A Semantic Design Pattern Language for Complex Event Processing

Adrian Paschke
Free Universtity Berlin
paschke AT inf.fu-berlin.de

Abstract

Complex Event Processing (CEP) is an emerging enabling technology to achieve relevant situational knowledge from distributed systems in real-time or almost real- time. It gains more and more attention due to the industry hype and momentum regarding situation-aware computing and service-orientation. Support for CEP is considered to become a critical success factor for many businesses, especially regarding the recent failures of many industrial ventures due to insufficient situation-awareness. However, first experiences in using the CEP technology and setting up CEP applications have shown that the potential adopters have major problems in understanding and adequately designing and implementing successful CEP solutions. CEP engineering remains a laborious trial and error process with slow development and change cycles.

The purpose of this paper is to contribute with a multi-dimensional categorization of CEP (design) patterns and a semantic design pattern language to describe successful CEP designs and build up libraries of CEP best practice descriptions and CEP patterns. This will lead towards a more structured approach for the design of CEP systems and will allow for the exact characterization and comparison of a broad variety of CEP media and systems. The intention is to support an interdisciplinary and efficient communication process about design solutions for various CEP problems in order to reach a thoroughly understanding of the proposed CEP pattern approach which offers an additional increase (1) in efficiency, aimed at cheaper and faster delivery of CEP-based systems by supporting CEP system engineers in their design decisions, and (2) in reusability of successful CEP solutions to frequently occurring CEP design problems in different domains.

1 Introduction

In recent years we have seen the rise of a new type of software called Complex Event Processing (CEP) media. These are systems to detect and process complex events from large numbers of real-time events, so called event-clouds, occurring in distributed, possibly heterogeneous systems, databases and applications. This addresses an urgent need businesses do have nowadays: detection, prediction and mastery of complex situations as a crucial prerequisite to the efficiency of a dynamic service-oriented environments and the competitiveness of networked businesses in a future Internet of Services.

However, there are a number of risks and difficulties that have to be taken into account when employing CEP. First industrial experiences in using the CEP technology and setting up CEP applications have shown that the potential adopters have major problems in understanding the CEP approach and adequately designing and implementing successful CEP solutions. Best practices and discussions about frequently occurring problems and their solutions in specific CEP application domains are missing and a systematic and profound debate about CEP patterns and anti-patterns is still in its early stage. CEP engineering remains a laborious trial and error process with slow development and change cycles. Hence, necessary theoretical groundwork to reach a thoroughly understanding of the proposed CEP methodology and technology and its applications in various problem domains needs to be done.

This paper should "prepare the ground" for a more structured and methodological CEP engineering approach. The goals are to:

- provide a common set of terms describing the CEP technology and CEP media;
- provide a common frame of reference as basis for the understanding the core properties of a model for complex events and CEP system architectures, and build-up a library of reference architecture, models and pattern for CEP design;
- help analyzing and understanding the dimensions of the CEP technologies and CEP media, and their interdependencies;
- support the selection of the right CEP scenario or an appropriate CEP media,
- assist the conceptual design of domain-specific CEP solution and support the abstraction necessary for the development of generic CEP systems

The intention is to stimulate constructive feedback from the pattern community[1], CEP community[2] and closely related

[1] PLoP and EuroPLoP conference series

[2] http://complexevents.com

communities such as the Reaction Rules community[3] in order to achieve a more general consensus about the proposed CEP architectures and patterns and build-up comprehensive CEP pattern libraries and reference architectures. Clearly, cataloguing and publishing CEP patterns is a community effort. The grand challenge is to make the CEP technology tractable by easy-to-use methods, technologies and tools with support for e.g. semantic search and tailored semantic integration. While the main intention for the proposed CEP pattern approach is to support CEP engineers in their design decisions for building robust CEP solutions with well understood tradeoffs, there are other use cases as well, e.g.:

- they can be used as a documentation tool, making it easier for a team to absorb new CEP developers
- they can be used to compare existing CEP systems and point out equivalences and differences;
- they lay the basis for the detection of peculiarities in existing CEP systems and analyze their root causes

This paper is structured in the following way:

In section 2 a multi-dimensional CEP design pattern classification scheme is introduced. Section 3 develops a semantic CEP design pattern language. In section 4 we describe a general approach of applying the CEP pattern-based approach in the design and engineering of CEP solutions Finally, in section 5 we conclude this work and give an outline to areas of future work.

