Complex Event Processing

CORDYS
Simplifying Business

Radboud Universiteit Nijmegen

Master Thesis Computer Science
Radboud University Nijmegen
October 2007

Name: Paul Dekkers
Thesis number: 574

Supervisors:

prof. dr. M. van Vliet	(Faculty of Computer Science)
dr. P.E.M. Ligthart	(Faculty of Management Sciences)
ir. H. de Man	(Cordys)
ir. G. Ligtenberg	(Cordys)

Colophon

Radboud University Nijmegen
Nijmegen Institute for Computing and Information Sciences (NIII)
Toernooiveld 1
6525 ED Nijmegen

Cordys (R&D NL department)
Vanenburgerallee 3
3882 RH Putten

Abstract

Nowadays enterprises are more complex than ever. Different processes take place all over the world and events are flying through the enterprise IT systems. These systems have grown from standalone applications that were able to handle a certain aspect within an enterprise to an enterprise wide IT system that provides a coupling between the different IT applications.

These enterprise wide IT systems are widespread across large enterprises and generate many events that flow through the enterprise system layers. The events feed other applications or services which generate new events on their turn. We can truly speak about an event-cloud that hangs within an enterprise. Because of this event-cloud the event-flow of an enterprise IT system becomes non-transparent and difficult to understand.

A new innovation is arising that can help tackle this problem: Complex Event Processing (CEP). With CEP it is possible to correlate events and detect complex situations. This thesis deals with a number of CEP related questions. The first part describes surrounding concepts that help understanding what CEP really is about. The second part introduces a general CEP language and six representative cases. Three different CEP engines are evaluated and each case is expressed for all engines. To complete the impression of these engines also some performance tests are run.

The third and last part is not directly about CEP but more about innovation management. It introduces innovation management on different levels and the integration between these levels. This will help to bring structure to innovations.

Table of contents

List of Figures

List of Definitions

List of Tables

Preface

This thesis is the result of a seven month lasting research and is the final assignment of the computer science education at the Radboud University of Nijmegen (RU). Because I chose the specialization of Management & Application (MA) the research is conducted at an external organization. In this case Cordys, a company that is on the edge of technology with their composite Business Process Management Suite (BPMS).

This thesis would not be as it is now without the contribution of several people, which I would like to thank for their effort and support. First of all I would like to thank my four supervisors: Mario van Vliet (supervisor computer science, RU), Paul Ligthart (supervisor applied management, RU), Henk de Man (supervisor content, Cordys), and Gerwin Ligtenberg (global supervisor/manager, Cordys).

For their help, time, and support I would like to thank Thomas Bernhardt (Esper), Ashwin Jayaprakash (StreamCruncher), and Marco Seiriö (ruleCore Server).

During the research some progress meetings were held at Cordys where I showed my latest findings. Fruity discussions helped me to fine-tune my thesis. For their time and effort I would like to thank Theodoor van Donge, Thijs Petter, and Hans Bank.

Last but not least I would like to thank Cor van Dijk and Mike Moran, both 'room-mates' at Cordys, for their help and company. Especially in the last part where most of the writing was done, they really helped a lot with some English 'problems', or as Mike likes to call it: "Dunglish".

As English is not my native language and even though Cor and Mike helped a lot, there still can be some misspelling or erroneous sentences in this thesis, which I would like to apologize for in advance.

1 Introduction

The subject of this thesis is complex event processing. Because this is a very broad research area there has to be focused on a section of this research area. This chapter starts with an introduction to complex event processing. After that more details about the research are given.

1.1 Complex Event Processing

Nowadays enterprises are more complex than ever. Different processes take place all over the world and events are flying through the enterprise IT systems. These systems have grown from standalone applications that were able to handle a certain aspect within an enterprise to an enterprise wide IT system that provides a coupling between the different IT applications.

The world of enterprise IT systems is full of acronyms. Some of the latest are SOA (Service-Oriented Architecture) and EDA (Event-Driven Architecture) which both are architectures that enable the coupling of IT applications and allow integrating processes, distributed and enterprise wide, possible even among multiple enterprises.

These enterprise wide IT systems are widespread across large enterprises and generate many events that flow through the enterprise system layers. The events feed other applications or services which generate new events on their turn. We can truly speak about an event-cloud that hangs within an enterprise. Because of this event-cloud the event-flow of an enterprise IT system becomes non-transparent and difficult to understand. The simplest events are traceable, but the more complex events (which consist of multiple, unrelated simple events) are hard to keep track off.

To tackle this problem and to make more use of complex events a new acronym is introduced: CEP (Complex Event Processing). CEP can be used to view and react to complex events, in real time. When a problem or opportunity arises it should be

noticed right away, in real time, to make sure the right action can be taken at the right moment. Otherwise there will only be historical data that reveals possible problems which already have become a real problem or opportunities which already have vanished. With CEP it is possible to act in real time and make better use of the already available events in an enterprise.

1.2 Research questions

In the following two subparagraphs the main research questions with the corresponding sub-questions are introduced. In the next paragraph further detail about the research is given.

1.2.1 Main research questions

The main research questions are defined as followed:

a) "What are the key concepts involved with complex event processing (CEP)?"
b) "What is an optimal engine for including CEP within the Cordys environment?"
c) "What are factors that involve innovation management on different levels and how do they integrate?" (Innovation management research, involving Cordys situation)

1.2.2 Sub-questions

Some sub-questions that will help answering the main questions.

Sub-questions with question 1:

a) "What are the prerequisites for a CEP engine in regard to the information architecture (environment)?" (CEP framework)
b) "What are usable algorithms to create a CEP engine?"

Sub-questions with question 2:

a) "What are key CEP pattern constructs?"
b) "Which of the existing engines satisfies these key pattern constructs the most?"
c) "What is the optimal engine?"

Sub-questions with question 3:

a) "What is innovation management and on which levels does it occur?"
b) "How can these levels be integrated to enhance innovation management within the company?"
c) "How is Cordys applying innovation management and how does this relate to the theory?"

1.3 Research details

In this paragraph a layout is given in which way the research is conducted. The research took place at Cordys. Their mission is design, develop and deliver technology and applications that allow their customers to rapidly design and enhance processes, integrate these processes with other processes, applications and data sets.

1.3.1 Phase 1: Literature / context study

In this phase the main task is studying literature and articles to get more general knowledge about the subject and concerning concepts. Also presentations and interviews with people that have special knowledge about this area are used in this process. At the end of this phase it can be decided where to focus on for further research. The end product of this phase will be a finished version of the research plan.

This phase acts as a sort of funnel; starting with a wide research area slowly narrowing this area down to a small focused area for conducting further research. Graphically this phase is presented in *Figure 1-1*.

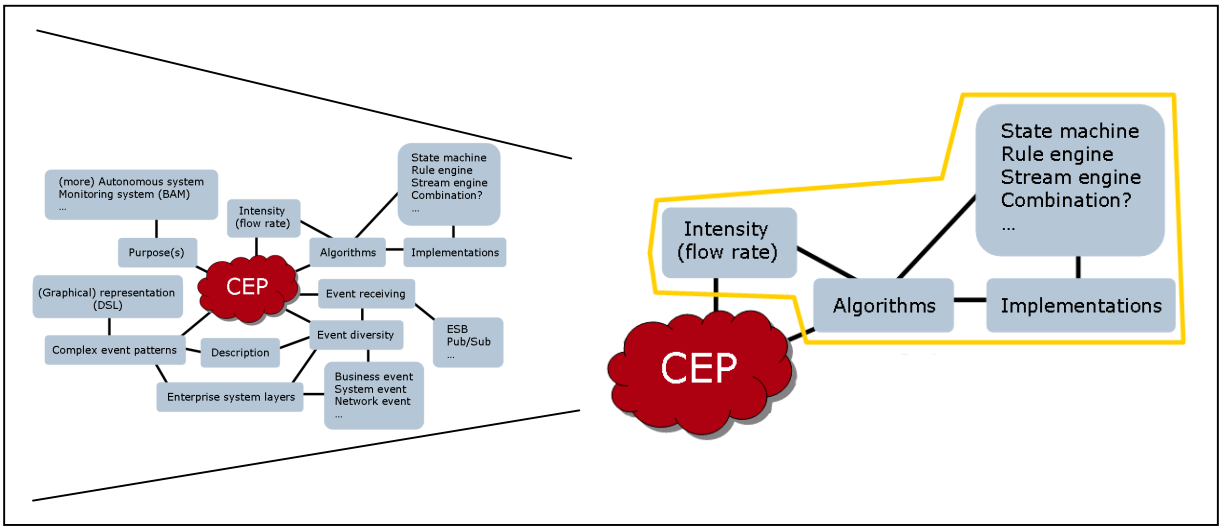


Figure 1-1: Focus in our research

At the left side of the funnel the CEP-cloud is shown. This is a cloud consisting of several key concepts surrounding CEP. More information about this CEP-cloud is given in chapter 2. At the right side of the funnel the focused area is shown. The focus is chosen according to the interests of Cordys.

1.3.2 Phase 2: Focused research

In this phase the main task is answering the research questions and thus conducting the main research. This will be divided in the following steps (in short):

1) Research what the most important aspects are when we want to use CEP within an enterprise architecture. The end product of this step will be a CEP framework (a description of the environment that is needed to incorporate CEP).

2) Research which constructs are the key ingredients for CEP engine patterns. These constructs or operators will be introduced in cases, facing real world problems. The end product of this step will be a description about the different constructs for CEP patterns and these will be introduced within several real world cases. These cases can be used to evaluate the different engines and to draw conclusions upon.

3) After the conclusion about which existing engine matches the criteria the best we go one step further, resulting in an overall conclusion about the optimal engine, where it is possible that a combination of existing engines with maybe some new ideas will be the optimal solution for Cordys. The end product of this step will be an overall conclusion concerning the optimal engine for including the power of CEP within the Cordys environment.

4) To address the last main question three different articles about innovation management will be reviewed and discussed. After that, possible overlap and gaps are discussed. To compare theory with practice a (small) interview will take place at Cordys, concerning their views on innovation management. These interviews will be used to make a comparison between theory and practice.

This phase can be divided into two parts: a *technology* part and a *management* part. The first two research questions concern the technology part, while the last question deals with the management part.

The second research phase is graphically presented in two research models, shown in *Figure 1-2: Technological research model* and *Figure 1-3: Management research model* (further referred to as the technological and management research model).

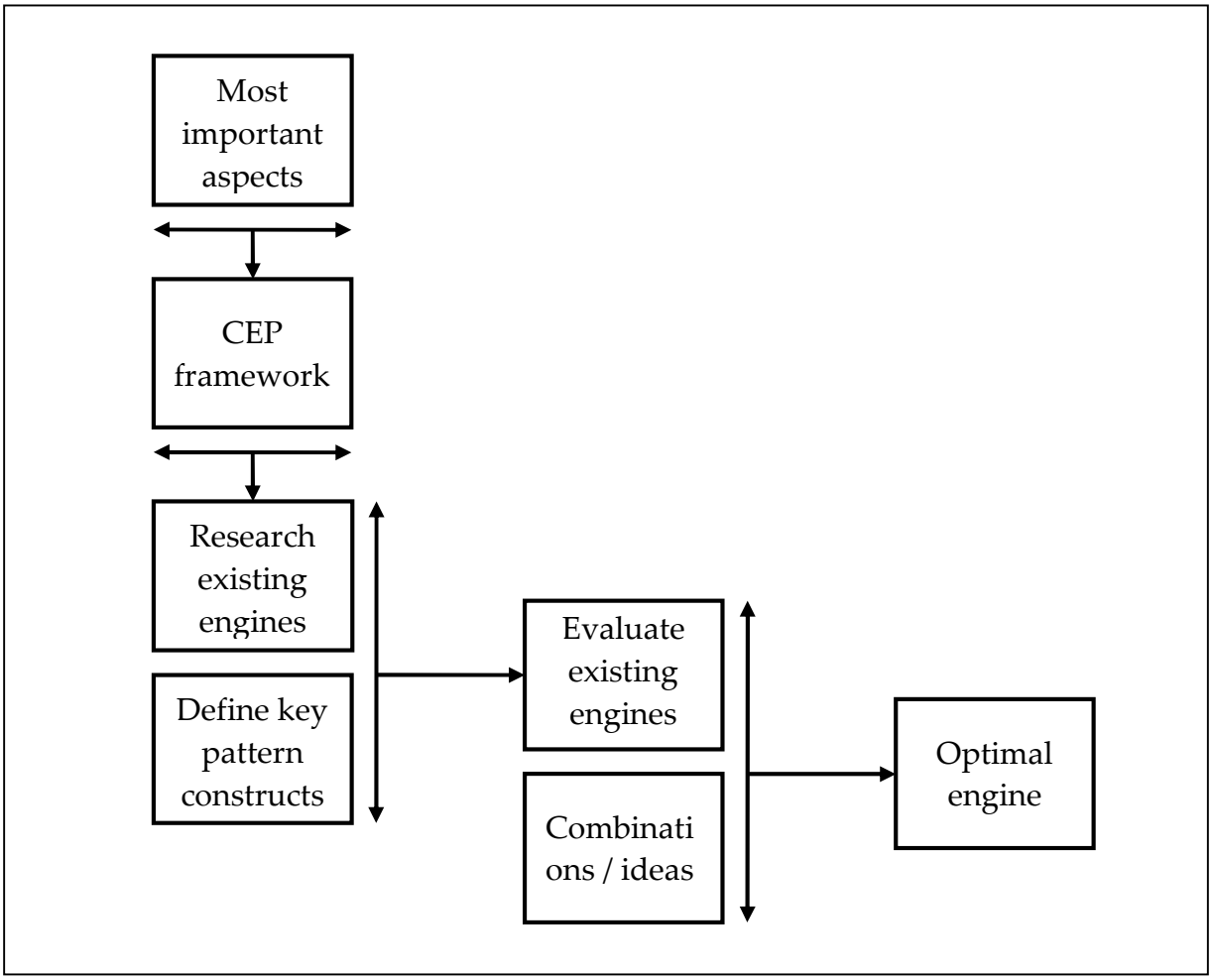


Figure 1-2: Technological research model

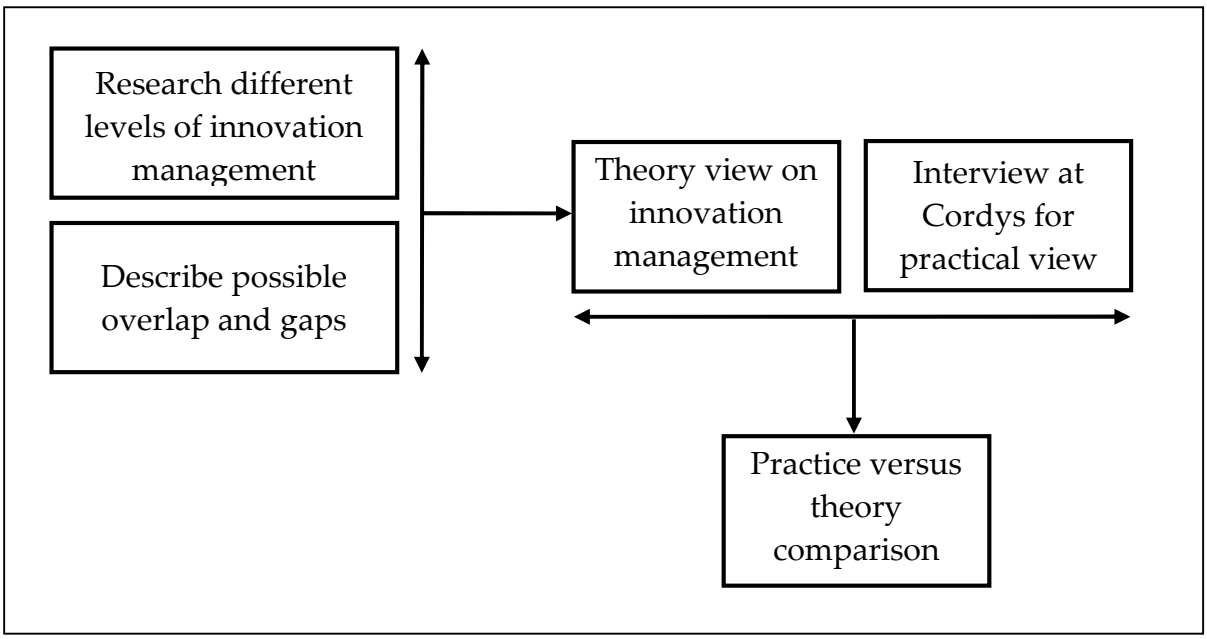


Figure 1-3: Management research model

1.4 Thesis structure

In this paragraph the structure of the thesis is explained. Chapter 2 introduces the context of CEP and leads to a better understanding of what CEP is. In chapter 3 the CEP framework is introduced, which provides a base for integrating a CEP engine within an enterprise IT system. The fourth chapter introduces a model that is used to compare different CEP engines. In chapter 5 the different CEP engines are introduced, which are tested for the ability to express the cases introduced in chapter 4. Also some (simple) performance tests are run. The results of the comparison of the different CEP engines are shown and discussed in chapter 6. In chapter 7 we state the conclusions. After the conclusions, chapter 8 deals with innovation management on three different levels.

