with the tool support and the practical experiences.

## 2 Knowledge Acquisition from EPC Models

Knowledge acquisition aims at extracting the semantic knowledge from natural language expressions used to describe EPC functions/events and to capture this knowledge in a shared reference ontology. The ontology is populated automatically during the semantic annotation process which results in semantically annotated EPC functions/events.

## 2.1 Reference Ontology

The introduced reference ontology represents the fundamentals for the comparison of EPC models. It provides concepts and relations that capture the semantics of lexical terms used for a natural language description of EPC functions/events. Ontology instances are identified by their names (e.g. software). The textual counterpart not necessarily equals the concept instance name. This can result from implicit semantics imposed by process modelers, especially when using different multiword expressions for the same concept instance name [BPZ05]. This implies the separation of lexical knowledge from process knowledge in the reference ontology; textual counterparts of process ontology instances are mapped to lexical concept instances (see Figure 1).

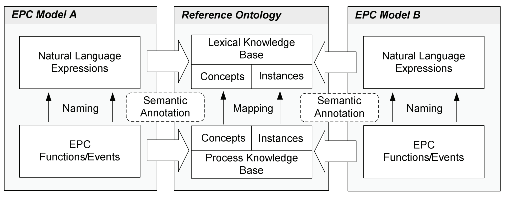

Figure 1: Coherence between Reference Ontology and EPC Functions/Events

The rationale behind the lexical knowledge base is to provide a lightweight, controlled vocabulary. The design is targeted to data semantic requirements in process modeling. It introduces concepts such as *isAcronymOf* that explicitly captures domain specific acronyms (e.g. SW for Software) or *Qualifier* that explicitly captures the semantics of mathematical operations used in EPC functions/events (e.g. Define Test Plan *for each* Software Module).

Publicly available resources such as WordNet [WN06] may provide commonsense vocabulary, but cannot be considered as fully suitable for capturing domain specific vocabulary. In general, such resources are open world dictionaries, comprising several hundred thousand open world entities and semantic relationships. It is an unreasonable demand for a process designer to maintain a controlled domain vocabulary with unambiguous word meanings within an open world lexicon. A domain specific controlled vocabulary within an engineering domain usually comprises several hundred entities that can be maintained easier.

The concepts and relations of process knowledge result from the analysis of natural language terms describing EPC functions/events and its associated meanings. Figure 2 depicts the process ontology concepts for capturing the semantics of EPC functions/events. The Top level concept EPC Entity classifies an lexical term either into a Task(T), or a Process Object(PO) or a State(S) concept. A Process Object represents a real or an abstract thing being of interest within a (business process) domain. According to [Ro95, 177 et seq.], the concept for describing a Process Object has the semantic relations isPartOf (e.g. Project Handbook isPartOf Development Project), isSublcassOf (e.g. Software Project isSubClassOf Project) and migratesTo (e.g. Software Requirements *migratesTo* Software Architecture). A Task can be performed manually by a human or electronically by a service (e.g. Web Service) for achieving a desired (business) objective. It can be specified at different levels of abstraction, refinements or specializations that are expressed by the semantic relationship hasSubTask.

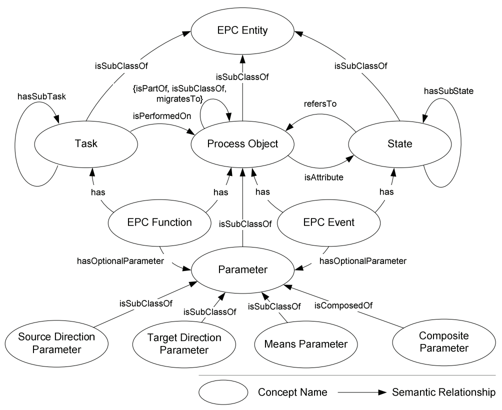

Figure 2: Process Ontology Concepts for capturing the Semantics of EPC Functions/Events