2 A Multi-Dimensional Categorization for CEP Patterns

The main goal for the multi-dimensional categorization scheme for CEP patterns introduced in this section is to prepare the common ground for a descriptive characterization and classification of frequently used and successful patterns in CEP. This categorization forms the basis for clustering CEP patterns into vertical domain-specific and generic horizontal across-the-domain dimensions. These dimensions can be further extended, accordingly. IT allows for a more thorough discussion on a common general CEP pattern language and adequate notational representation formats from which further domain-specific CEP pattern languages can be derived as subclasses. It also acts as common ground for the discussion on and improvement of existing CEP patterns and the description of new ones.

The collected, described and categorized patterns will support CEP application engineers in their design decisions, but will also provide many other use cases as well, e.g. the CEP patterns can be used as a documentation tool, making it easier to understand the tradeoffs of a particular domain-specific CEP solution, open new markets based on exploiting the CEP technology or provide means for an IT team to absorb new CEP developers such as the new role of a *CEP event modeler*.

2.1 Categorization according to Good and Bad Solutions

The first category distinguishes between successful CEP patterns and "bad" CEP Anti-patterns describing inefficient solutions.

CEP patterns

CEP patterns document a successful solution to a frequently occurring problem.

CEP Anti-pattern

CEP anti-patterns are conceptually similar to CEP patterns in that they document recurring solutions to common design problems. They are known as *anti*-patterns because their use (or misuse) produces negative consequences. Anti-patterns document common mistake made during CEP development as well as their solutions.

Note: In the following when we speak of patterns we always mean both patterns and anti-patterns, i.e. [anti-]patterns, unless we explicitly address only one class.

2.2 Categorization according to the Abstraction Level

The second category distinguishes between the levels of abstraction reaching from CEP application management, to architectural design, to concrete development, deployment and optimization patterns.

Guidelines and Best Practices

More or less informally described guidelines and best practices for the design, development, deployment, and management of CEP applications.

Management patterns

Management patterns address the management of CEP applications, i.e. they adopt general IT Service Management solutions and best practices such as ITIL[4] to the domain of CEP applications and services as described them as patterns.

Architecture patterns

CEP Architecture patterns are high level patterns on how CEP systems are laid out and how CEP large systems are divided. These typically account for the major components, their externally visible properties, the major functionality of each component, and the relationships between them.

[3] http://ibis.in.tum.de/research/ReactionRuleML

[4] www.itil.org

Design patterns

Design patterns provide a scheme for refining the subsystems or components of a CEP application, and the relationships between them. They describe a commonly recurring structure of event-communicating components that solves a general design problem within a specific context.

Mapping patterns

Mapping patterns connect design patterns to populate a larger solution, i.e. efficiently tailor successful design patterns to a concrete CEP product / application. These (product) mappings are based on proven implementations.

Idioms / Realization patterns

Idioms are realization patterns on the technical implementation level and may be specific to an event processing language (EPL) or CEP engine. An idiom guides the implementation aspects of CEP components and the relationships between them, using features specific to a given EPL or environment.

Smells / Refactoring patterns

Smells are related to CEP idioms, but usually address specific structures or parts in a concrete technical CEP implementation, e.g. structures in the complex event definition, that can be improved by the application of refactoring. The definition of smells is generally relatively informal as compared to [anti-] patterns.

Refactoring

Refactorings are transformations to improve the overall quality of a solution/implementation, in particular on the technical (code) layer on the level of smells and idioms, e.g. the optimization of a complex event pattern.

2.3 Categorization according to the Intended Goal

The third categorization distinguishes CEP patterns according to their intended goal, mainly from the view of a CEP solution provider, i.e. which kind of problems in employing the CEP technology should be solved by the pattern.

Adoption patterns

Adoption patterns describe general strategic decision patterns which ease or speed-up (resp. delay or hinder) the adoption of CEP solutions and tools by business and customers.