2 Context

In this chapter the key concepts concerning CEP are discussed. In the "CEP-cloud" all different concepts that can be related to CEP are indicated and the most relevant concepts out of the CEP-cloud are introduced in the following paragraphs. In *Figure 2-1* we indicate which concepts are further introduced. The book "The Power of Events" from David Luckham [Luckham 2002] provided much inspiration for this thesis and helped a lot to get a better insight in what CEP is all about. The complex events website [ComplexEvents n.d.], also initiated by Luckham contributed to this insight too.

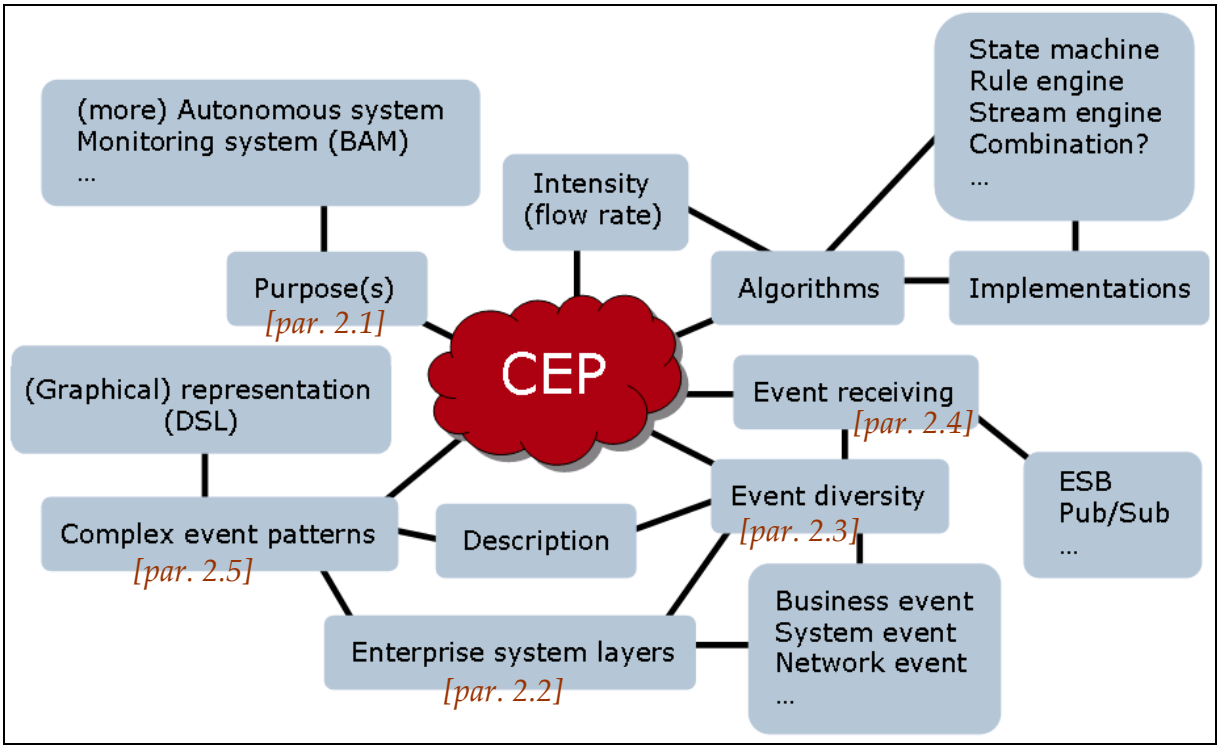


Figure 2-1: The CEP-cloud

2.1 Purposes

CEP can be used to serve many purposes. In this paragraph the most relevant purposes of CEP are introduced and briefly explained: Event-Driven Architectures, Enterprise Application Integration, Business Process Management, and Business Activity Monitoring.

2.1.1 Event-Driven Architecture

Event-Driven Architecture (EDA) is a software infrastructure that by nature is very loosely coupled. The main idea behind EDA is that a large software system consists of many small components that all have their own function. The communication between the components is done by using events. An event can be seen as a notification, which tells other components that a certain 'job' is done. Because events are very important within an Event-Driven Architecture also the handling and routing of events is very important. CEP is a very powerful addition to EDA, as it can detect complex situations in real-time. In a short summary about EDA, retrieved from [DataDirect n.d.] the need for CEP within EDA is also mentioned:

"…As enterprises embrace Event-Driven Architectures, the requirements of BAM / BPM initiatives dictate requirements for detecting complex patterns comprised of multiple events which may occur across an organizations event matrix. There is tremendous value in discovering and understanding these complex patterns in a real-time fashion, allowing critical business decisions to be made…"

A model of EDA components and an EDA example [Elemental 2006] are found in *Figure 2-2* and in *Figure 2-3*.

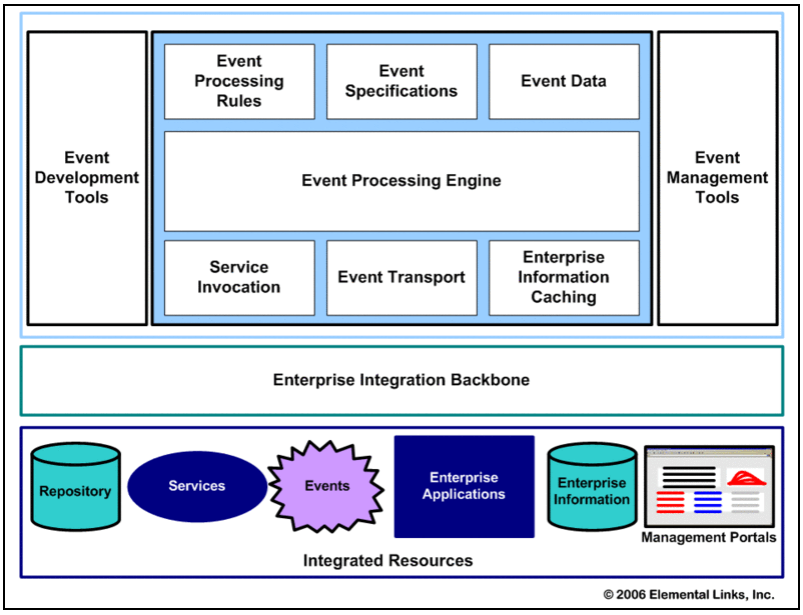


Figure 2-2: EDA components

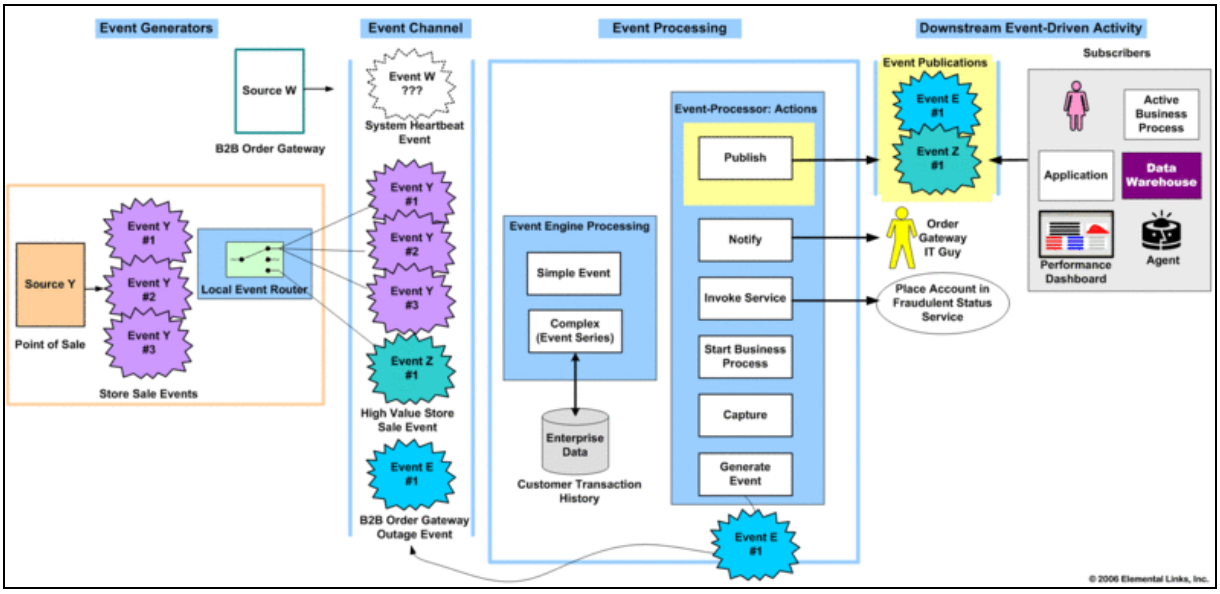


Figure 2-3: EDA example

2.1.2 Enterprise Application Integration

Today's enterprises already have many different types of applications, including: CRM (Customer Relationship Management), SCM (Supply Chain Management) and BI (Business intelligence) applications. Much information and knowledge is stored in these systems and a lot of money has been spent on them. Enterprise Application Integration (EAI) is a method to link these legacy applications and combine them with new applications. With EAI the data in different systems can also be kept consistent.

Many definitions of Enterprise Application Integration (EAI) are found. A global definition is retrieved from [SearchWebServices 2006]. SearchWebServices is a web services and SOA resource for enterprise IT professionals.

> *"EAI is a business computing term for the plans, methods, and tools aimed at modernizing, consolidating, and coordinating the computer applications in an enterprise.*

Definition 2-1: EAI definition

A model of EAI, retrieved from [SOAWorld 2004] is found in *Figure 2-4*.

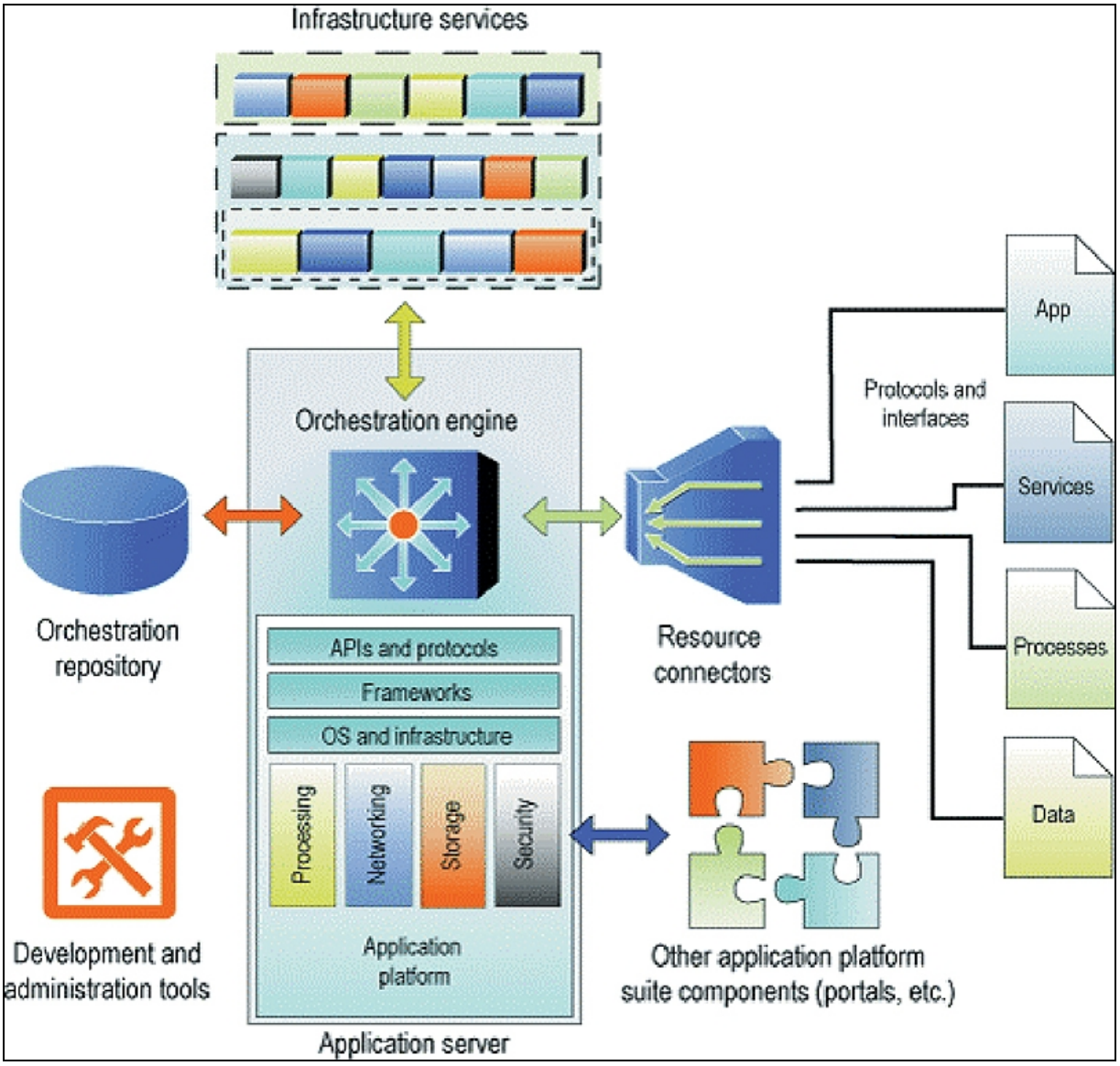


Figure 2-4: EAI model

2.1.3 Business Process Management

Business Process Management (BPM) is a concept that intersects the fields of management and Information Technology. BPM is all about business processes that, among others, consist of organizations, humans, and systems. BPM includes three activities: process design, execution, and monitoring. While the (business) management field provides the knowledge to design the business processes, the IT

field provides the technology to execute them. A tool for monitoring business processes is Business Activity Monitoring, more about that in paragraph 2.1.4.

A definition for BPM [Bitpipe n.d.] is shown in *Definition 2-2*.

> *BPM is a systematic approach to improving an organization's business processes. BPM activities seek to make business processes more effective, more efficient, and more capable of adapting to an ever-changing environment.*

Definition 2-2: BPM definition

Figure 2-5 [BEAS 2006] shows how business processes cut across organizational and system boundaries.

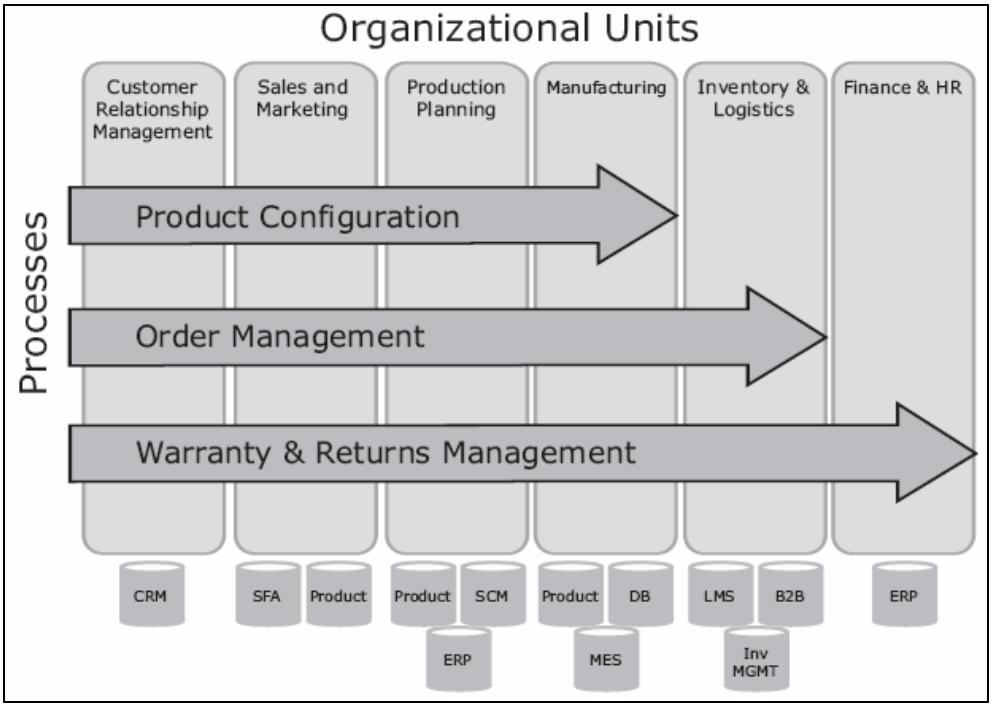


Figure 2-5: Business processes across product divisions and systems

2.1.4 Business Activity Monitoring

Business Activity Monitoring (BAM) is a supportive tool to give insight in the business performance and can help finding possible bottlenecks. BAM consists of three main steps: collecting data, processing data, and displaying the results. CEP is a very welcome addition to BAM, because it can detect complex situations that occur in

a large enterprise, and thus can help BAM report even more complex information. webMethods [webMethods 2006] also indicates that the newest BAM level includes 'complex pattern recognition'.

"…Second-level BAM strategies focused on Business Assurance and Visibility, Control Services and Complex Pattern Recognition blend the core strengths of BAM with these powerful analysis capabilities to provide an effective approach for targeting business problems in areas like compliance, change management, quality improvement, and operational business health, delivering more business value and reducing risk…"

SearchSMB [SearchSMB 2006] provides a global definition of BAM, found in *Definition 2-3*. In *Figure 2-6* a BAM example is shown.