A State always refers to a Process Object indicating the state that results from performing a Task on a Process Object. Parameters are process objects that are relevant for a task execution or a state description. The process ontology comprises the three parameter types Source Direction Parameter, Target Direction Parameter and Means Parameter. A Source Direction Parameter defines a source process object, e.g. *"Derive Quality Goal* from *Specification Document"* indicates *"Specification Document"* as a Source Direction Parameter. A Target Direction Parameter denotes a recipient process object within a function execution, e.g. the function *"Rework Specification* for *Project Plan"* specifies *"Project Plan"* as a Target Direction Parameter. A Means Parameter semantically describes a process object as an input requirement for task execution. For example, *"Rework Specification* with *Software Goals"* indicates *"Software Goals"* as a Means Parameter. Additionally, a function/event may contain a composition of parameters, expressed by the concept CompositeParameter. For example, *"Rework Specification with Software Goals for Project Handbook"* is a composition of the two parameters *"Software Goals"* (Means Parameter) and *"Project Handbook"* (Target Direction Parameter).

## 2.2 Semantic Annotation Process

The semantic annotation of EPC models entails the population of reference ontology instances and establishes a linkage of EPC functions/events to these instances as illustrated in Figure 3. Knowledge base population requires the extraction of the semantics of used lexical terms. The semantics relies on propagated guidelines for naming EPC functions/events. Traditional modeling conventions [Sc98, 189] suggest the usage of a verb to express a task followed by a noun that refers to a process object for the naming of EPC functions (e.g. Define:[Task] Architecture:[Process Object]) and the usage of a passive verb to express a state followed by a noun (associated process object) for the naming of EPC events (e.g. [Process Object]: Architecture [State]: Defined).

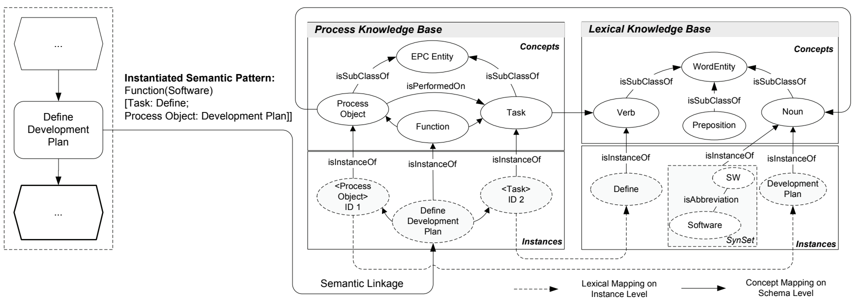

Figure 3: Example for Semantic Annotation of an EPC Function

The semantic annotation process starts with the extraction of used lexical terms in natural language expressions. Extracted lexical terms are the input for a term normalizer component that addresses the problems of word classification and naming conflicts. This approach reduces the number of potential naming conflicts to synonyms and abbreviations. Homonyms are neglected since we assume a non ambiguous meaning of used vocabulary in engineering domains. The determination of word classes (e.g. noun, verb) requires a matching between a lexical term and a lexical instance captured in the lexical knowledge base. The matching procedure considers semantic relationships (e.g. isAbbreviation) associated to a lexical knowledge base instance (e.g. SW is an abbreviation of Software). If a search for an existing lexical term within the lexical knowledge base is successful, the certain word class derives from the concept name. In case of naming conflicts, the term normalizer follows the rule to deliver the base word. For instance, SW has been identified as an abbreviation for software, consequently, the term normalizer delivers the term software as a noun. If a query for a lexical term in the lexical knowledge base delivers an empty result, an automatically driven word classification is not feasible. In this situation, the publicly available dictionary WordNet comes into play which is appropriate for the requirement of word classification and synonym detection. According to [LS04], it is *"optimized for lexical categorization and word-similarity determination"*. The term normalizer tries to retrieve semantic information by consulting WordNet. A WordNet query delivers either a set of word classes and synonyms (associated to the queried lexical term) or an empty set. In case of delivering an empty set the term normalizer component requires an interaction with the analyst in order to get a human classification entry. The term normalizer determines the word classes and resolves conflicts of lexical. For resolving the process semantics, this approach suggests the usage of semantic patterns to bridge this gap. Rolland and Achour [RA98] employ semantic patterns to extract a use case model from ambiguous textual use case descriptions. Semantic patterns allow the specification of lexical structures of EPC functions/events and its associated semantics that bridges the gap between informal and formal representation. Its declarative nature enables the analyst to define an arbitrary set of naming conventions expressed by means of lexical structures. The following discussion summarizes the concepts of using and specifying semantic patterns. The semantic pattern *Function (Name) [Task; Process Object]* consists of the process ontology concepts Task(T) and Process Object(PO). It is used as a template for naming an EPC function. This semantic pattern comprises the following two lexical structures