Business patterns

Business patterns identify the interaction between CEP customers, businesses, and event data. Business patterns are used to create end-to-end CEP business applications.

Integration patterns

Integration patterns connect other business patterns together to create CEP applications with advanced functionality. Integration patterns are used to combine business patterns in advanced CEP applications.

Composite patterns

Composite patterns are combinations of business patterns and integration patterns that have themselves become commonly used types of CEP applications. Composite patterns are advanced CEP applications.

Workflow patterns

Workflow/Process patterns define the concrete process flow in a CEP system or application, hence are concrete specifications of business processes (business patterns) and/or application workflows (integration and composite patterns). Several general workflow patterns have been described e.g. in (Aalst, Hofstede et al. 2003).

Coordination patterns

Coordination patterns partially overlap with workflow and process patterns. But while such process or workflow patterns describe the control flow of the business or CEP application logic, the coordination patterns focus on the different points of the interaction between components in a CEP business process, i.e. describe successful coordination protocols. Here, with protocol we do not mean a low level communication protocol, but a high-level coordination protocol which determines the possible (inter-)actions the application can choose in the context of a detected complex event and an occurred situation. Several well-known coordination and negotiation protocols from multiple disciplines have been collected e.g. in the Negotiation Pattern Library (NPL) (Paschke, Kiss et al. 2006).

Customized patterns

Customized patterns are similar to composite patterns, as they combine business patterns and integration patterns to form an advanced, end-to-end solution. These solutions, however, have not been implemented to the extent of composite patterns, but are instead developed to solve the CEP application problems of one specific company, or perhaps several enterprises with similar problems.

Application patterns

Application patterns are driven by the customer's requirements and describe the shape of CEP applications and the supporting runtime needed to build the CEP application.

2.5 Categorization according to the Management Level

Finally, the last category makes a general classification of CEP patterns into strategic patterns, tactical patterns and operational pattern, i.e. they describe design or management decisions on the operational, tactical and strategic level of CEP service management.

Strategic patterns

These type of patterns (aka CEP business value management patterns) describe the strategic alignment of the CEP-based IT. They are an integral part of the enterprise governance and describe successful leadership and organizational structures and processes that ensure that the organization's CEP infrastructure sustains and extends the organization's strategy and objectives. They are part of the general IT governance strategy of an enterprise.

Tactical patterns

These type of patterns superimpose the management patterns and describe best practices for processes that cooperate to ensure the persistent quality of CEP applications, according to the levels of service agreed to by the customer. Typically such processes are superimposed on IT Service Management (ITSM) domains such as systems management, network management, systems development, and on many process domains like change management, asset management and problem management.

Operational patterns

Operational patterns focus on optimizing the management of the CEP application infrastructure, i.e., the components it contains and the data it creates. It relates to IT infrastructure management (ITIM).

2.4 Multi-dimensional mapping of CEP pattern categorizations levels

Based on these three categories[5] we can derive a multi-dimensional categorization scheme which reveals connections and dependencies between these three levels of CEP patterns patterns as shown in figure 1.

[5] We omit the operational, tactical and strategic categorization level here since it superimposes the other categories

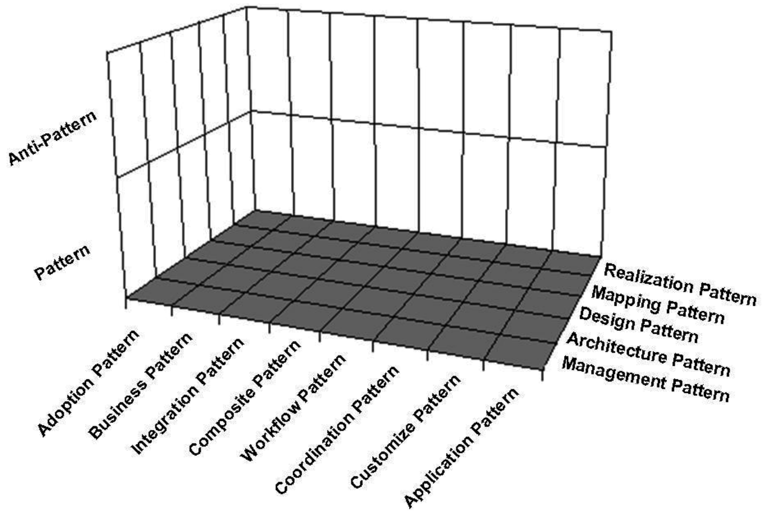


Figure 1: Three-dimensional categorization scheme for CEP patterns

The following example shows a refactoring pattern which addresses inconsistency in a complex event pattern definition due to exceptional situations.