> *"Business activity monitoring (BAM), also called business activity management, is the use of technology to proactively define and analyze critical opportunities and risks in an enterprise to maximize profitability and optimize efficiency.*

Definition 2-3: BAM definition

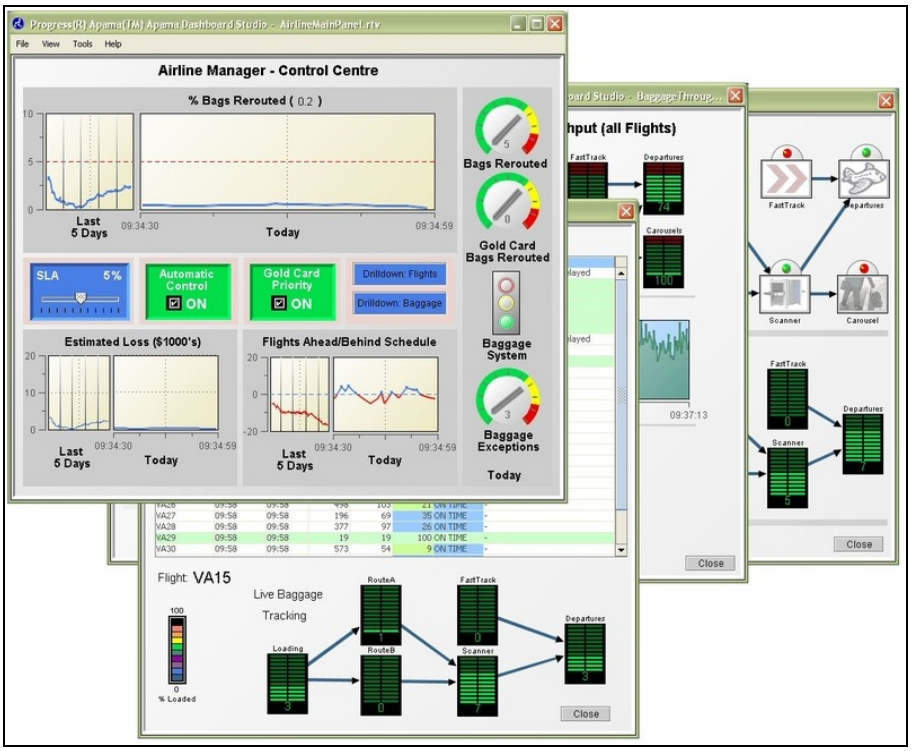


Figure 2-6: BAM application used in flight business

2.2 *Enterprise system layers*

A typical enterprise system consist of several layers. These layers provide a level of abstraction, making integration of different systems more easy. An example of enterprise system layers is found in *Figure 2-7*.

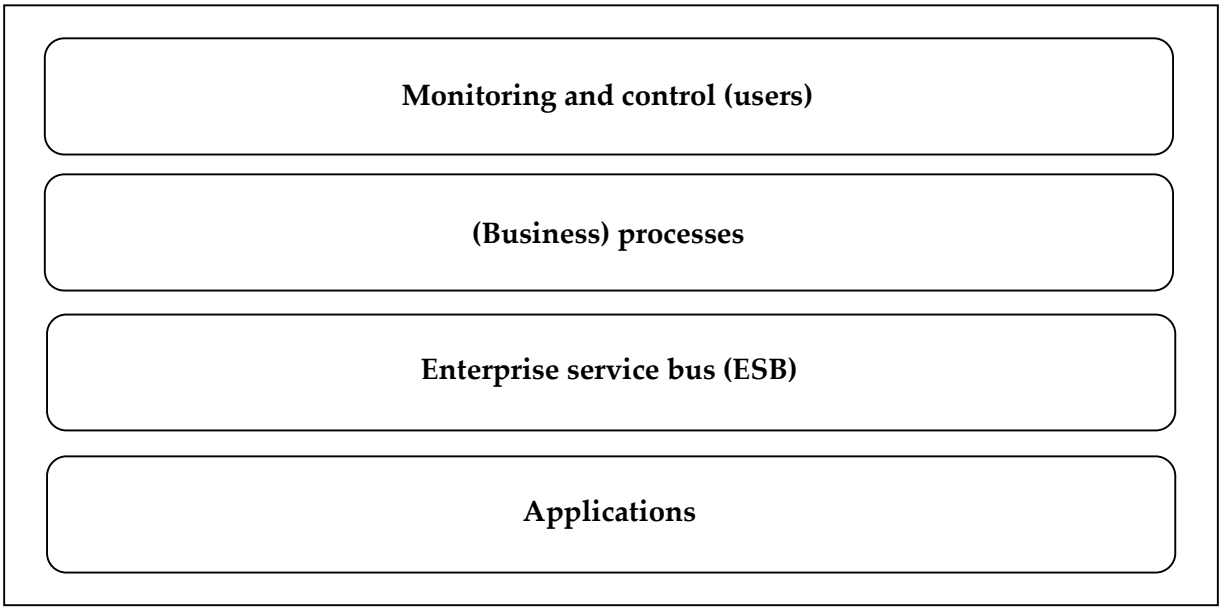


Figure 2-7: Typical enterprise system layers

The two top layers are the most abstract layers; monitoring and control provides user interfaces for the users to work with and gives them control over the system. In the (business) processes layer the different processes are defined. As explained in paragraph 2.1.3, these processes often cut across system and organizational boundaries. The ESB provides the connection to the different (often legacy) applications and makes integrating these applications possible.

2.3 *Event diversity*

In an enterprise IT system different types of events can occur. Ranging from lower level network events to higher level business events. Examples of lower level events are: networkTrafficHigh, routerDown, IPAddressUnknown, etcetera. Some examples

of higher level business events are: newCustomer, newOrder, orderCancelled, etcetera.

2.4 *Event receiving*

Events are the main ingredient for a CEP engine. It is of great importance that all events of interest are received by the CEP engine, otherwise defined situations will never be detected. The CEP engine can be connected to the enterprise IT system in different ways, depending on the type of connection used within the enterprise IT system. More details on connecting the CEP engine can be found in chapter 3.

2.5 *Complex event patterns*

Patterns are used to express situations that have to be detected. In a pattern, events are correlated with each other, with the use of operators. Because detecting patterns is the main function of a CEP engine, it is very important to know which operators are necessary and to have a uniform pattern notation. A definition of a uniform pattern notation is found in paragraph 4.3.

3 Framework

In this chapter the framework for a CEP engine is described. This framework provides a base for using a CEP engine within an enterprise IT system. Enterprise IT systems are event-driven systems that involve many different applications. These applications have to communicate with each other, but what has that to do with the CEP engine?

3.1 Topologies

Before looking into this we first go into the different ways in which communication is arranged. In literature there are several communication arrangements described (also known as (network) topologies). Before we proceed with the framework first the most common topologies are introduced:

- **Point-to-point**
  Point-to-point (mesh) topology is defined by single connections between the nodes. When all nodes are connected this is called a full or true mesh. Because of the many connections (for *n* nodes there are *n**(*n*-1) connections) the point-to-point topology is often referred to as a 'spaghetti'. Every connection has a connector which translates from an application domain to another application domain. For example when application A is connected to application B the side of the connector connected to application A translates 'A' to 'B', while the other side of the connector, connected to application B works the other way around, translating 'B' to 'A'. This example, together with an example of a point-to-point topology are combined in *Figure 3-1*.

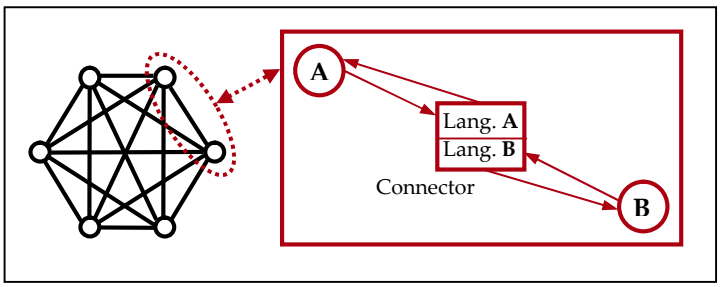


Figure 3-1: Point-to-point topology

- **Bus**

  The bus topology is defined by a bus connection that is used by all nodes to communicate with each other. The "language" that is "spoken" on the bus should be a universal language. The great advantage over a mesh network becomes immediately clear: there are only n connectors needed for n nodes. When a node is added, only one extra connector is needed to translate from the application domain to the bus domain and the other way around. When adding a note to a point-to-point connection there should be made a connector for every existing node in the mesh, so also adding nodes becomes more easy when a bus topology is used. In *Figure 3-2* a graphical representation of a bus is shown.

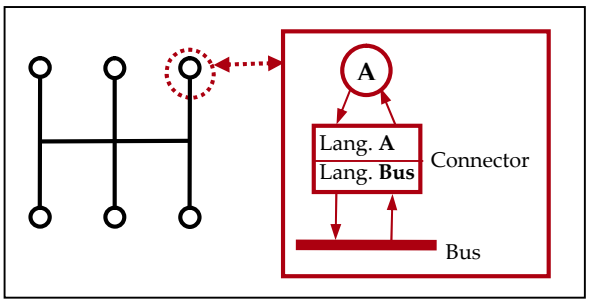


Figure 3-2: Bus topology

- **Star**

  The star topology is defined by a central node that is connected to all other nodes. When two nodes want to communicate this is always done via the central node. This topology is also known as "hub" and "spoke", where the central node acts like a hub and the other nodes are the spokes. There are two

possible connector types: the connector can translate the application domain of the node to a general domain (like the bus topology), or the connector consists of modules that are able to translate from any of the application domains of all the other nodes to the application domain of the node where the connector is connected to. To keep things simple the first variant is of course favourable, but in older systems this variant is maybe not yet used. An graphical representation of a star is shown in *Figure 3-3*.

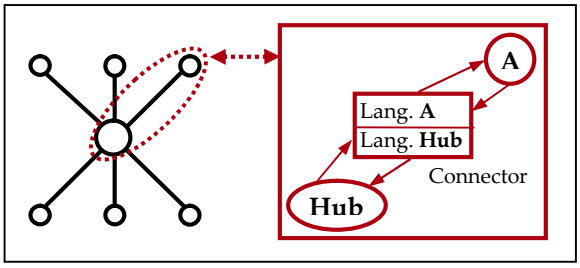


Figure 3-3: Star topology

3.2 Communication and CEP

At the start of this chapter the question "Why is communication within an enterprise IT system of any importance to a CEP engine?" arose. The answer to this question is simple: "A CEP engine uses events to reason upon". An example will clarify this: let's say that within a particular enterprise IT system the events A,B, C and D exist. These events are abstract but in reality they are synonym to for example the following events: newCustomer, newOrder, newTransaction, etcetera. The CEP engine can detect patterns of events, like "A and D" or "A followed by C", more on these event patterns is explained in chapter 4. If the CEP engine has to detect these patterns it becomes clear that it is of great importance that a CEP engine receives all events from the enterprise IT system that are used within these patterns. The events can be divided into the different layers that are typically used within an enterprise IT system, ranging from lower level network events to higher level business events. Examples of lower level events are: networkTrafficHigh, routerDown,

IPAddressUnknown, etcetera. In this thesis the focus is on business events, like: newCustomer, newOrder, orderCancelled, etcetera. While answering the question "Why is communication within an enterprise IT system of any importance to a CEP engine?" is simple, bringing it in practice can be quite hard. Several problems occur in different communication topologies. The CEP framework gives support when using a CEP engine within different communication topologies.

First, the basic preconditions that need to be satisfied to successfully implement a CEP engine within an enterprise IT system are described. After that, these preconditions will be dealt with for each topology.

The CEP engine should be able to:

- **receive all events of interest, system wide.**
- **send events to different nodes, system wide.**
- **understand the events from the different nodes.**

3.2.1 Point-to-point

In the point-to-point topology all nodes are directly connected to each other, as described in paragraph 3.1. When application A sends an event to application B this will be done through a direct connection between A and B. Other nodes do not receive this event. Here a problem arises: the CEP engine should get (a copy of) every event. How can this be achieved? A possible solution is to create communication lines between the CEP engine and all other nodes. In this way events can be send just as normal, only a copy arrives at the CEP engine. The second problem that has to be faced is the format of the events. In the point-to-point topology there is not one universal language that is "spoken". There are two solutions for this problem:

- The first solution is that for each node a new connector is created that translates from the application domain to the CEP engine domain. This means

that for every added node, connectors to all other nodes (including the CEP engine) have to be created. This is not very good for complexity, but that was already a problem with a point-to-point topology. A graphical reproduction of this solution is presented in *Figure 3-4*.

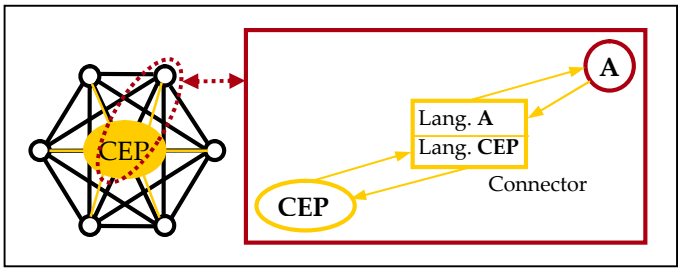


Figure 3-4: CEP point-to-point integration with new connectors

- The second solution is to extend the existing connectors, in the way that they also translate from the application domains to the CEP engine domain. The total amount of effort is about the same as with the first solution. For every application domain there should be a translate module from that application domain to the CEP engine domain. This module has to be added to all connectors of the concerning application connectors. The solution is graphically presented in *Figure 3-5*.

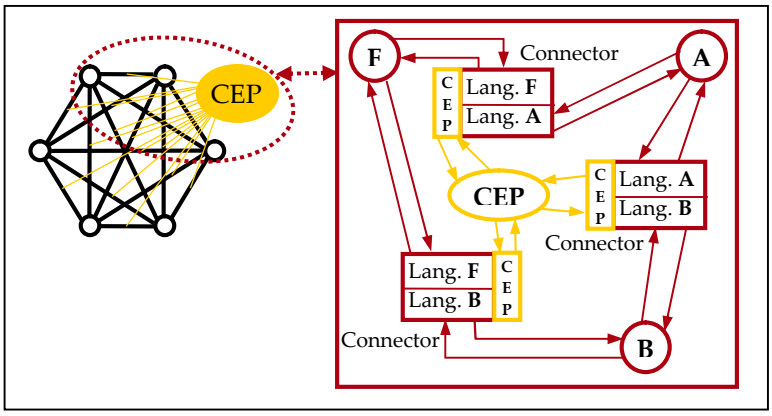


Figure 3-5: CEP point-to-point integration with existing connectors

3.2.2 Bus

The rather modern bus topology allows for a more easy integration of a CEP engine. This is due to the universal language that is used on the bus. The CEP engine only has to understand this language to be able to read every event and to generate new events.

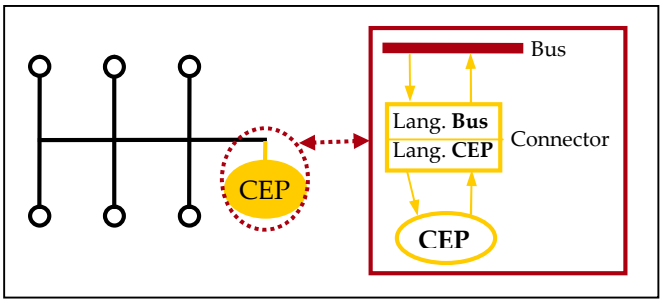


Figure 3-6: CEP bus integration

3.2.3 Star

In a star topology the easiest way to integrate a CEP engine is to connect the CEP engine to the central hub. The hub has to be altered in such a way that it sends (a copy of) all the events that he receives. In this way the CEP engine receives all events of importance, and with that the first precondition is met. In this framework only the "simple" variant of the star topology is dealt with. This variant is described in paragraph 3.1, and is recognized by the general language that is introduced at the hub. The problem when there is no general hub language is that, in order to let the CEP engine understand the events, a special connector has to be designed. This connector should be able to translate from every application domain in the enterprise IT system to the CEP engine domain and vice versa of course. With the special connector the CEP engine is able to understand the incoming events and can generate new events and send these to different applications in the enterprise IT system. In *Figure 3-7* is graphically shown how the CEP engine can be integrated with a star topology.

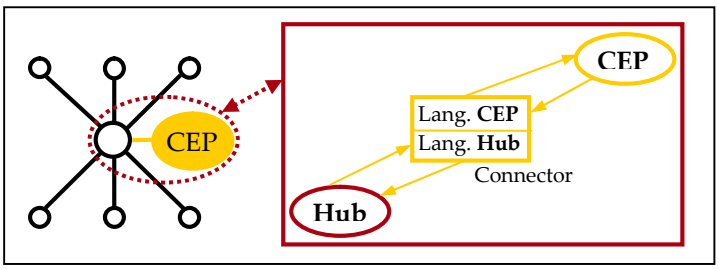


Figure 3-7: CEP star integration

3.3 Cordys SOA Grid

Cordys has an advanced communication topology named the Cordys SOA Grid. The unique feature of this SOA Grid is that it is fully based on a widely adopted standard: XML. The SOA Grid is presented as a kind of bus, but it actually works slightly different. The main difference is that the shared connection bus is used to locate a node in the system. After the location is known the two nodes start a direct communication line. The key factors of the Cordys SOA Grid are the *state registry* and the *location registry*. These two registries make sure that the Cordys SOA Grid maintains its high availability and that the state and location of the different nodes are available. A great advantage of this way of communicating is that the traffic over the shared part is limited. However, it makes integrating the CEP engine a little bit harder, because receiving all events of interest can be somewhat of a hazard. In *Figure 3-8* a graphical representation of the Cordys SOA Grid is presented.