(LS) $LS_1$ = [Verb $\rightarrow_{map}$ T] [Noun $\rightarrow_{map}$ PO] and $LS_2$ = [Verb $\rightarrow_{map}$ T] [NounGroup$\rightarrow_{map}$ PO]. This semantic pattern and its defined lexical structures can be used to extract for example the semantics of the EPC functions *"Define Quality Goal"* or *"Design Architecture"*. The semantic pattern *Event(Name)[Process Object; State]* addresses the process ontology concepts Process Object and State. The following initial set of lexical structures is given $LS_1$ = [Noun $\rightarrow_{map}$ PO] [PassiveVerb $\rightarrow_{map}$ S] and $LS_2$ = [NounGroup$\rightarrow_{map}$ PO] [Verb $\rightarrow_{map}$ S]. *"Quality Goal Defined"* or *"Architecture Designed"* denote simple examples for this semantic pattern. Lexical structures defined for a semantic pattern type (function or event pattern type) provide the fundamentals in order to instantiate a semantic pattern.

The semantic pattern analyzer employs analysis rules to generate instantiated semantic patterns [RA98]. They are specified by a precondition and a body separated by a "→". The precondition is expressed on lexical terms and defines when the rule is applicable. The body denotes an action that generates the instantiated semantic pattern. As an example, the analysis rule $R_1$: IF MATCH([Verb $\rightarrow_{map}$ T] [Noun $\rightarrow_{map}$ PO]) → GENERATE(Function (Name) [T: Verb; PO: Noun]) applied to the EPC function *"Design Architecture"* (application domain software) results in the instantiated semantic pattern *Function (Software)[T: "Design"; PO: "Architecture"].* Analysis rules are used to determine the semantics of parameters (PA). The parameter type (Source Direction, Target Direction, Means Parameter) depends on its associated preposition (for, with and from). The rule $R_2$: IF MATCH([Verb $\rightarrow_{map}$ T] [Noun $\rightarrow_{map}$ PO] [Preposition ='For'] [Noun $\rightarrow_{map}$ PA]) → GENERATE(Elementary Function (Name) [T:Verb; PO: Noun; Target Direction Parameter:Noun]) generates an instantiated semantic pattern having a Target Direction Parameter. Finally, instantiated semantic patterns and its applied analysis rules are input for the ontology instance generator that populates the underlying lexical and process knowledge base. The instance generator is responsible for the establishment of the semantic linkage between EPC functions/events.