Name	Exception to the Complex Event
Description	A complex event is not detected in a particular situation. This situation can be described by ¬EXEC
Pattern definition before refactoring	… (A, B, C) …
Pattern definition after refactoring	… ¬EXEC (A, B, C) …
Addresses (Smell):	Inconsistency

3 Semantic CEP Pattern Language

There are many existing design pattern languages (see e.g. Gamma 1995, Hillside.net , proceedings of the major PLoP conferences) which are primarily narrative descriptions, sometimes enhanced with formal models like UML diagrams to define design patterns. While this clearly supports software engineers to understand and use the patterns, it does not provide support for tools such as an online pattern library which should initiate an interdisciplinary community effort for building up comprehensive CEP pattern catalogues as toolboxes for CEP vendors, engineers and adopters. To address this we make use of Semantic Web technologies, in particular using the web ontology language (OWL) which facilitate shared terminologies and powerful inference processes to deal with inconsistencies and conflicts between used terminologies. By using OWL Lite we define a formal ontological CEP pattern language (ontology).

This allows meeting the following design objectives:

- A formal, machine-readable pattern definition language

- An open and extensible representation language facilitating shared terminologies
- A human-readable language with structured narrative and formal pattern descriptions
- Meta data annotations that allow reasoning about pattern descriptions, in particular to provide efficient search functionalities
- A format that fits nicely into the standard Web technologies and the upcoming Semantic Web

The CEP Pattern Language is implemented as a semi-formal XML Pattern Description Language (PDL) with a combination of machine-processable XML mark-up, human-readable narrative descriptions and a formal ontology design language (DOL) implemented as an OWL vocabulary model (see figure 2).

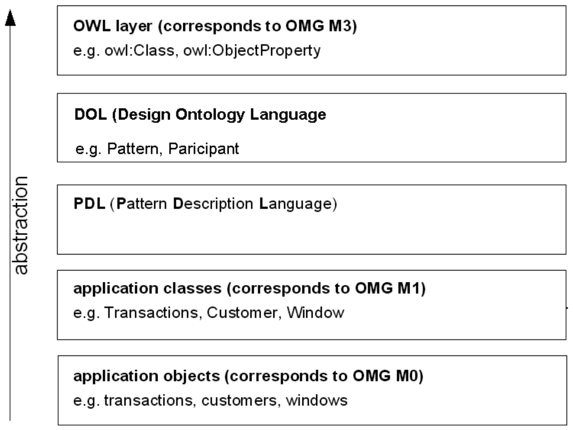
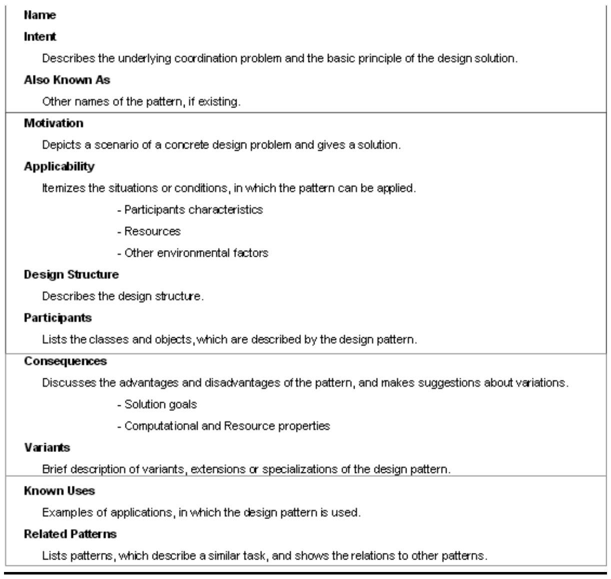