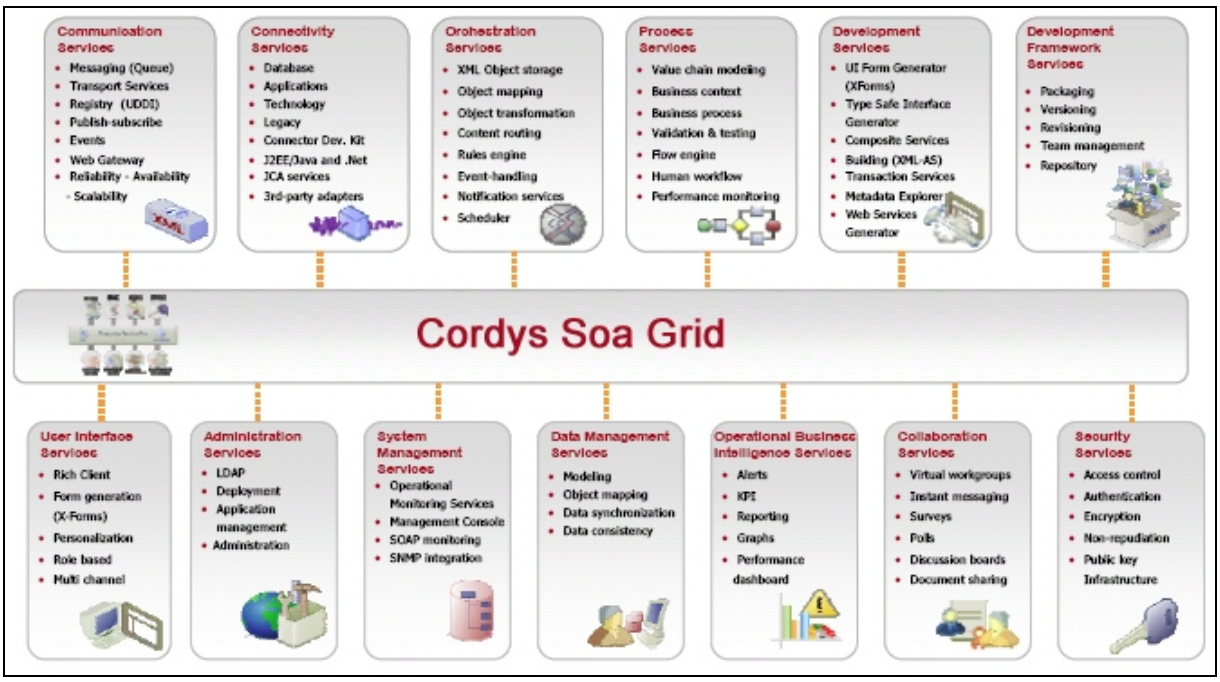


Figure 3-8: Cordys SOA Grid

A possibility to integrate the CEP engine with Cordys is to use the publish / subscribe methods. All nodes that have to be connected to the CEP engine can subscribe to the CEP engine. In this way the Cordys SOA Grid will be optimally used to support CEP. A graphical representation of the CEP engine integrated with the Cordys SOA Grid is found in *Figure 3-9*.

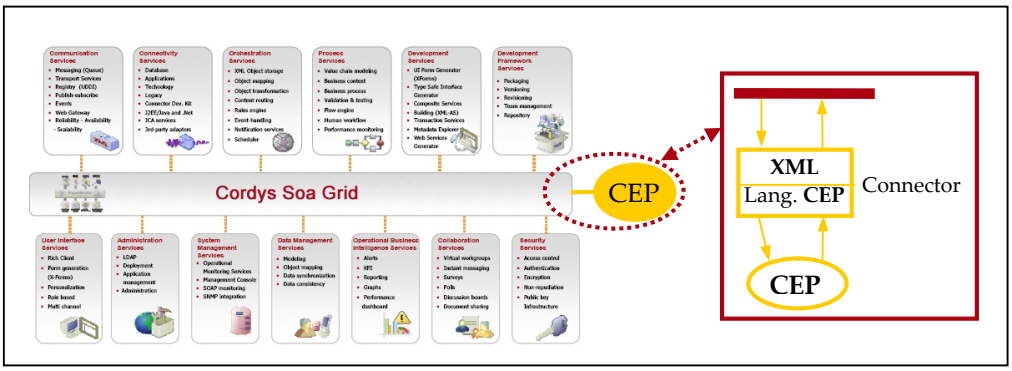


Figure 3-9: CEP Cordys integration

4 Model

In this chapter the model that is used for researching and comparing different CEP engines is described.

4.1 Streaming vs. non-streaming

There are two major streams within the area of event processing, namely CEP and ESP (Event Stream Processing). ESP also contains complex event processing features, but focuses mainly on processing large, dense data streams. A much used ESP example is processing streams from different stock markets. Because the ESP engines have to handle up to millions of events per second the key factor of ESP is high speed. The "normal" CEP is more about non-streaming events, like business events that are traveling around through an enterprise IT system.

4.2 CEP engine characteristics

In this research different CEP engines will be reviewed on the abilities of "normal" (non-streaming) CEP with the focus being on business events. The CEP engine should not be seen as a "black box" that does its thing and gives output. CEP engines consist of many linked components that all have a small function in the total CEP solution. Possible functions are: event extraction, event sampling, event filtering, event parsing, semantic matching, structure transformation, event enrichment, content based routing, *event aggregation / composition / correlation*, event splitting, event generation, event storing, action triggering, etcetera. In this thesis the focus is on event correlation. This is a very important function of a CEP engine because this is the place where patterns are defined and handled, which is the basic concept of CEP.

4.3 Six CEP cases

To review the different CEP engines it is crucial to know what important criteria for CEP engines are. The recognition of patterns, including the composition of the

patterns themselves is the main criterion for comparing the engines. A second criterion is performance, more about this in paragraph 4.4. In this paragraph the main criterion for comparing the engines is described. The result of the comparison is found in chapter 6.

To describe situations that need to be detected, the pattern language should introduce the needed key constructs to do that. These constructs define the relationship between different events that are used in a pattern (event aggregation / composition / correlation). In the six cases in this paragraph the key CEP pattern constructs are introduced. These cases consist of two parts, where the first part is an example of a real life problem that has to be solved. In the second part a CEP solution is described that is able to solve the problem. The CEP solution is again divided into two parts; in the first part necessary events are defined and in the second part a CEP pattern is introduced that will detect the "problem situation". The cases will start with simple 'complex' event patterns and will gradually become more complex. They also they give a good view on what is possible with CEP.

In the cases a general CEP language is used that is inspired by the CEP language of Esper [Esper n.d.]. The definition of this language is found in *Definition 4-1*.

Notation:

- *C = Set of all events.*
- *V = value*
- *X_i, Y_i = Event with order number i (= X, Y if max i = 1 and X_i, Y_i Î C).*
- *X_i(a,b,…) = a,b,… are attributes of event X_i.*
- *X_i(where a=V) = Attribute a is matched with value V.*
- *X_i(where a= Y_i.a) = Attribute a is matched with attribute a from event Y_i.*
- *T = time interval, expressed in seconds, minutes, hours, days, weeks, months or years.*
- *Z = expression that is build with elements from this general CEP language.*
- *Events are written in lower camel case; the name starts with lower case and each new word starts with a capital: "newOrder".*

Operators:

Operators are divided into three classes:

- *Logical operators: "and", "or" and "not".*
- *Time operator: "within T (Z)".*
- *Sequence operator: "->".*

Example expressions:

- *Z = "X and Y".*
- *"within 40 seconds (Z)".*
- *"A -> B" (event B has to arrive after A).*

Definition 4-1: General CEP language definition

In *Table 4-1* the introduction of the different operators in the six cases is shown.

	Logical operators		Time operator	Sequence operator
	and / or	not	within	->
Case 1	●			
Case 2	●	●		
Case 3	●		●	
Case 4			●	●
Case 5	●	●	●	
Case 6	●	●	●	●

Table 4-1: Introduction of operators in the cases

4.3.1 Case 1: "Simple use of logical operators"

Logical operators are widely used within different approaches of describing CEP patterns, and that's logical: these logical operators are very intuitive when events have to be aggregated and relations between them have to be expressed.

Imagine a sales company that has its own warehouse. When a customer orders a product, the order is send to the package department. Here the product will be packaged, ready for transport to the customer. After packaging the order will go to the post department that puts the address on the package and delivers the package at the post-office. When a customer cancels an order and the order is already packaged then the order normally will be unpacked and put back at the stock in the warehouse. But what if another customer just placed the same order? Then the first order should just get the address of the last customer. When a customer cancels an order this can be seen as an event, also the fact that a product is packaged can be seen as an event. This way of using events in real time can provide extra efficiency within a company.

First the three events that are used in this case are defined:

- "newOrder" event. This event is raised when a customer places an order. The attributes of this event are "customerID", "orderID" and "productID". In this simple example a customer can only choose one product per order.

- "orderPackaged" event. This event is raised when an order is packaged and thus heading to the post department for addressing and sending. The attributes of this event are "orderID" and "productID".

- "orderCancelled" event. This event is raised when an order is cancelled by the customer. The attributes of this event are "customerID" and "orderID".

Now a pattern can be defined that will detect the situation where customer A cancels an already packaged order and customer B orders the same product. The following pattern will detect this situation:

"orderPackaged and orderCancelled(where orderID=orderPackaged.orderID) and newOrder(where productID=orderPackaged.productID)"

A brief explanation of this pattern: the first part of the pattern "orderPackaged and orderCancelled(where orderID=orderPackaged.orderID)" detects when both - an orderPackaged and an orderCancelled - events take place. The part between brackets "(where orderID=orderPackaged.orderID)" makes sure that both events are about the same order. The last part of the pattern "and newOrder(where productID=orderPackaged.productID)" detects that a customer places an new order with the same product as the just cancelled order that was already packaged. The pattern only fires (becomes true) when both parts of the pattern become true.
When the where statements between brackets are omitted (the pattern then would be "(orderPackaged and orderCancelled) and newOrder") the pattern also would become true when for example the order with a product "chair" is packaged and an order from another customer is cancelled and again another customer places a new order that contains a "table". In some detection situations the 'where statements' will not be necessary, but it's clear that in this case they are needed to make any sense of the detected situation.

4.3.2 Case 2: "Why the not operator is so important"

With a not operator it is possible to detect if some event did not happen. At first it's not directly clear what the added value of a not operator would be. But the next example will clarify this and shows that not having a not operator, is not a good idea.

Let's say that the sales company in the previous case wants that expensive orders (for example orders that are priced higher than 5.000 euros) are first verified by a sales manager. When everything is okay, the sales manager gives "green light" to the order and the order is sent to the package department. But what if there's an "orderPackaged" event of an order that was 5.500 euros, but there has never been an "orderApproved" event from the same order? In that case an expensive order is processed without approval. In this example CEP is used to detect a "faulty" situation.

First the two events that are used in this case are defined:

- "orderPackaged" event. The description of this event is the same as for case 1, only the attribute "totalPrice" is added.

- "orderApproved" event. This event is raised when a sales manager approves an expensive order. The only attribute of this event is "orderID".

Now a pattern can be defined that will detect the situation where expense order X is packaged without an approval from a sales manager. The following pattern will detect this situation:

"orderPackaged(totalPrice>5000) and not orderApproved(where orderID=orderPackaged.orderID)"

A brief explanation of this pattern: "orderPackaged(totalPrice>5000)" filters out expensive packaged orders from all packaged orders. "and not orderApproved(where orderID=orderPackaged.orderID)" looks if there is no order

approval for an expensive order. For every expensive packaged order there will be a check if there was an order approval. If this is not the case the pattern will become true and fires.

4.3.3 Case 3: "No CEP without time"

Time is a very important aspect within CEP. Many patterns have a time aspect to express a situation. In this case an example is provided where time is an essential aspect.

In this case we are helping a bank dealing with fraud. When someone draws money with a bankcard from an ATM and within a short time again money is drawn from another ATM in a different city then this should be investigated. For drawing money from an ATM a bankcard is necessary, so if there are two withdrawals within specified time limit in two different cities it looks like bankcard fraud.

First the event that is used in this case is defined:

- "cashWithdrawal" event. This event is raised when someone withdraws cash from an ATM. The attributes of this event are "bankcardNumber", "accountNumber", "cityID".

Now a pattern can be defined that will detect the situation where two withdrawals in two different cities within a specified time interval take place. The following pattern will detect this situation:

"within 120 seconds (cashWithdrawal1 and cashWithdrawal2 (where bankcardNumber=cashWithdrawal1.bankcardNumber, accountNumber=cashWithdrawal1.accountNumber, cityID!=cashWithdrawal1.cityID))"

A brief explanation of this pattern: "within 120 seconds (…)" does exactly what is expected: the part between brackets should become true within the time limit of 120 seconds. Now let's look at the part between brackets: "cashWithdrawal1 and cashWithdrawal2 (where bankcardNumber=cashWithdrawal1.bankcardNumber, accountNumber=cashWithdrawal1.accountNumber, cityID!=cashWithdrawal1. cityID))". This part 'looks' for two cash withdrawals from one bank account with one bankcard but in two different cities.

4.3.4 Case 4: "The sequence of events"

Next to the time aspect also the sequence aspect of events is very important. In many situations it is important that events happen in a specified sequence, in this case the sequence operator is introduced.

For the next example we stay in the financial business, this time in the stock market business. Sometimes it happens that a company or individual sells a fairly large amount of stocks just before the price of that particular stock takes a deep fall. This is known as insider trading and is of course illegal. In this problem, as in many problems, the time aspect is also of importance. The insider trading story holds only when the selling of the stock happens just before the price crash. Where 'just' should be read as a couple of days.

First the two events that are used in this case are defined:

- "sellStock" event. This event is raised when stocks are sold. The attributes of this event are "stockID", "sellerID" and "amount".

- "stockPriceChange" event. This event is raised when the price of a stock is changed. The attributes of this event are "stockID", "newPrice" and "priceChangePercentage".

Now a pattern can be defined that will detect the insider trading situation:

"within 7 days (sellStock (amount>10000) -> stockPriceChange (where stockID=sellStock.stockID, priceChangePercentage< -20))"

A brief explanation of this pattern: "within 7 days (…)"does exactly what is expected: the part between brackets should become true within the time limit of 7 days. Now let's look at the part between brackets: "sellStock (amount>10000) -> stockPriceChange (where stockID=sellStock.stockID, priceChangePercentage< -20)". "sellStock (amount>10000)" selects sellStock events that describe a large amount of stocks sold. "-> stockPriceChange (where stockID=sellStock.stockID, priceChangePercentage< -20)" selects a stockPriceChange event that is about the same stock as the large sellStock event. "priceChangePercentage< -20" makes sure that only big price falls are detected. The sequence operator ("->") makes sure that first the sellStock event comes in and after that the stockPriceChange event. The order of the events is critical to detect an insider trading situation.

4.3.5 Case 5: "Not in stock"

In this case the not operator and time operator are combined. This is a common combination because it is able to check if something did not happen within a certain amount of time.

Let's say that the sales company in the first two cases wants that customers that place orders are notified when a product is not in stock and thus there order is not shipped

yet. The policy of this sales company is that placed orders should be packaged within three hours after the order is placed. If the order is not packaged within this time limit then the product is not in stock and the customer should be notified of this.

First the two events that are used in this case are defined:

- "newOrder" event. This event is raised when a customer places an order. The attributes of this event are "customerID", "orderID" and "productID". In this simple example a customer can only choose one product per order.

- "orderPackaged" event. This event is raised when an order is packaged and thus heading to the post department for addressing and sending. The attributes of this event are "orderID" and "productID".

Now a pattern can be defined that will detect the situation where an ordered product is not in stock:

"newOrder and not within 3 hours (orderPackaged (where orderID=newOrder.orderID))"

A brief explanation of this pattern: "newOrder and not within 3 hours (orderPackaged (where orderID=newOrder.orderID))" checks that within 3 hours the orderPackage event of an order does not come. If this is the case then the pattern will become true.

4.3.6 Case 6: "Combination of different operators: detect possible unsatisfied customer"

In this last case the different operators are combined into one pattern to show some real power of what CEP can add to enterprise IT systems.

Again the sales company in the first two cases is used, but for this case the service department is added. This department is responsible for helping customers when they have questions or problems. In this case the situation where a customer contacts the service department twice within two weeks and after the calls this customers suddenly stops placing new orders, is detected. An extra condition of this case is that the customer in question is a regularly ordering customer. Of course this 'order stop' can be a coincidence but it is also possible that the customer is not satisfied by the support that is delivered by the service department and therefore chose not to place new orders anymore. In this case an unsatisfied customer is detected and maybe the problems that were not solved to full satisfaction of the customer can be solved in another way. This is the right moment to contact this customer and see if there is a way to solve the problem and with that try to win over the customer again, before it is too late and this customer is lost to a rival company.