## 3. Extraction of Process Patterns

A process pattern represents a *common* or *best practice* solution to solve a particular problem in a certain context. We interpret *problem* in the sense of *how to achieve* a specific *process goal*. Goals are either functional or non-functional goals. Functional goals "*represent the purpose or the outcome that the business as a whole is trying to achieve. Goals control the behaviour of the business and show the desired states of some resource in the business."* [Me01]. Non-functional goals refer to expected process qualities such as customization, flexibility, interoperability, performance, safety, security, usability etc. The *process solution* associated to a process pattern can either be an ordered sequence of semantically described EPC functions or a set of interrelated process patterns. Employing process goals yields several advantages. Process goals enable a comparison of processes and process fragments. Process fragments pursuing the same or a similar process goal indicate variants in goal achieving. Hence, a process goal has at least one process solution. Process goals are not isolated items, they are organized as a hierarchical tree, representing a decomposition of goals into subgoals through AND/OR graph structures borrowed from problem reduction techniques in artificial intelligence [NI71]. The hierarchical tree consists of process goals and its semantic relationships that reflect the goal structure of the underlying engineering process. Each goal in the hierarchical tree has at least one assigned process pattern that provides a process solution to achieve that goal. Essentially, semantic goal relationships, captured in a hierarchical process goal tree, specify the semantic relationships between process patterns. Semantic relationships combine process patterns to larger pattern structures to unfold their full strength [HG04]. Examples for semantic relationships are hasSuccessor, isParallelTo, isAlternativeTo, hasRefinement and uses.

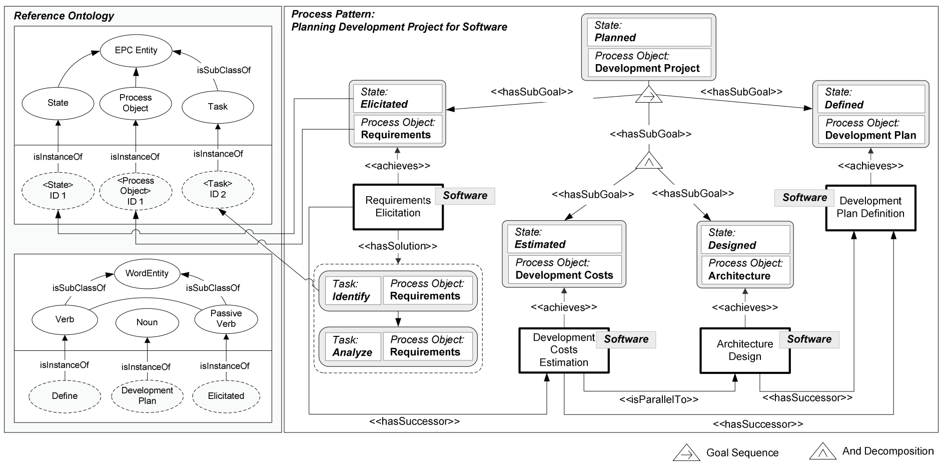

Figure 4: Example for a Process Pattern depicted as hierarchical Process Goal Tree

The right part of Figure 4 illustrates an example for a process pattern depicted as a hierarchical process goal tree. It can be interpreted as follows: in order to achieve the process goal *"Designed Architecture"* one may achieve the two process goals *"Elicitated Requirements"* and *"Estimated Development Costs"*. The process pattern *"Requirements Elicitation"* has the semantic relationship *<<hasSuccessor>>* to the process patterns *"Development Costs Estimation"* and *"Architecture Design"* that are in a *<<isParallelTo>>* relationship. The semantics of an *<<isParallelTo>>* relationship expresses that solutions of these patterns can be modeled as a parallelization or as a sequence. Figure 4 also sketches a process solution associated to the process pattern *"Requirements Elicitation"*. It comprises the two semantically described EPC functions *"Identify Requirements"* followed by "*Analyze Requirements"*. The reference ontology captures the semantics of process goals and semantically described EPC functions. Due to space limitations, these references are only depicted for the process pattern *"Requirements Elicitation"*, the lexical and concept mappings between lexical knowledge base and process knowledge base are also omitted.

EPC events provide the fundamentals for identifying functional process goals. Each event indicates a functional process goal. Figure 5 illustrates the interplay of semantically annotated EPC functions/events and the extraction of functional process goals and its associated process solutions for a part of the depicted process pattern in Figure 5. For reasons of simplicity the depicted reference ontology is reduced to its process concept instances. The top level goal *"Planned Development Project"* denotes the root goal and it indicates the overall goal satisfied by the software engineering process that is depicted as EPC model in the left part of Figure 5. This functional goal derives from the end event *"Software Development Project Planned"*.