OWL layer (corresponds to OMG M3)
e.g. owl:Class, owl:ObjectProperty
DOL (Design Ontology Language
e.g. Pattern, Paricipant
PDL (Pattern Description Language)
application classes (corresponds to OMG M1)
e.g. Transactions, Customer, Window
application objects (corresponds to OMG M0)
e.g. transactions, customers, windows
abstraction

Figure 2: CEP Pattern Language Meta Model

Standard Semantic Web vocabularies (RDF based Dublin Core) are used to attach further meta data to the pattern definitions.
We adopt the scheme of software design patterns as described in the Gang of Four book (Gamma 1995) and adapt it to the needs of our CEP engineering domain. The XML Schema of the pattern languages conforms to the structure shown in figure 3 and corresponds to the concepts defined in the ontology can then be used to formally define design patterns.

The core vocabulary of the pattern language is defined by the design ontology language (DOL) in which the concepts - such as Pattern, Participants etc. - needed to describe design patterns are defined. The DOL ontology also defines relationships of classes and their properties. The following script shows part of DOL:

```
<rdf:Description rdf:about="#Pattern">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<!-- participant -->
<rdf:Description rdf:about="#Participant">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
...
<!-- subpattern relationship -->
<rdf:Description rdf:about="#isSubPatternOf">
<rdfs:domain rdf:resource="#Pattern"/>
<rdfs:range rdf:resource="#Pattern"/>
<rdf:type rdf:resource=
"http://www.w3.org/2002/07/owl#ObjectProperty">
<rdf:type rdf:resource=
"http://www.w3.org/2002/07/owl#TransitiveProperty"/>
</rdf:Description>
...
```



Name
Intent
Describes the underlying coordination problem and the basic principle of the design solution.
Also Known As
Other names of the pattern, if existing.
Motivation
Depicts a scenario of a concrete design problem and gives a solution.
Applicability
Itemizes the situations or conditions, in which the pattern can be applied.
- Participants characteristics
- Resources
- Other environmental factors
Design Structure
Describes the design structure.
Participants
Lists the classes and objects, which are described by the design pattern.
Consequences
Discusses the advantages and disadvantages of the pattern, and makes suggestions about variations.
- Solution goals
- Computational and Resource properties
Variants
Brief description of variants, extensions or specializations of the design pattern.
Known Uses
Examples of applications, in which the design pattern is used.
Related Patterns
Lists patterns, which describe a similar task, and shows the relations to other patterns.

Figure 3: CEP Pattern Language

The ODL concepts are then used to defined CEP design patterns in a machine-interpretable format.

The semantics of the CEP pattern language consists of three parts:
1. Explicit semantics intended for machine processing (reasoning about the ontology, reasoning to establish trust).
2. Formal semantics for human processing (the participants and their properties and relationships that have to be mapped to the programming language).
3. Informal semantics for human processing (further guidelines, such as comments in the ontology, helping the programmer to write applications which are consistent with the intended meaning of the concepts in the ontology).

4 CEP System Design and Engineering

The process of designing and engineering CEP applications is composed of a *requirement analysis phase* and a *synthesis phase*. In the analysis phase the needs and desires for a given CEP design situation are investigated and transformed into a more or less formal set of requirements and constraints. The extracted requirements are then used in the synthesis phase for the selection of the *technological spaces*[6] and the reference architecture,

[6] A technological space is, in short, a zone of established expertise and ongoing research in a field. It is a working context together with a set of associated concepts, body of knowledge, tools, required skills, and possibilities and aims to improve efficiency of work by using the best possibilities of different technologies.

reference models and CEP patterns which should be applied and tailored to the context of the application domain in order to implement a concrete CEP solution. In the following we propose a general approach to systematically determine the requirements and constraints in a specific CEP design situation match them to the documented design variations written down in terms of reference architectures, models and patterns, and finally decide which one works best for the intended purpose.