First the two events that are used in this case are defined:

- "newOrder" event. This event is raised when a customer places an order. The attributes of this event are "customerID", "orderID" and "productID". In this simple example a customer can only choose one product per order.

- "serviceCalled" event. This event is raised when a customer makes a call to the service department. The attribute of this event is "customerID".

Now a pattern can be defined that will detect the situation of a possible unsatisfied customer:

"within 2 weeks (serviceCalled1 and serviceCalled2 (where customerID = serviceCalled1.customerID)) -> not within 2 weeks (newOrder (where customerID = serviceCalled1.customerID))"

A brief explanation of this pattern: "within 2 weeks (serviceCalled1 and serviceCalled2 (where customerID = serviceCalled1.customerID))" checks for the situation where the same customer calls the service department twice within two weeks. "-> not within 2 weeks (newOrder (where customerID = serviceCalled1.customerID))" checks for the situation where the same customer did not place new orders anymore. The sequence operator ("->") makes sure that the check for no new orders is done after the check for the two calls to the service department.

4.4 Performance tests

The six CEP cases, described in paragraph 4.3 are the main research factor, but to make the results more complete also some (simple) performance tests cannot be left out. In this paragraph the performance tests are defined, results are presented and discussed in paragraph 6.2.

Of the three researched engines, Esper, StreamCruncher and ruleCore, only the first two could be tested for performance. This is because ruleCore is only available for testing on the servers of MS Analog Software kb, the company behind ruleCore. In that case the results will not be comparable because the tests are not executed on the same machine. Performance tests are executed on a Dell Latitude D600 notebook, equipped with an Intel Pentium-M 1.6 GHz processor and 1 GB main memory. Windows XP Professional SP2 is used as operating system. Both Esper and StreamCruncher use Java, the installed Java version is 1.6.0_02. For the tests version 1.10.0 (19-07-2007) of Esper, and version 2.2 Beta (19-08-2007) of StreamCruncher are used.

The performance tests are divided into two parts: the latency test and the throughput test, which will be described in the two following paragraphs.

4.4.1 Latency test

The latency test measures the total time that is needed to completely detect a pattern, from sending the first event to the engine until a callback is received when the pattern fires. Initialization time is not taken into account with test as this is not a factor of the latency. First a simple pattern is set, after that events are raised causing the pattern to fire and the engine to do a call back. In the first run all the events needed for the pattern to fire are raised, without any other events that are not used or are out of order. This first run is so to speak free of any noise. To see how the engine handles noise the test is repeated some times with increasing noise.

A function that describes the results of this test is:

$$\textbf{Latency } (X^n, N^m) = T_{x1} + T_{x2} + \ldots + T_{xn} + T_{n1} + T_{n2} + \ldots + T_{nm} + T_{callback}$$

Where X^n are the events needed for the pattern to fire, sent in the right order. N^m stands for the noise events. This can be events out of order or events that are not correlated to the pattern. The result of this function is the total sum of time needed to send all the (noise) events and added to that the time needed to do the callback when the pattern is detected.

Test plan

The test plan of this test is:

- Set the pattern "A and B (where ID = A.ID) -> C (where ID = A.ID)"

- First test is without noise: just send A(1), B(1), C(1) and measure the total time from sending the A event until the engine gives a callback after the pattern is detected.
- Now rerun this test a couple of times with increasing noise and again measure the total time: send A(1), *Y times B(2)*, B(1), *Y times C(2)*, C(1) (where Y = 50, 100, 200, 500, and 2000).

4.4.2 Throughput test

The throughput test measures how the engine is performing in more stressed situations. In this test the engine gets flooded with events and there is more than one pattern registered for the engine to detect. The two variables in this test are: the total number of events send and the event-noise ratio. The total number of events and the event-noise ratio will both increase, but first there is a series of test runs that do not involve noise.

Each test the following factors are measured: total time to complete test, total number of recognized patterns and the memory usage of the engine.

A function that describes the results of this test is:

$$\textbf{Throughput } (X^n, N^m) = T_{x1} + T_{x2} + \ldots + T_{xn} + T_{n1} + T_{n2} + \ldots + T_{nm} + T_{callback}$$

Where X^n are the events needed for the patterns to fire, sent in the right order. N^m stands for the noise events. This can be events out of order or events that are not correlated to the pattern. The result of this function is the total sum of time needed to send all the (noise) events and added to that the time needed to do the callback when the pattern is detected.

Test plan

The test plan of this test is:

- Set three patterns that have to be detected:
  - "A and B (where ID = A.ID) -> C (where ID = A.ID)"
  - "C -> D (where ID = C.ID)"
  - "A and C (where ID = A.ID) and D (where ID = A.ID)"
- First a series of tests without noise are run: just send X times (A(x), B(x), C(x), D(x)), where X = 2, 500, 1.000, 2.000, 5.000, 10.000, 50.000, 200.000, and 500.000, and measure the total time from sending the first An event until the engine gives the callback after the last pattern is detected. The three patterns should be detected all once per X: total patterns that have to be recognized: 3 * X.
- Now a second series of tests, with increasing noise, are run. The events are sent: X times (A(x), *Y times B(-1)*, B(x), C(x), *Y times C(-1)*, D(x)). X and Y are both increasing: (X = 2, Y = 1), (X = 500, Y = 2), (X = 1.000, Y = 3), (X = 2.000, Y = 4), (X = 5.000, Y = 5), (X = 10.000, Y = 6), and (X = 50.000, Y = 7).

5 Engines

In this chapter the three researched CEP engines are introduced.

5.1 Esper

Esper [Esper n.d.] is an open source engine that combines both ESP and CEP capabilities. Esper is available as Java source and C# .Net source (NEsper). In October 2004 the Esper open source project started, delivering the first alpha version (0.7.0) of Esper in January 2006. Now, more than one and a half years later a mature version of the product is available (1.10.0).

The Esper CEP engine is build with the use of state machine technology, while the ESP engine works with delta-flow networks. State machines are a very intuitive choice when building a CEP engine due to the fact that a CEP pattern is easy to express in a state machine. Let's consider the pattern "A -> (B and C) -> D", this pattern in state machine notation is presented in *Figure 5-1*.

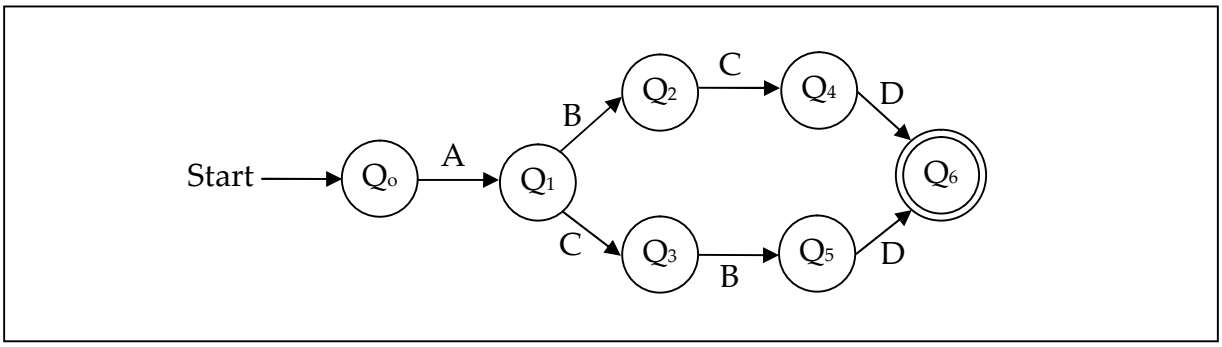


Figure 5-1: Using a state machine to express a pattern

To arrange extra support, training and licensing for commercial use EsperTech Inc. is founded. Two figures from their website [EsperTech n.d.] visualize the design and capabilities of the Esper ESP/CEP engine. *Figure 5-2* shows how Esper combines ESP and CEP. Their Event Query Language (EQL) is used to express filtering, aggregation, and joins, possibly over multiple event streams, while the pattern

language is used to define more complex patterns on different types of events. In *Figure 5-3* an overview of the Esper engine is presented.

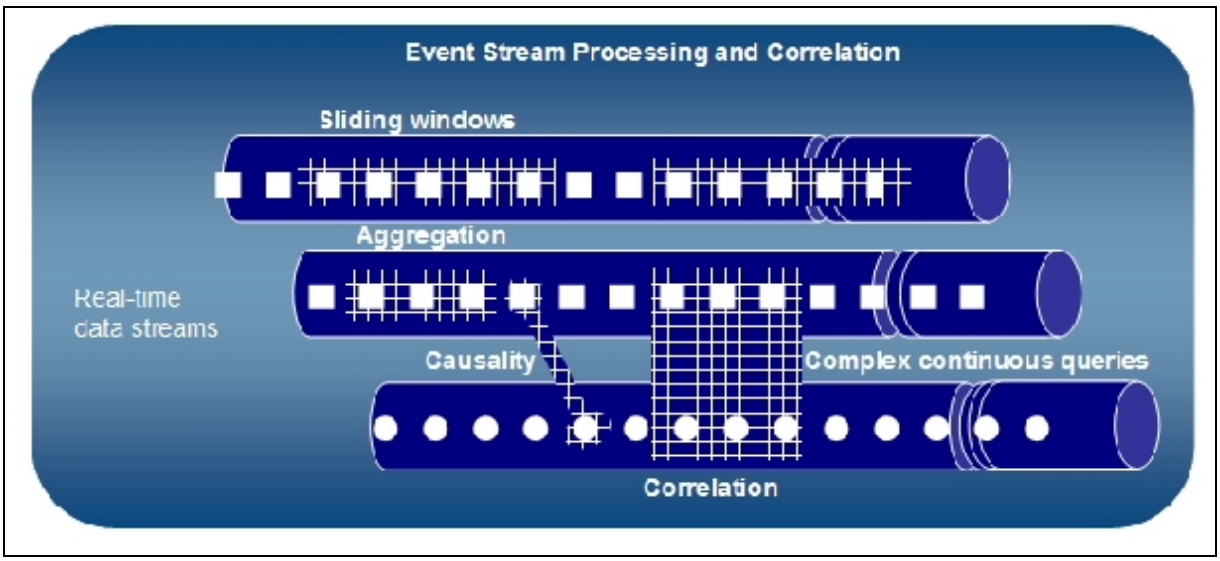


Figure 5-2: Esper combines ESP and CEP

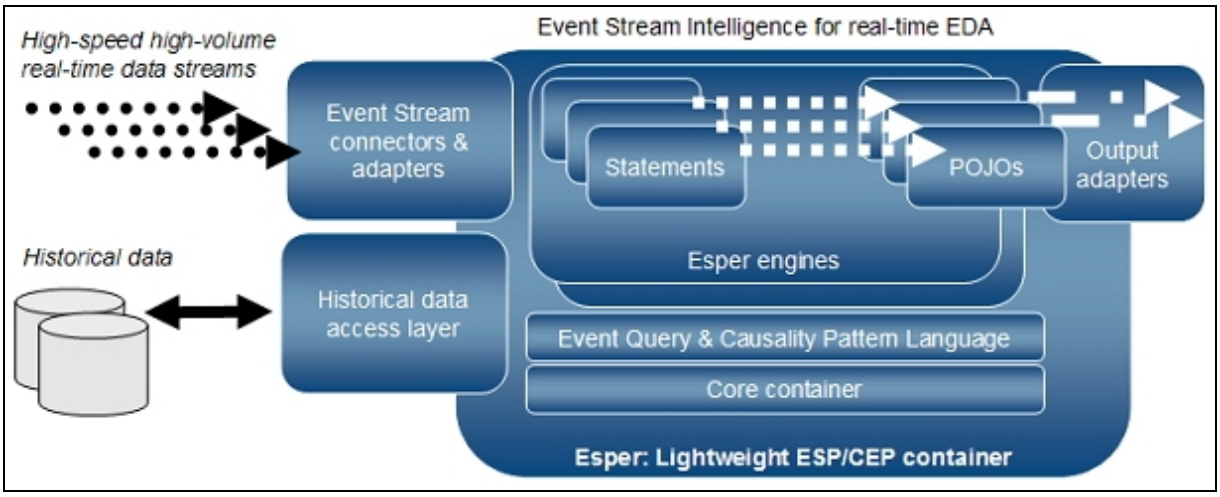


Figure 5-3: Esper engine overview

Esper also includes a historical data access layer to connect to the most popular databases, also combining historical data and real time data in one single query is possible. Esper can be easily integrated with most available servers, ranging from JEE server (Weblogic, Websphere, JBoss, Tomcat, etcetera), service busses, grid platforms, and Microsoft based .Net technologies. Esper supports different kind of event formats, from Java / .Net objects and maps to XML documents. In Q3 2007 Esper will

release an extension for their engine to include failover and recovery capabilities, ensuring that the engine is non-stop usable (high-availability). Also an extension to add event storage options will be released in Q3 2007. With this extension persistence of events becomes fully customizable.

As performance tests show Esper scales vertically nearly linearly (adding more CPU power). In a VWAP (Volume Weighted Average) benchmark [Esper benchmark n.d.] Esper exceeded 500.000 events per second on a dual CPU server class hardware, with only 5 microsecond average latency. Horizontally scaling is best handled by logical partitioning of statements and data streams to separate Esper instances.

5.2 StreamCruncher

StreamCruncher [StreamCruncher n.d.] is a closed source, free to use ESP engine. The program is created and maintained by one person, Ashwin Jayaprakash. Ashwin's interest in event processing started when he worked for BEA Systems, where he worked on their "event generators". Towards the end of 2005 he started noticing several ESP/CEP products and his interest in event processing emerged again. StreamCruncher has been developed by Ashwin in his personal time. The current version being 2.2 Beta, which was released in August 2007.

StreamCruncher, being an ESP engine, works with streams of events. Also correlation between streams is possible, state machines are used for this functionality. Next to correlation between different streams, StreamCruncher also incorporates event filtering, partitioning and aggregation. To help vertical scalability StreamCruncher uses a configuration file to extensively configure kernel performance. Also the kernel uses SEDA (Staged Event-Driven Architecture) technique to pipeline various processing stages that are processed by thread pools. StreamCruncher scales well on multi-CPU hardware. Horizontal scalability is not possible: StreamCruncher does not support a cluster of servers.

The user has a choice of using different databases, either in-memory or persistent databases. So, it is possible to combine real time data streams with historical data. Queries can access/lookup/join database data transparently with streams. StreamCruncher is not high-available and non-stop usable, also storing of events can only be done manually.

StreamCruncher is a pure Java kernel, ensuring easy integration with most existing applications, and it is free, giving users (even commercial ones) the opportunity to get accustomed to the world of event processing.

5.3 ruleCore Server

The CEP solution ruleCore Server [ruleCore n.d.], from MS Analog Software kb originates from the (business) rule world. The ruleCore engine is based on a graph based algorithm. This algorithm has its roots in the active database research. An interesting paper on this subject is "Snoop: An Expressive Event Specification Language For Active Databases" [Snoop 1993], written by Sharma Chakravarthy and Deepak Mishra. Some CEP concepts are addressed in this paper, long before ESP or CEP were known, so maybe this paper was ahead of its time. Chakravarthy and Mishra for example made a distinction between primitive events and composite/complex events: "primitive events: events that are pre-defined in the system, composite or complex events: events that are formed by applying a set of operators to primitive and composite events".

The ruleCore engine algorithm is presented as a tree, where the operators represent the nodes and events the leafs. Each node becomes true as the operator in it becomes true. Depending on the type of operator, for example an "and" operator becomes true when both its leafs become true, but for the "or" operator only one of its leafs has to become true to let the or operator become true. When the root node becomes true the defined situation is detected. To clarify this tree approach the pattern "A and B and C or D" is presented in tree notation in *Figure 5-4*.

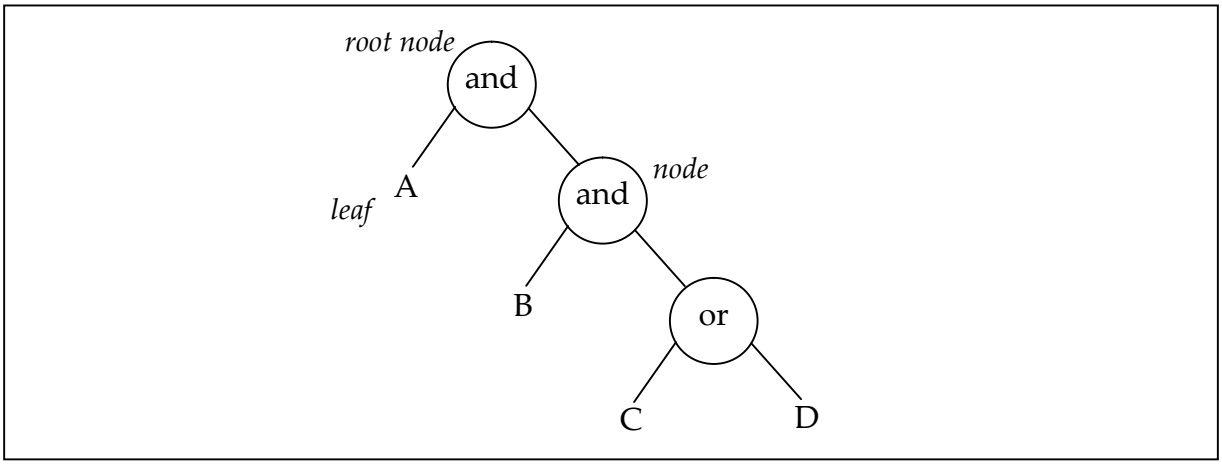


Figure 5-4: Using a tree notation to express a pattern

The ruleCore Server is partially scaleable; it consists of two major parts: the event I/O part and the engine. These two components communicate with each other through TCP/IP sockets. The event I/O part is based on Mule, an open source ESB with routing capabilities, making it is possible to run several engines on multiple servers that all receive their events of interest though the event I/O machine. Even the use of multiple event I/O machines is possible.