Subgoals are derived from EPC events by considering the following, general applied mapping rule: if an event is a trivial event it indicates a leaf node within the goal tree, otherwise non-trivial events indicate a supergoal for preceding subgoals derived from trivial events. Hence, it is necessary to determine whether an event indicates a trivial event or a non-trivial event. This information can easily be retrieved by using a reference ontology. The reference ontology captures state information associated to a performed task, e.g. function *"Identify Requirements"* results to the state *"Requirement Identified"*. The event *"Software Requirements Elicitated"* indicates a non-trivial event.

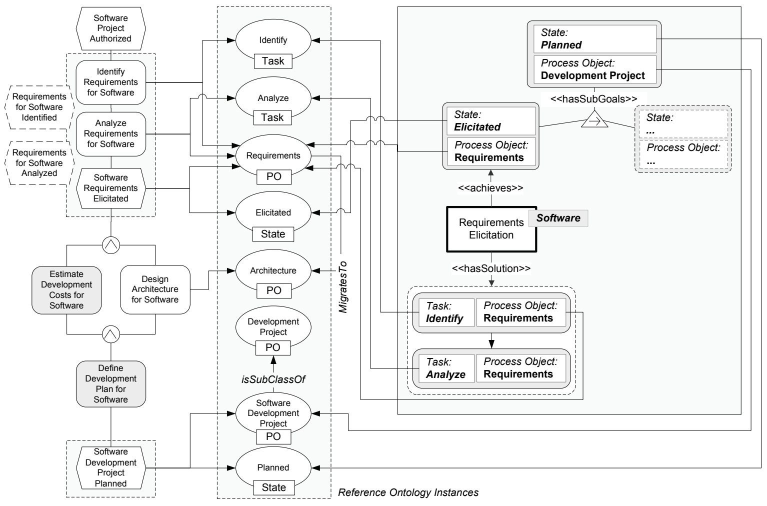
Software Project Authorized
Identify Requirements for Software
Requirements for Software Identified
Analyze Requirements for Software
Requirements for Software Analyzed
Software Requirements Elicitated
Estimate Development Costs for Software
Design Architecture for Software
Define Development Plan for Software
Software Development Project Planned
Identify
Task
Analyze
Task
Requirements
PO
Elicitated
State
Architecture
PO
Development Project
PO
isSubClassOf
Software Development Project
PO
Planned
State
MigratesTo
Reference Ontology Instances
State: Planned
Process Object: Development Project
<<hasSubGoals>>
State: Elicitated
Process Object: Requirements
State: ...
Process Object: ...
<<achieves>>
Requirements Elicitation
Software
<<hasSolution>>
Task: Identify
Process Object: Requirements
Task: Analyze
Process Object: Requirements

Figure 5: Extraction of Functional Process Goals from EPC Models

The not modeled event *"Requirements for Software Analyzed"* at the left part of Figure 5 would denote a trivial event. Hence, the functional process goal *"Elicitated Requirements"* denotes a supergoal for the not modeled functional goal *"Analyzed Requirements for Software"*. Using EPC events for extracting functional goals and for organizing them as a hierarchical tree requires to introduce a goal inheritance relationship. This means that a subgoal inherits all associated goals to a supergoal. This relationship comes into action if an EPC model comprises several end events, since each end event constitutes a supergoal.

Process pattern extraction from EPC models aims at conducting a construction of process patterns automated. It addresses the elicitation of functional process goals and the construction of a hierarchically organized goal tree. Elicitation of functional process goals also implies the assignment of semantically described EPC solutions. An approach for the extraction of non-functional goals is described in the following section.

## 4. Construction of Generic Process Patterns

Engineering processes are characterized by an identifiable progression from requirements, through specification to design and implementation. Each of these phases comprises context specific tasks, conducted similarly in different engineering domains [MIC00], e.g. the task *“Design Architecture”* usually follows the task *“Identify Requirements”*. Similarly conducted tasks in different engineering domains motivate the extraction of generic process patterns, tracing back to obviously existing structural analogies in process models. Generic process patterns reflect this similarity. They represent abstracted reference solutions for engineering domains.