CEP Scenarios can be regarded as models for complex event processing applications with certain parameters. CEP is typically executed according to a model. A design criterion represents a distinctive CEP scenario property, which is represented as a rule and is associated with a number of potential criterion values (or rule expressions and design goals). These values embody the variation of the semantics that can be defined for a CEP scenario regarding a certain property or rule. The sum of all criteria values assigned to a specific CEP scenario, medium, or instance constitutes its type (aka usage context).

We distinguish two orthogonal dimensions of design criteria: *exogenous / endogenous* and *explicit / implicit decision criteria*. While exogenous criteria are determined by the business context in which the CEP application is supposed to be situated and can not or only to some extend be influenced by the designer, endogenous criteria represent the choices and parameters which can be directly selected within the design of a CEP application. Explicit criteria can be directly specified and quantified. Implicit criteria assess the consequences of explicit criteria. They are not written down but determined during the execution of a particular CEP application. Endogenous criteria depend to some extent on the exogenous criteria and have to be analyzed and mapped into corresponding endogenous explicit criteria which are then applied in the synthesis phase for the selection and implementation. The observation and results of the execution can then be evaluated with endogenous implicit criteria at run-time or proactively on the basis of a theoretical run-time approximation such as a simulation or experiment, e.g. discrete event simulations. Table 1 shows some typical questions in each category from which to derive the requirements.

Table 1: Categorization of CEP design criteria

	Exogenous	Endogenous
Explicit	Rules or quantifiable knowledge about the business context	Choices made by the scenario designer
Implicit	Domain knowledge which cannot be directly determined or represented in a formal way, e.g. ethical standards	Facts determined through assessing the CEP execution

Once the type, i.e. the usage context, of a CEP scenario, medium, or instance has been specified in terms of rule expressions and design goals it can be matched against the library of CEP reference models and patterns. Manual comparison, especially of large usage contexts against reference models / patterns, is tedious and error-prone. Therefore, are prerequisite is that the patterns and reference models are semi-formally described and provide semi-automated means to reuse them for devising concrete solutions by customizing them to a particular usage context, i.e. tailor (known as 'variability') them to the type of the CEP scenario. We assume that this type of support must allow an intuitively understandable description of the reference model's and pattern's variability at design level, since it is sensible to expect that the primary user/creator of these models and patterns is mostly the non-technical business person. Nevertheless, these descriptions should be executable in an automated way. To specify variability in nearly natural language is to exploit (business) rules enhanced with semantic annotations which map to semantic design pattern language to declaratively represent the restrictions (constraints) and their combinations in form of rules. To raise the level of abstraction at which these rules are specified, the availability of a model-driven approach for rules is also a prerequisite. Our rule-based implementation the OMG model driven architecture (MDA) approach:

1. On the computational independent level rules are engineered in a business rules management system (Rule Responder) in a natural controlled English language (ACE RuleML) using blueprint templates and domain-specific design vocabularies (ontologies).
2. The rules are mapped and serialized in RuleML XML which is used as platform independent rule interchange format to interchange rules between rule inference services and arbitrary rule execution environments. RuleML is the quasi-standard for Web rules. It consists of a huge family of rule languages including SWRL, Derivation RuleML, Reaction RuleML (with PR RuleML and CEP RuleML), etc. It is also the basis for the upcoming W3C RIF (Rule Interchange Format) Standard which includes rule dialects for logic rule languages, production rule languages and reactive rule languages.
3. The RuleML rules are translated into the platform specific Prova rule language for execution. Prova is both a Semantic Web rule language and a highly expressive distributed Semantic Web rule engine which supports, complex reaction rule-based workflows, rule-based complex event processing, distributed Web inference services / agents deployed on an enterprise service bus middleware, rule interchange, declarative (backward reasoning) decision rules and dynamic access to external data sources such as databases, Web services, and Java APIs. Prova follows the spirit and design of the W3C Semantic Web initiative and combines declarative rules, ontologies (vocabularies) and inference with dynamic object-oriented programming and access to external data sources via query languages such as SQL, SPARQL, and XQuery. One of the key advantages of Prova is its elegant separation of logic, data access, and computation as well as its tight integration of Java, Semantic Web technologies and enterprise service-oriented computing and complex event processing technologies.