All events that flow to the event I/O part are stored in a persistent database. Although ruleCore has no real non-stop capabilities, the events and the states of the patterns are stored in a database, so after a server crash nothing is lost. This crash recovery mechanism can be used to create real non-stop capabilities, by letting a second server with a ruleCore engine running recover the states of the patterns and events from the crashed server.

6 Research results

In this chapter the result of the research is presented. In the first paragraph the six cases, described in paragraph 4.3 are evaluated for the three researched engines. In the second paragraph the results of the performance tests, described in paragraph 4.4 are presented and discussed.

6.1 Six CEP cases

In this paragraph the six CEP cases are evaluated in the different CEP engine languages. Each case will be presented in a new subparagraph. Only in the first case the patterns of the different engines are shown, and discussed in detail. The patterns of all cases are included in the appendix. From the five other cases only striking differences are discussed.

6.1.1 Case 1

The detailed description of this case can be found in subparagraph 4.3.1. The pattern that has to be detected in this case, detects the situation where customer A cancels an already packaged order and customer B orders the same product. The following pattern will detect this situation:

"orderPackaged and orderCancelled(where orderID=orderPackaged.orderID) and newOrder(where productID=orderPackaged.productID)"

Now the patterns to detect this situation in Esper, StreamCruncher and ruleCore are presented in *Table 6-1*.

It is notable that there is a big difference between the different pattern languages. The Esper pattern really looks like the general CEP language that is defined in *Definition 4-1*. In fact it is the other way around: the general CEP language is inspired by the

CEP language of Esper, this is because the notation really feels intuitive when defining patterns. However there are some differences: note the "every" keyword in the beginning of the pattern. This every keyword sees to it that for every new orderPackaged event that arrives at the engine a new pattern instance is created. Another difference is the way to create a reference to an event. In the general CEP language this is done by adding increasing numbers to events form the same type to prevent ambiguousness, but when an event type only occurrence, the reference to it is just its type name. Also the use of the sequence operator catches the eye. The reason why this is necessary is because when the orderCancelled event arrives the engine wants to evaluate the attribute of the reference a, in this case the orderPackaged event. So using the "and" operator in combination with an attribute match between both events is not possible. Of course there is a workaround: lets say the pattern "A and B (where ID=A.ID)" is defined. This pattern looks for an A and B event with an equal "ID" attribute. It doesn't matter which events arrives first, otherwise the sequence operator should have been used. In Esper the pattern "a=A and B(ID=a.ID)" is not valid, because of the 'dangling reference' problem described earlier. A work around with use of the sequence operator is able to express the same as the pattern with the "and" operator and the attribute matching between the two events: "(a=A -> B(ID=a.ID)) or (b=B -> A(ID=b.ID))".

When looking at the StreamCruncher pattern there are several notable items. The first item is the structure, this looks very much like SQL (Structured Query Language) that is used for most databases. An advantage of this is that it will look familiar to many developers, a disadvantage is that the structure is less suitable for describing CEP patterns and therefore takes more writing effort. The second item is the use of streams. StreamCruncher works with streams of events, where every streams can only hold one type of event. In the world of business events (like newOrder, newCustomer, etcetera) the use of streams seems not so logical. There are

many event types used in enterprise IT systems and the interval between the events can be much larger than the interval between events in event streams, therefore working with loose events seems much more intuitive. The last item are the partitions that are defined manually, in the pattern. This partitions are a sort of temporal memory that remember the last X events in a stream or the last X minutes of events in a stream.

Esper	StreamCruncher	ruleCore
"every a=orderPackaged -> (orderCancelled(orderID=a.orderID) and newOrder(productID= a.productID))"	select newOrderStream.orderId, orderPackStream.orderId, orderPackStream.productId from orderPack (partition store last 40 minutes) as orderPackStream, orderCancel (partition store last 40 minutes) as orderCancelStream, newOrder (partition store latest 1000) as newOrderStream where newOrderStream.$row_status is new and orderPackStream.$row_status is not dead and orderCancelStream.$row_status is not dead and newOrderStream.productId = orderPackStream.productId and orderPackStream.orderId = orderCancelStream.orderId;	<view> <match> <value> <event>$xpath("event-def[@eventType= "orderPackaged"]")</event> <field>$xpath("EventBody/orderID")</field> </value> <value> <event>$xpath("event-def[@eventType= "orderCancelled"]")</event> <field>$xpath("EventBody/orderID")</field> </value> </match> <match> <value> <event>$xpath("event-def[@eventType= "newOrder"]")</event> <field>$xpath("EventBody/productID")</field> </value> <value> <event>$xpath("event-def[@eventType= "orderPackaged"]")</event> <field>$xpath("EventBody/productID")</field> </value> </match> </view> <detector> <and> <event-pickup>$xpath("view/event[@type= "orderPackaged"]")</event-pickup> <event-pickup>$xpath("view/event[@type= "orderCancelled"]")</event-pickup> <event-pickup>$xpath("view/event[@type= "newOrder"]")</event-pickup> </and> </detector>

Table 6-1: Patterns for case 1

The first item noted from the ruleCore pattern is again the structure. ruleCore patterns are defined in XML. This can be seen as an advantage, XML being one of the most embraced standards for data exchange now-a-days. Attributes of an event are selected with the <value> tag. The matching of attributes is done with the <match> tag. All matching is done in the first definition: the view definition. After the view is defined the detector has to be defined. In this definition the different operators are defined. In this pattern the two "and" operators from the general pattern can be replaced by only one "and" operator in the <detector> part, due to the fact that the two "and" operators in the general pattern are used to check three events (with their attributes).

6.1.2 Case 2 to 6

In case 2, two events of the same type are correlated. Because StreamCruncher works with event streams it has a special self-join operator to effectively match multiple events from the same stream.

Also case 2 shows that next to matching event attributes with each other also matching event attributes with a value is done in the <view> part, for this the <assert> tag is used. Also case 3 shows a new tag for the ruleCore <view> part: the <age> tag. With the age tag it is possible to create time related patterns. Only it does not seem logical to use the <age> tag in the <view> part, as the <age> tag is a time operator it would better fit in the <detector> part where all other operators are used. Why ruleCore chose for this location for the <age> tag is unclear, but that it leads to problems becomes clear in case 6. Case six consists of two parts, both with a time operator. These parts are joined with a sequence operator. Because the time operator (<age> tag) is in the view part and the rest of the operators are in the <detector> part case 6 can not be expressed in one ruleCore pattern. When it is split into two parts; the first part detecting the two customer calls within two weeks and emitting a

customerCallsALot event when this situation happens, and the second part that uses this customerCallsALot event and then looks for no new orders within two weeks.

When looking at the Esper patterns for case 5 and 6 it becomes clear that there are two types of timers in Esper: "timer:within" and "timer:interval". The "timer:within" acts as a sort of stopwatch: if that part of the pattern where the "timer:within" is defined does not become true within the specified time then the pattern becomes false. "timer:interval" waits before that part of the pattern where it is defined becomes true. A many used situation is the combination with "and not". For example the situation where an A event can not be followed by a B event within 60 seconds.

The last thing that is worth noting is the way StreamCruncher solves the last case. It uses a ordinary SQL-exists clause and a chained partition. The first partition counts the number of service calls per customer in an interval of two weeks, while the seconds partition checks if this number is larger than 1. If that is the case then it starts checking if that customer does not place a new order within two weeks.

6.2 *Performance tests*

In this paragraph the results of the performance tests, defined in paragraph 4.4, are shown and discussed. All shown scores are averages of three executed tests. Because Esper and StreamCruncher are both written in java, first the Java Virtual Machine (JVM) tuning that is used per engine is given:

- Esper: Only JVM tuning is used at the last throughput test with noise (X = 50.000, Y = 7), because this test needed more memory. The used JVM tuning in this case was: "-Xms512m -Xmx512m -XX:+UseParNewGC".
- StreamCruncher: For all tests with StreamCruncher the JVM used tuning was: "-server -XX:+UseBiasedLocking -XX:CompileThreshold=5000 -XX:ThreadStackSize=256 -XX:+UseParallelGC -XX:ParallelGCThreads=2 -XX:MaxGCPauseMillis=100 -Xms768m -Xmx768m".

Also it should be noted that the next release of StreamCruncher should give a great performance boost (30%-80%), because that version will use byte code instead of interpreted code.

First the latency tests are shown. As a reminder the pattern to detect and the sent events per test are given:

- Pattern to detect: "A and B (where ID = A.ID) -> C (where ID = A.ID)"
- Sent events (Y are noise events): "A(1), *Y times B(2)*, B(1), *Y times C(2)*, C(1)"

	Y = 0	Y = 50	Y = 100	Y = 200	Y = 500	Y = 2.000
Esper	15	18	21	28	43	80
StreamCruncher	13	47	100	484	331	361

Table 6-2: Latency tests results

In *Table 6-2* the results of the latency tests are given, to show the course of the results better they are also shown in a graph in *Figure 6-1*. The first test (Y = 0) is without noise. The curve in the beginning of the StreamCruncher results line is probably due to optimization of the StreamCruncher engine for higher volumes. In the end both Esper and StreamCruncher seem to develop a linear results line. Please note that the number of noise events in the graph are twice as high as the corresponding Y number. That is because in the send events part the Y is used twice.

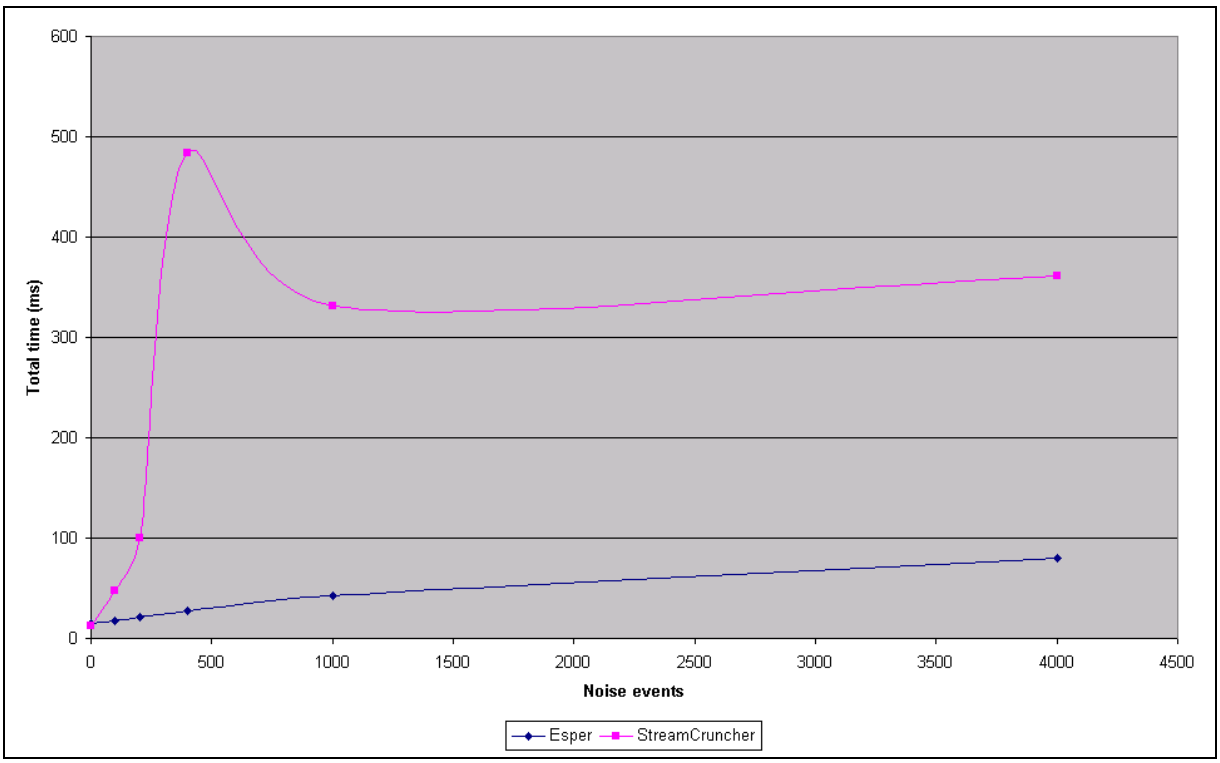


Figure 6-1: Latency tests results

Now the results of the throughput tests, with and without noise are shown. First the patterns to detect and the sent events per test are given:

- Patterns to detect:
  - "A and B (where ID = A.ID) -> C (where ID = A.ID)"
  - "C -> D (where ID = C.ID)"
  - "A and C (where ID = A.ID) and D (where ID = A.ID)"
- Sent events (tests without noise): X times (A(x), B(x), C(x), D(x))
- Sent events (tests with noise): X times (A(x), *Y times B(-1)*, B(x), C(x), *Y times C(-1)*, D(x))

In *Table 6-3* the results of the throughput tests without noise are shown. The X = 200.000 test results of StreamCruncher are shown between brackets because not all patterns that had to be detected were detected, but the test did finish. The tests were

run on a older model notebook, so it is possible that StreamCruncher hit the roof of the notebooks performance with this many events.

	Esper (ms)	SC (ms)	Esper (MB)	SC (MB)
X = 2	30	159	35	35
X = 500	474	981	35	46
X = 1.000	494	641	35	56
X = 2.000	574	1.073	35	78
X = 5.000	744	1.685	35	93
X = 10.000	1.035	2.763	35	117
X = 50.000	3.291	14.869	35	224
X = 200.000	11.660	(92.723)	35	(431)
X = 500.000	28.024		36	

Table 6-3: Throughput tests results (without noise)

From the graph in *Figure 6-2*, that shows the results of the throughput tests without noise, it becomes clear that Esper scales roughly linearly while StreamCruncher results show a much steeper line. Also the event numbers in the graph are higher than the corresponding X numbers, for the same reason as with the latency tests.

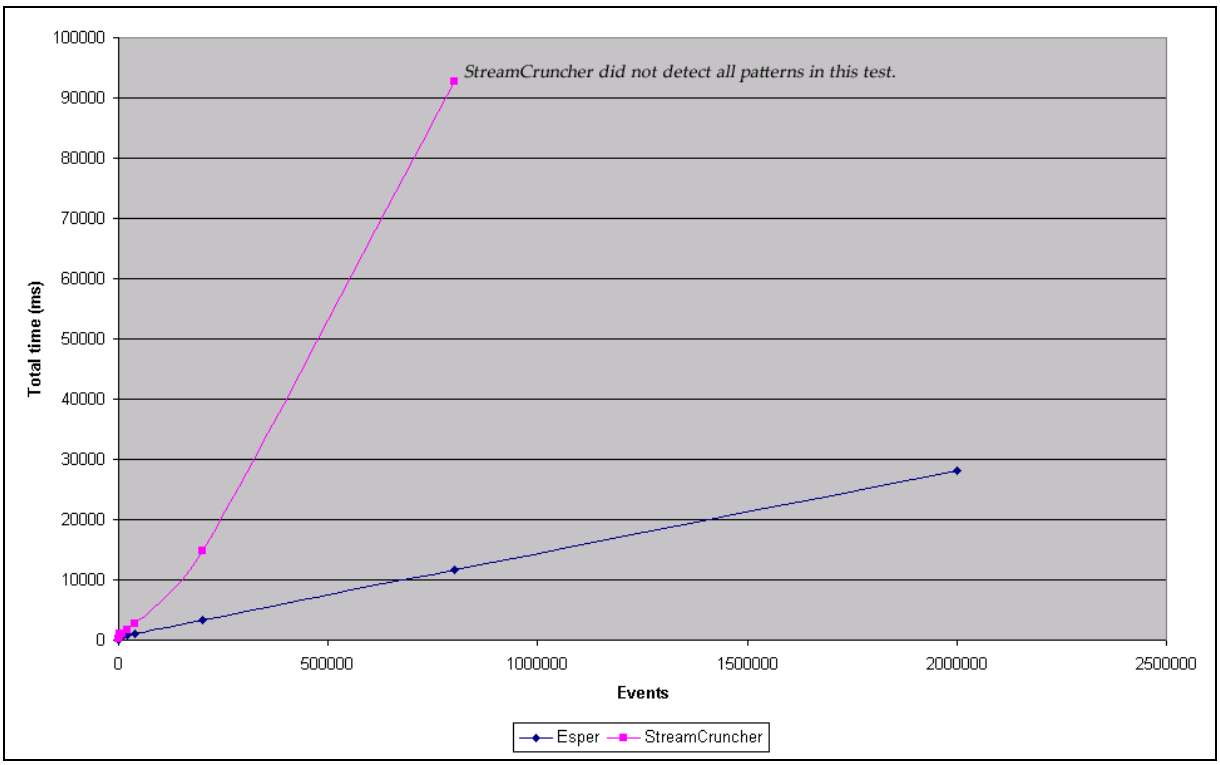


Figure 6-2: Throughput tests results (without noise)

Table 6-4 shows the results of the throughput tests with noise. Again the last StreamCruncher test did not detect all patterns. Also note the memory use of the last Esper test, that is because of the needed JVM tuning to let Esper complete that test. Without this tuning a heap overflow error occurred.