The left part of Figure 6 depicts the generic process pattern *“Planning Development Project”* abstracted from two process patterns. Each of these process patterns captures a project planning process in different engineering domains. They are depicted as simplified process pattern models that omit their associated process goals. One may recognize the similarity between the two process patterns. The generic process pattern represents an abstracted solution for achieving the goal $G_F(M_2)$ *“Project Planned”* in the engineering domain. The generic process pattern is a composition of the process patterns *“Requirements Elicitation”*, *“Costs Estimation”* and *“Definition Development Plan”*. These patterns are coupled by means of the semantic relationship <<*hasSuccessor*>>.

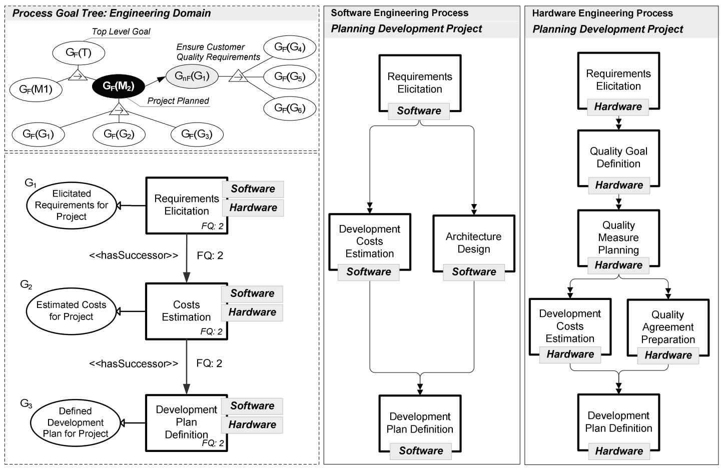
Process Goal Tree: Engineering Domain
Top Level Goal
G_F(T)
G_F(M1)
G_F(M_2)
Project Planned
Ensure Customer Quality Requirements
G_nF(G_1)
G_F(G_4)
G_F(G_5)
G_F(G_6)
G_F(G_1)
G_F(G_2)
G_F(G_3)
G_1
Elicitated Requirements for Project
Requirements Elicitation
FQ: 2
<<hasSuccessor>>
FQ: 2
G_2
Estimated Costs for Project
Costs Estimation
FQ: 2
<<hasSuccessor>>
FQ: 2
G_3
Defined Development Plan for Project
Development Plan Definition
FQ: 2
Software
Hardware
Software Engineering Process
Planning Development Project
Requirements Elicitation
Software
Development Costs Estimation
Software
Architecture Design
Software
Development Plan Definition
Software
Hardware Engineering Process
Planning Development Project
Requirements Elicitation
Hardware
Quality Goal Definition
Hardware
Quality Measure Planning
Hardware
Development Costs Estimation
Hardware
Quality Agreement Preparation
Hardware
Development Plan Definition
Hardware

Figure 6: Example for a Generic Process Pattern

The construction of generic process patterns relies on matching between process goals that calculate a similarity degree mapped to a matching table. The matching table consists of relations that are pairs of process goals having a similarity degree exceeding a predefined parameter value (e.g. 0.5). Process patterns achieving these goals are candidates for the generic process pattern construction. In a first step, a pair of matching process goals is abstracted to one generic process goal. Afterwards, assigned process patterns of a process goal matching pair represent candidates for a generic process pattern.