The translation between the controlled English rule language and RuleML is based on a vocabulary template-driven approach in combination with a controlled English translator. XSLT is used for the translation from RuleML into Prova execution syntax.

Through applying the semantically formalized design goals and rule expressions to the formalized variability points (described in terms of rules and constraints) design candidates can be extracted and ranked list of reference models respectively patterns from the library can be created, from which the CEP designer can choose.

5 Conclusion and Future Steps

Detection, prediction and mastery of complex situations are crucial to the competitiveness of networked businesses in the emerging Internet of Services and the efficiency of dynamic distributed infrastructures in manifold domains such as Finance/Banking, Logistics, Automotive, Telco, Life Sciences. Complex Event Processing (CEP) is an emerging enabling technology to achieve actionable, situational knowledge from distributed heterogeneous systems, databases and applications/services in real-time or almost real- time. The CEP technology is considered as one of the main prerequisites for many highly relevant technology trends such as predictive business, real-time adaptive enterprise or autonomic systems. First experiences in setting up CEP-applications have shown that the potential adopters have major problems in adequately designing and implementing successful CEP solutions to frequently occurring domain-specific problems. CEP engineering remains a laborious trial and error process with slow development and change cycles.

In “classical” software engineering there is a rather large body of research in the area of design pattern description. A wide range of formalisms are used to describe patterns including UML Profiles (Dong 2003) and logic programs (Krämer 1996). Eden (Eden, 2002) proposes a logic based framework that allows precise definitions of patterns, pattern instances, and pattern refinement. His higher order entities are used to address the problems which we are trying to solve with pattern aggregation.

In this paper we have presented a general methodological framework and the basic concepts and classification schemes to further evolve a pattern and model based engineering approach for CEP applications in a more structured way. The normative foundations given in this paper should prepare the theoretical ground for a more thorough discussion of the proposed CEP methodology and technology and should contribute to an interdisciplinary and efficient communication process about design solutions for various CEP problems. Obviously, this also amounts for empirical knowledge that can only be obtained from the practice in an inter-disciplinary community effort to which this paper contributes. Since a comprehensive library of reference models and patterns does not yet exist for CEP applications we can identify three important areas for future research:

1. Determine, describe and categorize best practices and successful CEP solutions according to the pattern categorization scheme introduced in this paper. This should lead to a detailed and comprehensive library of domain-specific and across-the-domain CEP reference architectures, reference models and patterns.
2. Develop typical design criteria and describe them in a declarative rule-based representation which allows for the automated complex mapping of exogenous criteria to endogenous criteria, from implicit to explicit criteria can be undertaken, thus allowing a designer to choose the ”right” scenario for a given business and CEP application context, which might be characterized through exogenous classification criteria.
3. Significant efforts are necessary to come up with a (semi-)formal specification/modeling framework facilitating the (semi-) automated generation of new CEP applications by customization of reference architecture and models, and their solution-oriented design pattern specifications to the context of an application domain;

These future contributions, together with the categorization and the semantic design pattern language provided in this paper, should illustrate that first steps towards an design pattern engineering approach for CEP applications are already undertaken and that there will be a community building effort to make the CEP technology tractable by easy-to-use methods, technologies and tools, and to provide integrated solutions and best practices to practitioners in major industry sectors.

References

Aalst, W. v. d., A. t. Hofstede, et al. (2003). "Workow patterns." MIS Quarterly **14**(1).

Hillside.net, *Design Patterns Homepage* *http://hillside.net/patterns/*

Gamma, E., et al., *Design Patterns - Elements of Reusable Software*. 1995: Addison-Wesley

Dong, J., Yang, S.: Visualizing Design Patterns With A UML Pro_le, Proceedings of the IEEE Symposium on Visual/Multimedia Languages (VL), pp123-125, Auckland, New Zealand, 2003.

Krämer, C., Prechelt, L.: Design Recovery by Automated Search for Structural Design Patterns in Object-Oriented Software, Proceedings of the 3rd Working Conference on Reverse Engineering (WCRE '96), 1996.

Eden, A. H.: LePUS, A Visual Formalism for Object-Oriented Architectures, The 6th World Conference on Integrated Design and Process Technology, Pasadena, California, June 2228, 2002.