	Esper (ms)	SC (ms)	Esper (MB)	SC (MB)
X = 2, Y = 1	37	116	35	35
X = 500, Y = 2	494	818	35	46
X = 1.000, Y = 3	608	878	35	56
X = 2.000, Y = 4	905	1.118	35	89
X = 5.000, Y = 5	1.592	2.529	35	113
X = 10.000, Y = 6	3.107	7.373	35	144
X = 50.000, Y = 7	13.877	(82.364)	222	(449)

Table 6-4: Throughput tests results (with noise)

In the graph shown in *Figure 6-3* the results of the throughput tests are shown. To make comparison easy also the results of the throughput tests without noise are shown. The last throughput test with noise is not added to the graph because that would make the graph zoom out too much. Do not underestimate the number of events in these tests. The 'normal' A(x), B(x), C(x), and D(x) events are all sent X times, so that makes a total of X * 4 'normal' events. Because the Y is used two times in the main loop (X) the number of noise events is equal to: X * Y * 2. So the total number of all sent events (normal and noise) in the last test is: 50.000 * 4 + 50.000 * 7 * 2 = 200.000 + 700.000 = 900.000. Note that the events number used in the graph represents only the normal events, because otherwise the domain of the X-axle would be too large and details would have been lost.

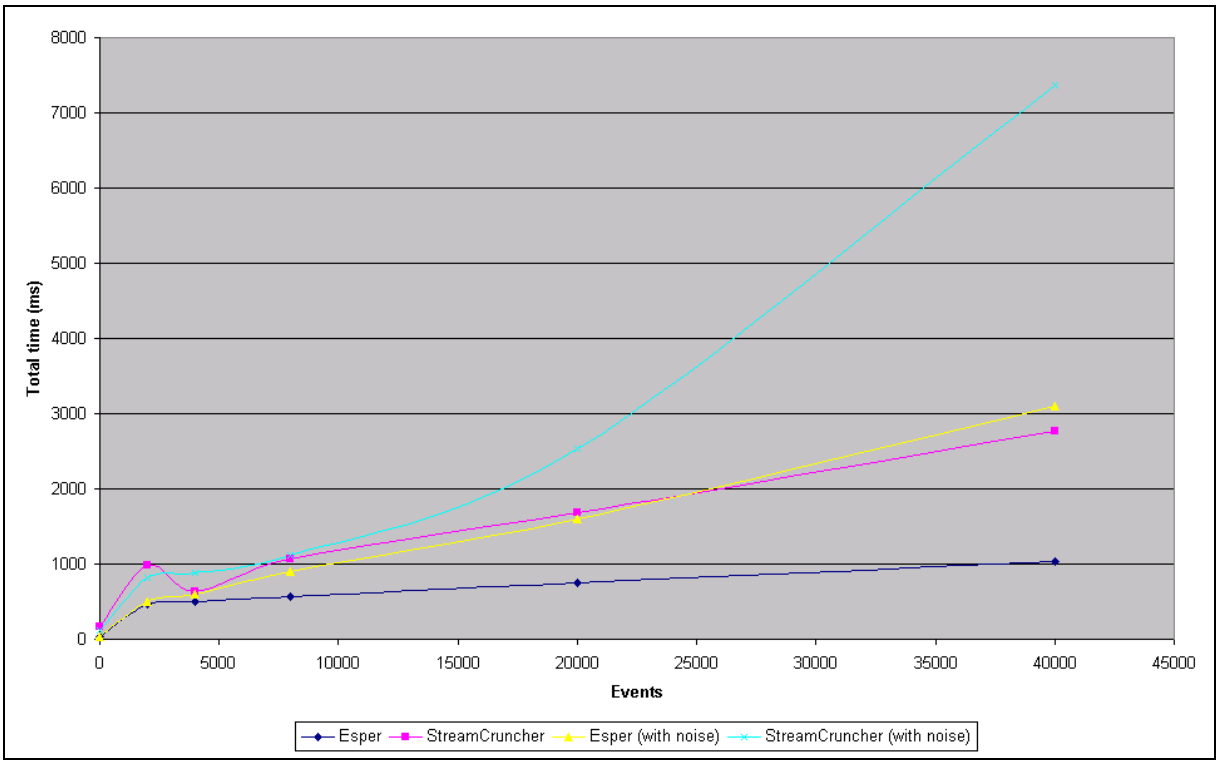


Figure 6-3: Throughput tests results (with noise)

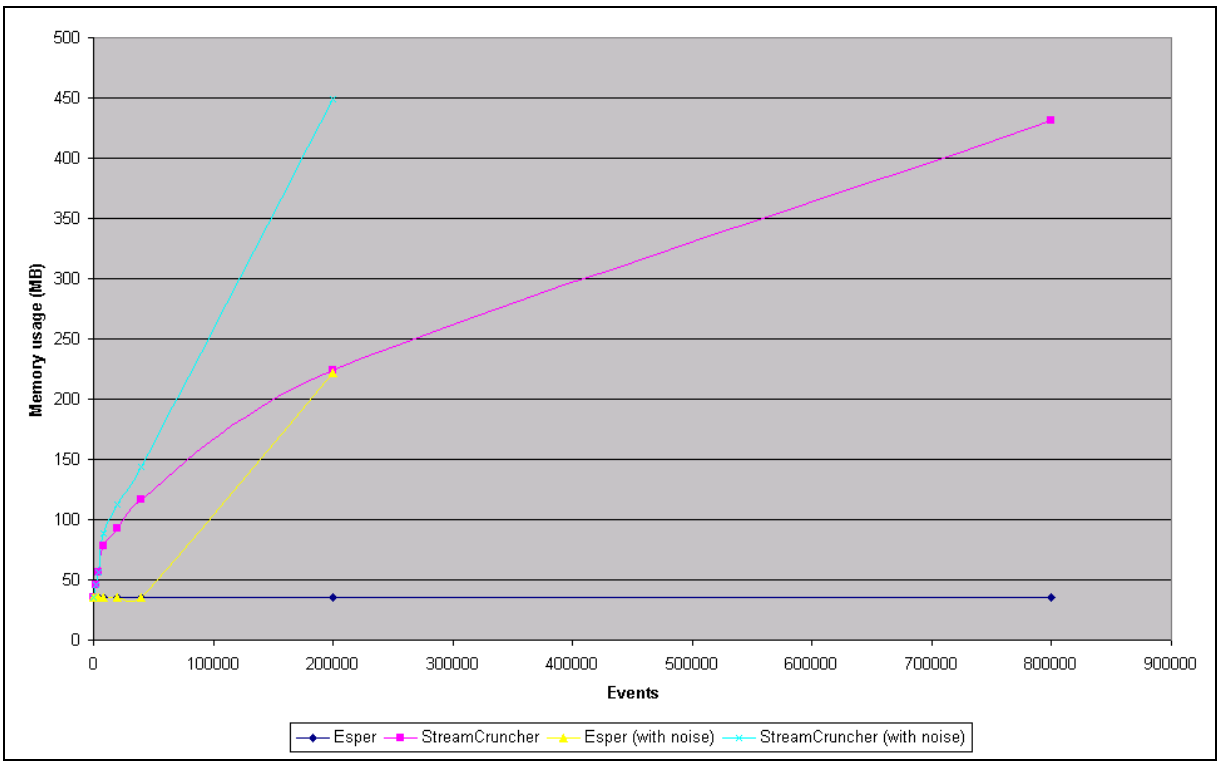


Figure 6-4: Memory usage

Figure 6-4 shows the memory usage of Esper and StreamCruncher while running the different throughput tests. As stated before only for the last test with noise, JVM tuning for Esper had to be used in order to give the engine enough memory. In all other tests Esper only used roughly 35 MB. StreamCruncher's memory usage increased when the number of events did. This is probably due to the fact that StreamCruncher has to build up the four events streams and then runs the queries against them, while Esper evaluates the incoming events on-the-fly. Also in this tests StreamCruncher stored the test result in memory.

7 Conclusions

In this chapter the conclusions, based on the research questions are drawn, also an indication for possible future research is provided. The conclusions on the innovation management research can be found in chapter 8.

7.1 Key concepts

The first research question deals with key concepts that are involved with CEP. After an intense literature study it became clear that CEP is linked with many concepts, like event flow intensity, algorithms, event communication and diversity, complex pattern description, and of course enterprise IT systems. The most relevant concepts are introduced in chapter 2. This first question also lead to the definition of a CEP framework, which helps to integrate a CEP engine within an enterprise IT system.

7.2 Optimal engine

The second research question concerns about what the optimal CEP engine would be for integrating with the Cordys environment. In order to answer this question there are three different engines chosen: Esper, StreamCruncher, and ruleCore Server. To compare these three engines six representative CEP business cases are defined and expressed for all three engines. The results of this comparison are extensively described and discussed in chapter 6. *The overall conclusion is that all three engines were able to handle all six cases, so more roads lead to Rome.* Is one road better, or maybe shorter than the other? Yes, while every engine has its advantage and disadvantage, Esper seems to be the most mature engine. Their combination of ESP and CEP is well-considered and their CEP pattern language is short and intuitive. Also the performance tests show that Esper is performing very well, even on a dated laptop.

7.3 Recommendations

While conducting this research, it became clear that CEP is a very promising technology to enhance current enterprise IT systems. However, no real standards are formed at the moment and it is very important for Cordys to follow this closely and even take part in this standardization process if possible.

The Cordys SOA grid is very advanced and with minor alterations/extensions it should be capable to incorporate a CEP engine. Esper really shows some promising features and it should definitely be taken into consideration for use within the Cordys environment, because developing a CEP engine from the scratch can cost a lot of resources and knowledge.

7.4 Future research

Talking about ESP and CEP often leads to a discussion if these are two different concepts or that they are both (roughly) the same. My view on this is that ESP is more about high throughput and streams with one event type per stream, while CEP is more about detecting more complex situations. Further research has to be done in order to provide clearness in this discussion. Also it is highly necessary that CEP standards are formed. The CEP glossary on the website of David Luckham is a start for this. These standards should insure that integrating CEP in common enterprise IT systems becomes a smooth process.

7.5 Final words

This thesis should give some understanding about the world of CEP, which is certainly one of the most promising developments for enterprise IT systems at the moment, and for the near future. I hope that this thesis, and the six CEP business cases in particular, put people to think about what is possible with CEP. More research in the CEP field are necessary and hopefully this thesis helps waking the interest of researchers for this interesting and promising subject.

8 Innovation management on three levels

A recent book on innovation management, "Managing Innovation" from Tidd et al. [2005] shows that innovation is a very important aspect for corporations to ensure continuity, staying ahead of competition and keeping customers satisfied. Innovation does not come for 'free', it should be a well structured, managed process. In this research innovation management will be viewed on three different levels: product/project level, company level and global/national level. What are the similarities and differences on these three levels and do they link up to each other or are there gaps between them? An interview within a medium sized high-tech company will reveal if this theory is put into practice.

8.1 Overview

Tidd et al. [2005] conclude in the second chapter of their book that innovation management is more than just an invention: "Definitions of innovation may vary in their wording, but they all stress the need to complete the development and exploitation aspects of new knowledge, not just its invention". So innovation management is "the process of growing innovations into practical use".

What is the link between innovation management and CEP? These are both quite different subjects, but CEP can be seen as an innovation in current enterprise IT systems. Working "on" CEP in such systems should be a well managed process, not only for the companies developing CEP, but also for the clients using it. The study of innovation management on three different levels, which are introduced hereafter, can contribute to that, and to many other innovative processes.

- *Product / project level*

  The product level describes the trajectory that a product or project has to cover to have the greatest chance of success. For this level a paper about Stage-Gate Systems from Robert G. Cooper [1990] is used.

- *Company level*

  The company level is more focussed on a long term overview. Two technologically focussed views are combined: the technology maps from the book "Marketing of High-Technology Products and Innovations" written by Mohr et all. [2005] and the article "Roadmapping in the Corporation" by Albright and Kappel [2003].

- *Global / National level*

  The research of Dunphy et al. [1996] reveals four global and national level innovation factors. Also some additions to the micro level (product/project and company level) are shown.

Innovation management studies, like the four used for this research [Cooper, Mohr, Albright, Dunphy], are often focused to only one level of innovation management. It is very important to see how these different levels relate to each other, because possible gaps and overlaps can obstruct integration. This research focuses on that and the following research question is formulated:

"What are factors that involve innovation management on different levels and how do they integrate?"

The management research model, shown in *Figure 8-1*, helps answering this question.

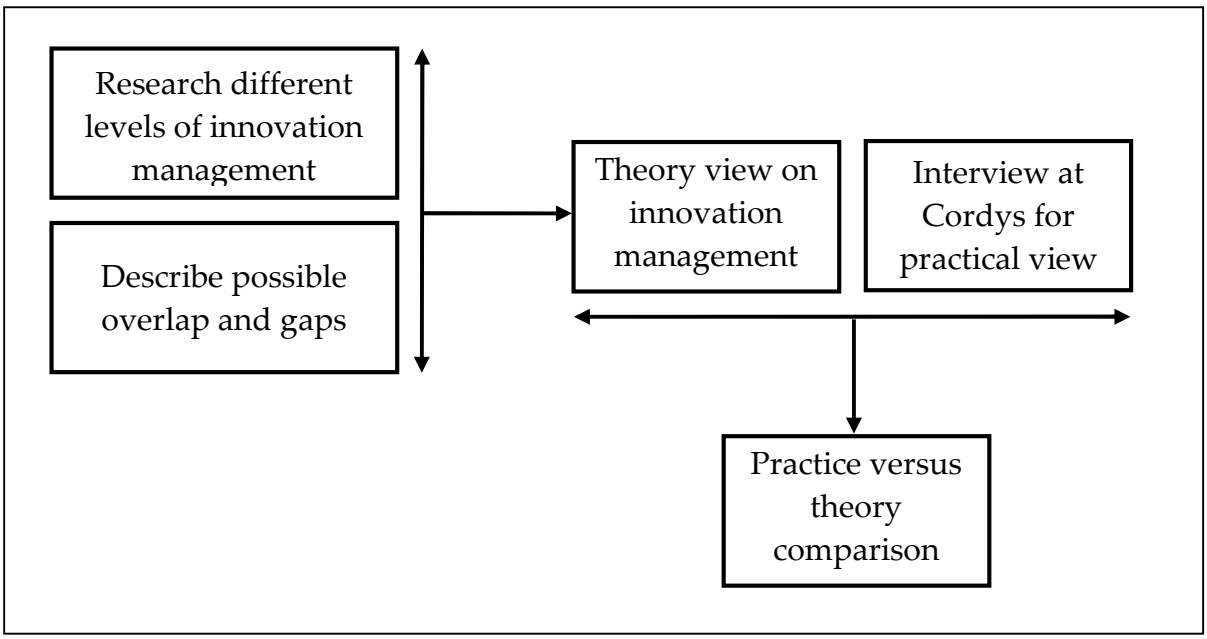


Figure 8-1: Management research model

The upper left block in this research represents the theory view of innovation management, viewed upon three different levels, from three different insights. A zoom-out view of these three levels is used because the first part of this thesis is a zoom-in view on a single innovation: CEP. Also the most zoomed-out view; global/national level, reveals many interesting factors that are not so common in most innovation management researches.

To complete the theory view the different overlaps and gaps are analysed in paragraph 8.5, titled: 'Gaps and overlap'.

After the theory view is completed an interview at Cordys is used to see what innovation management constructs are used in a midsized high-tech company like Cordys.

8.2 Stage-Gate Systems

In 1988 Robert G. Cooper introduced Stage-Gate systems [1990] to give corporations a tool to manage their innovative projects. In many different fields corporations are battling for their share in the market. The key in winning this war is, according to

Cooper, "To get better at the innovation process: to drive new products from idea to market faster and with fewer mistakes". Many researches share this thought: A study by the Conference Board [Hopkins 1980] shows that most CEOs believed that new products would become much more important for their firms in the coming years. A Coopers & Lybrand survey [1985] reported that most corporations are counting heavily on new product development for growth and profitability. Even an annual Fortune survey among the top American corporations underlines the importance of innovation: "The results were provocative: The single strongest predictor of investment value is 'degree of innovativeness of the company'".

So it becomes clear that innovation is a very important weapon for companies in their struggle for market share and survival. A new tool for managing innovation is sorely needed because only one in four projects becomes a winner. Furthermore corporations are facing increased pressure to reduce cycle time (time to market) and at the same time improve the 'hit rate' of their new products.