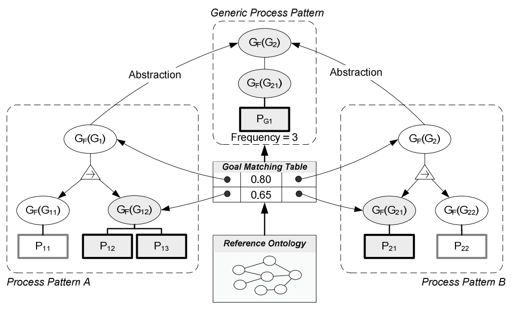

Figure 7: Generic Process Pattern Construction

The example in Figure 7 abstracts the process patterns $P_{12}$, $P_{13}$ and $P_{21}$ to the generic process pattern $P_{G1}$. Generic process patterns are characterized by a frequency value, expressing the number of instances a generic process pattern has. Abstracting process patterns to generic process patterns also requires the definition of semantic relationships between generic process patterns. Let us reconsider Figure 6, the generic process patterns *"Requirements Elicitation"* and *"Costs Estimation"* have a <<*hasSuccessor*>> relationship with a frequency of two. The semantic relationship and its associated frequency value results from evaluating the semantic relationships defined in process patterns that are candidates for generic pattern construction. Generic process pattern construction also enables identifying non-functional process goals. The non-functional goal *"Ensure Customer Quality Requirements"* consists of three functional subgoals (*"Define Quality Goals"* etc.) that achieve this non-functional goal. It results from the fact that three process patterns of the hardware engineering process do not have a similar equivalent within the software development model and each of these process patterns addresses the process object *Quality*. An automatic acquisition of non-functional goals is only capable to extract potential non-functional goal candidates; the final semantics must be specified and verified by the analyst.

## 5. Related Work

Related work primarily emerges from the areas process and plan ontology. Enhancing Business Process Management (BPM) with semantic web technologies to overcome obstacles in automated processing has triggered a new wave in research and practice (e.g. [He05] and the articles in [He07]). A lot of work already has been done in process ontology design. Business Process Management Ontology (BPMO) is a fully-fledged semantic business process modeling framework [Ya07]. Semantic EPC (sEPC) [HR07] has emerged from the SUPER Project [Su07] and aims at supporting the annotation of EPC models. Plan ontologies such as the Dolce+DnS Plan Ontology (DPPO) [Ga04] are founded on a theory of planning problems and on semantic descriptions of plans. [TF07] describes a similar approach that addresses the semantic annotation of EPC models. Summarizing the related, ontology-based approaches for a semantic process description, the proposed approaches do not focus on the semantics of natural language elements in EPC functions/events and provide scarcely solutions for an automatic population of underlying ontologies. Finally, we differentiate our process pattern approach from similar-sounding process mining. Process mining [Aa03] operates on process logs that are produced during process execution. These logs are used to construct adequately process specifications that model the behavior registered. Our mining approach operates on a given set of EPC models with the objective to construct generalized process patterns.

## 6. Conclusion: Tool Support and Practical Experiences

The presented approach for process pattern extraction from EPC models shows how to extract semantically described process patterns from EPC models and how to deal with the involved natural language problem. It can be used to bridge the gap between semi-formal process representations and formal ontologies. Semantically annotated EPC functions/events are input for process pattern extraction that identifies semantically described process goals of EPC functions and organizes them into hierarchies. Generic process patterns emerge from process patterns that have common process goals. The introduced approach has been prototypically implemented within the Business Process Improvement (BPI) project that was funded by Siemens Munich. We performed several analysis sessions for hardware and software engineering processes with our prototype. The achieved results are promising for reference model construction since we identified several common practices.

## Acknowledgements

We would like to thank Siemens CT SE3 for providing EPC models in engineering domains. Additionally, the authors want to thank Gustav Pomberger and Friedrich Roithmayr for their engagement and great support during the Business Process Improvement (BPI) project. Finally, we want to mention the several diploma students being part of the prototype development team, especially Martin Bartsch.

## References

[Aa03] van der Aalst, W.M.P.; et.al.: Workflow Mining: A Survey of Issues and Approaches. Data & Knowledge Engineering, (47) 2, 2003, p. 237-267.

[BDK07] Becker, J.; Delfmann, P.; Knackstedt, R.: Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In: Becker J.; Delfmann, P. (Ed.): Reference Modeling: Efficient Information Systems Design Through Reuse of Information Models. Physica, Heidelberg, 2007, p. 27-58.

[BPZ05] Basili, R.; Pennacchiotti, M.; Zanzotto, F.M.: Language Learning and Ontology Engineering: an Integrated Model for the Semantic Web. 2nd Meaning Workshop, Trento, Italy, 2005.

[Ga04] Gangemi, A. et.al.: Task Taxonomies for Knowledge Content D07, 2005. www.loa-cnr.it/Papers/D07_v21a.pdf, 13.09.2007.

[HG04] Hagen, M., Gruhn V.: Towards Flexible Software Processes by using Process

Patterns. 8th International Conference on Software Engineering and Applications. Cambridge, USA, 2004, p. 436-441.

[He05] Hepp, M. et.al.: Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. IEEE International Conference on e-Business Engineering. Beijing, China, 2005, p. 535-540.

[He07] Hepp, M. et.al. (ed.): Proceedings on Semantic Business Process and Product Lifecycle Management, 3rd European Semantic Web Conference. Innsbruck, Austria, 2007.

[HR07] Hepp, M.; Roman, D.: An Ontology Framework for Semantic Business Process Management. 8th International Conference Wirtschaftsinformatik. Karlsruhe, Germany, 2007, p. 423-440.

[JS07] Janiesch, C.; Stein, A.: Adapting Standards to Facilitate the Transition from Situational Model to Reference Model. 10th International Workshop on Reference Modeling (RefMod). Brisbane, Australia, 2007, p. 1-12.

[LS04] Liu, H.; Singh, P.: ConceptNet – A Practical Commonsense Reasoning Tool-Kit, BT Technology Journal, (22) 4, 2004, p. 211-226.

[Me01] Mendes, R. et.al.: Understanding Atrategy: a Goal Modeling Methodology. 7th International Conference on Object-Oriented Information Systems, Calgary, Canada, 2001, p. 437-446.

[MIC00] Moore, J.; Inder, R.; Chung, P.: Combining and Adapting Process Patterns for Flexible Workflow. 11th International Conference on Database and Expert Systems Applications, London, United Kingdom, 2000, p. 797-801.

[Ni71] Nilsson, N.J.: Problem Solving Methods in Artificial Intelligence. McGraw Hill, New York, 1971.

[PG05] Pfeiffer, D.; Gehlert, A.: A Framework for Comparing Conceptual Models. Workshop on Enterprise Modelling and Information Systems Architectures. Klagenfurt, Austria. 2005, p. 108-122.

[RA98] Rolland, C.; Achour B.C.: Guiding the Construction of Textual Use Case Specifications. Data & Knowledge Engineering, (25) 1-2, 1998, p. 125-160.

[Ro95] Rosemann, M.: Komplexitätsmanagement in Prozeßmodellen. Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung. Gabler, Wiesbaden, 1996.

[SBK07] Schermann, M.; Böhmann, T.; Krcmar, H.: Fostering the Evaluation of Reference Models: Application and Extension of the Concept of IS Design Theories. 8th International Conference Wirtschaftsinformatik. Karlsruhe, Germany, 2007, p. 181-198.

[Sc98] Schütte, R.: Grundsätze ordnungsgemäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle. Gabler, Wiesbaden, 1998.

[Su07] Integrated Project Semantics Utilised for Process Management within and between Enterprises. http://www.ip-super.org, 13.09.2007.

[TF07] Thomas, F., Fellmann, M.: Semantic Business Process Management: Ontology Based Process Modeling Using Event-Driven Process Chains. International Journal of Interoperability in Business Information Systems, (2) 1, 2007, p. 29-44.

[WN06] RDF/OWL Representation of WordNet: W3C Working Draft June 2006. http://www.w3.org/TR/wordnet-rdf/, 13.09.2007.

[Ya07] Yan, Z. et.al.: BPMO : Semantic Business Process Modeling and WSMO Extension. International Conference on Web Services. Salt Lake City, USA, 2007, p. 1185-1186.