Stage-Gate systems form one solution for these firms to effectively manage their innovations. What exactly are stage gate systems? "A Stage-Gate systems is both a conceptual and an operational model for moving a new product from idea to launch. It is a blueprint for managing the new product process to improve effectiveness and efficiency." Stage-Gate systems recognize that product innovation is a process. And like other processes, innovation can be managed. Stage-Gate systems simply apply process-management methodologies to this innovation process. The innovation process can be compared with the production process of a physical product. To improve the quality of the output of the process the focus should be on the process itself: "to remove variances in the process".

Typically a process is divided into a number of stages. To keep the output quality under control there is a quality control checkpoint or gate at the end of each stage. For each gate a set of deliverables and quality criteria are specified. The product has

to pass this gate in order to continue to the next stage. The work is done in the stages and the gates guard the quality and progress of the product. Note that it is not possible to work on every innovative idea, the first gate acts as a selection device but when there are too many ideas to evaluate possible some pre-selection might be necessary.

Stage-Gate systems work on a similar way. The innovation process is divided into a set of predetermined stages. These stages consist of prescribed, related and often parallel activities. The number of stages and gates may vary from implementation to implementation, but are often between four and seven stages and gates. Also in general each stage gets more expensive than its preceding one, but also information becomes better so risk is managed.

In *Figure 8-2* an overview of a typical Stage-Gate system is shown. Before each stage is a gate that guards the entrance to that stage. This gate controls the process and is characterized by a set of deliverables or inputs, a set of exit criteria, and an output.

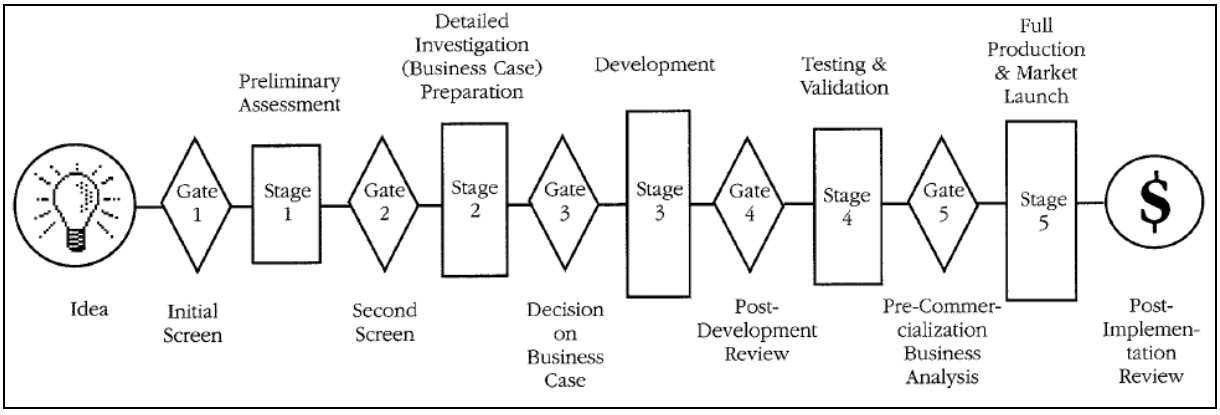


Figure 8-2: Stage-Gate system overview

- *Gate 1: Initial Screen*

  At this gate there is decided if an idea has enough potential to be further developed. This first screen is very "gentle" and doesn't include financial criteria. Only a handful of key "must meet" and "should meet" criteria that deal with strategic alignment, project feasibility, magnitude of the

opportunity, differential advantage, synergy with the firm's core business and resources, and market attractiveness are included.

- *Stage 1: Preliminary Assessment*

  This first, inexpensive stage has the objective to gather more information about the market and the technical aspects of the product. In this short stage a preliminary market assessment is carried out, this involves a variety of relatively inexpensive activities: a library search, contacts with key users, focus groups, and a quick concept test with a handful of potential users. Purpose of the preliminary market assessment is to determine market size, market potential, and likely market acceptance. The purpose of the technical preliminary assessment is to assess development and manufacturing feasibility, and possible costs and times to execute.

  In summary the first stage provides for the gathering of market and technical information, at low cost and in a short time, so the project can be re-evaluated more thoroughly at the next gate.

- *Gate 2: Second Screen*

  This gate is more a repeat of the first gate. The same "must meet" and "should meet" criteria are considered, but now with the new information gathered in the first stage. Some additional "should meet" criteria dealing with sales force and customer reaction to the proposed product are introduced. The Go/No go decision that has to be taken at this point is very important, because the next stage will bring increasing costs.

- *Stage 2: Definition*

This is the last stage before the development starts. In this stage the definition of the project must be very clear. Because the next stage will be very expensive it is important that the prospects of the product are very good and it should be technically feasible. Market research studies are undertaken to determine the customer's needs, wants and preferences to help define the "winning" new product. Also a competitive analysis, concept testing and a detailed technical appraisal are part of this stage. Finally, a detailed financial analysis is conducted as an input to the next gate.

- *Gate 3: Decision on Business Case*

  Gate 3 is a very important gate, because this is the gate prior to the very expensive development stage. The financial analysis, conducted in stage 2 has to be thoroughly checked. Once again the "must meet" and "should meet" criteria from gate 2 are reviewed, and as a qualitative addition the activities in stage 2 are reviewed: checking of the quality of execution was sound, and the results were positive. The second part of gate 3 concerns definition of the project. Agreement on a number of key items must me reached, including: target market definition, definition of the product concept, specification of a product positioning strategy, delineation of the product benefits to be delivered, and last but not least agreement on essential and desired product features, attributes, and specifications.

- *Stage 3: Development*

  Next to development also detailed testing, marketing and operation plans are part of this stage. Also an updated financial analysis is prepared and legal/patent/copyright issues are solved.

- *Gate 4: Post-Development Review*

  The fourth gate checks the progress and continued attractiveness of the product and project. The development work done in stage 3 is reviewed and checked. Also the economic questions is revisited via a revised financial analysis based on new and more accurate data. Test and validation plans for stage 4 are approved.

- *Stage 4: Validation*

  This stage is all about validating the entire viability of the project, including: the product itself, the production process, customer acceptance, and the economics of the project. Validating is done by a number of activities: in-house product tests, user of field trails of the product, trial or pilot production, pre test market or trial sell, and a revised financial analysis.

- *Gate 5: Pre-Commercialization Decision*

  This is the final gate where a project still can be killed. After this gate the door to the last stage is opened and full commercialization starts. The main function is this gate is to review and check the quality and the results of the activities in the fourth stage. Before going ahead financial aspects are checked once again, after that the operations and marketing plans are reviewed and approved.

- *Stage 5: Commercialization*

  In this final stage the marketing launch plan and the operation plan are both implemented.

- *Post-Implementation Review*

  After the commercialization stage the product becomes a "regular product" of the firm. This is also the right point to review the performance of the project and product. Finally a post-audit is carried out (a critical assessment of the project's strengths and weaknesses, learning points and what can be done better in the future) and implemented.

The gates are manned by senior managers that together form a multidisciplinary and multifunctional group with enough experience to take the decisions and enough authority to approve the needed resources. Also some organizational changes may be required in order to implement a Stage-Gate system in a firm: project teams are fundamental for this approach, where projects no longer moves from department to department but one project leader carries the project through all stages.

Stage-Gate systems seem to form a sort of paved road for a project to travel on, where crossings guard the quality and progress of the project. The project being a truck, driven by the project leader that leads the project through all gates, ensuring the greatest change on success.

After this zoomed-in view on innovation management, on product/project level, we zoom-out to company level. Mohr [2005] gives a great tool for managing innovations that is introduced in the next paragraph.

8.3 Technology Maps

When developing new products it is, according to Mohr [2005], very important to be up to date on the latest technological developments and to be sure that new products enter the market on time. A tool to help with this is the *technology map*. This map, shown in *Figure 8-3*, describes the stream of new products (breakthrough and incremental) that a company can develop in the coming time. This technology map can act as a flexible blueprint for corporations and has to be revised on a regular base.

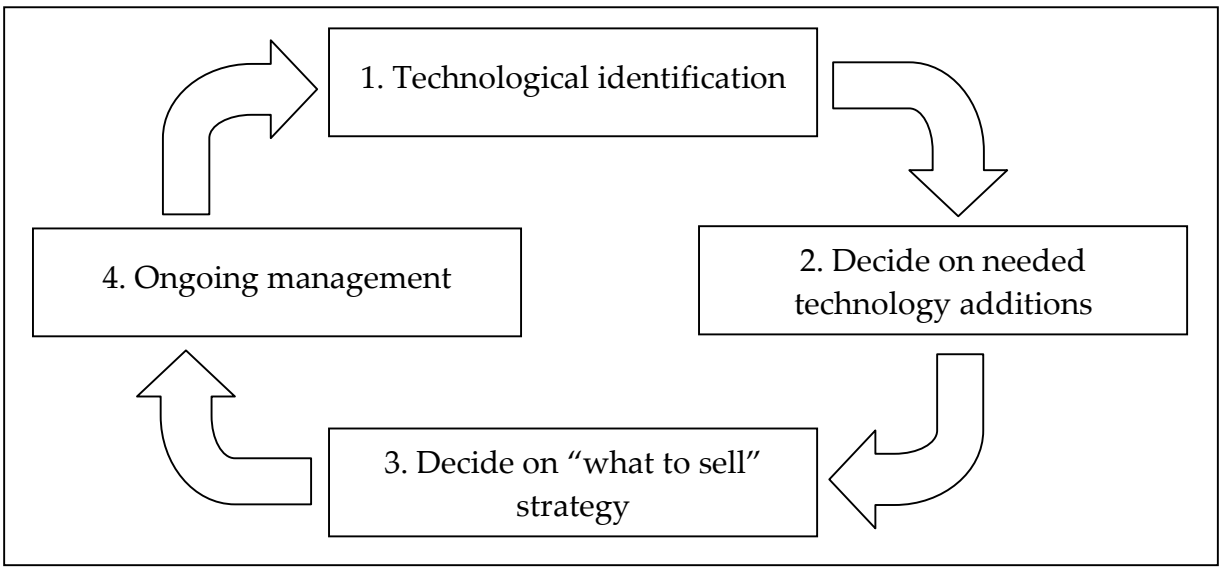


Figure 8-3: Technology map

The technology map consists of four phases:

- *Technological identification*

  Decide which ideas are the most valuable, based on al present knowledge in a company.

- *Decide on needed technology additions*

  Weaknesses from the first phase can arise that can be solved with the use of extra knowledge. This knowledge can be developed internally, but can also be acquired externally for example by buying a company that possesses the needed knowledge.

- *Decide on "what to sell" strategy*

  The marketing issue is the crux in this phase. The company has to decide when during the process of development the product has to be advertised. Also the decision about what to sell has to be made. Is the company going to

produce, market and sell the end product itself, or are they going to outsource this. Another possibility is to sell the technology in the form of licenses.

- *Ongoing management*

  In this last phase it is of great importance that the core technology is well managed, this includes managing the development of possible product derivations. Also it is important to know when a new project has to be stopped.

To strengthen the innovation management view on this level a second model is used from the article "Roadmapping in the corporation" by Albright and Kappel [2003]. Both models have their strong points, but technology is definitely the central keyword. The "Product-Technology Roadmap" is the central model in the article from Albright and Kappel. In this model there are four different sections that are short reviewed below. After that the strongest points of both models are integrated.

- *The market*

  This sections handles different market related subjects like market size and growth, the most import customer needs and an analysis of the strong and weak points of the competitors.

- *The product*

  In the product section the central notion is the product drivers. Product drivers are the properties of a product that potential customers value the most, and that they find the most important aspect when considering to buy such product.

- *The technology*

  The technology section is the central Point within the roadmap and contains much information. Technological changes are not only written as evolution in time, but are also linked to the product strategy. Also it is important that only technologies that support the product drivers are mentioned.

- *Summary / action plan*

  This section is referred to as the 'attack strategy'. The purpose of this section is to define the technologies with the highest priority and to identify the matching action plans that are necessary for developing these technologies.

Similarities and differences

The two described models have a number of similarities. In both models technology is the central keyword. Also the market where the product is sold is very important. A difference between the models is that the model from Mohr [2005] starts with the use of present knowledge in a company to define product development, while the model of Albright and Kappel [2003] points at the product drivers for this definition. Putting the product drivers central looks like a good idea, product drivers are namely the properties within a product group that contribute to the decision of customers choosing for your product and not that of the competitor. This product drivers should not be lost out of sight while further developing the product, because this can lead to losing customers and with that the market position. Below is a summary that combines the best of both models:

- *The market*

  This sections is taken from the "Product-Technology roadmap" model and handles different market related subjects like market size and growth, the

most import customer needs and an analysis of the strong and weak points of the competitors.

- *The product*

  The base of this section is taken from the "Product-Technology roadmap" model. In the product section the central notion is the product drivers. Product drivers are the properties of a product that potential customers value the most and that they find the most important aspect when considering to buy such product.

  From the "Technology maps" model the process of defining exactly what is sold is added. Next to the product itself this can also be the technology behind the product, when doing so the technology becomes the actual product.

- *The technology*

  The base of this section is again taken from the "Product-Technology roadmap" model. The technology section is the central Point within the roadmap and contains much information. Technological changes are not only written as evolution in time, but are also linked to the product strategy. Also it is important that only technologies that support the product drivers are mentioned.

  From the "Technology maps" model the process of defining which knowledge is present within the company and which knowledge should be acquired from external sources. This is important because innovations that need much external knowledge are maybe more risky or expensive than the ones that depend on present internal knowledge.

- *Action plan*

  This sections is taken from the "Product-Technology roadmap" model and is referred to as the 'attack strategy'. The purpose of this section is to define the technologies with the highest priority and to identify the matching action plans that are necessary for developing these technologies.

After this company view level, it is time to further zoom-out, even outside the company and proceed to the global/national level, which shows for factors that influence innovation that do not directly come into mind when thinking of innovation.

8.4 Innovation Funnel

In the research "The Innovation Funnel", conducted by Steven M. Dunphy, Paul R. Herbig, and Mary E. Howes [Dunphy 1996] the path to technological innovation is divided into two levels: macro and micro level. In total seven *discriminators* on these two levels are identified. Especially the macro level is interesting because it is subdivided into two levels: global level and national level, which link up neatly to our previous two levels. The global level consists of two discriminators:

- *Technological prerequisites*

  Before most innovations become feasible often certain technological prerequisites must be met. A good example of this is the analytical engine created by Charles Babbage during the 1840s. This analytical engine can be seen as the predecessor of the modern computer. All elements of a modern computer were included in the design: memory, control, arithmetic unit and input/output devices like discs and printers. Babbage's invention was, although elaborate and ingenious, doomed to failure because it attempted to defy the then current, technological limitations. Without the necessary

innovations (Boolean algebra and electronics) it was then impossible to create a viable mechanical computer.

Often innovation is simulated by necessity. When the need of some type of product becomes critical, the time to innovation shortens significantly. Also governmental or marketplace irregularity such as the lack of sewing machines in India can delay a need. Also wartime can have its influence on certain needs: the audio tube, invented in 1906 by DeForest should, under normal usage and diffusion patterns, have led to a radio product in the late 1930s, but World War I forced governments to speed the development of wireless transmission. World War II speeded the development of penicillin because the British and later the American governments needed a drug to fight infections.

Innovations rarely occur in isolation, but rather tend to arrive in swarms or clusters; the computer was feasible by 1950, but it required a certain set of clustered innovations before it was diffusible: software, peripherals, communication links, support services, and applications.

- *Socio-cultural tendencies*

  After the adoption of any and all new technologies, social change is imminent. All societies have a certain inertia to such change, but some cultures are more conservative, risk averse, and change resistant than others. Innovations sometimes thrives in one country but not in another. For example even before the advent of the Middle Ages, China invented paper and gunpowder, but it was Western Europe that improved upon them. Algebra was an Arabic discovery, but it was the Western world that exploited the mathematics.

  The traditions of the society and the nature of the market often have their influence on innovation. Some culture even permit changes and do not reward entrepreneurs, which will postpone or even stop innovative advances. To

support innovation the cultural climate must be open to change and risk. Also the environment has its influence on innovation. Implementing changes in an organisation is easier when the environment is perceived as non-threatening. The limits of change are defined by a culture, beyond this limits the innovator may risk social ostracism or even death by violating a taboo. Culture has been shown to have profound influence on a society's innate ability to innovate. A research of Maccoby [1990] showed that the individualism of a society was positively correlated with its innovative potential; the greater the freedom of the individual to explore and express his opinions, the greater the likelihood the individual will develop new ideas. In a research of Rothwell and Wissemann [1986] a number of important adoption factors for innovations are described. Four of them were directly related to culture:

1. A willingness to face uncertainties and take balanced risks.
2. Urgency and timeliness.
3. The readiness to accept change.
4. A dynamic long-term orientation.

Culture is, next to technological prerequisites a very important discriminator of innovation.

On national level Dunphy et al. [1996] also distinguish two discriminators: infrastructure and industry structure. Together with the two global discriminators these discriminators form the upper global/national level of this innovation management research.

- *Infrastructure*

  Infrastructure has a great influence on innovation. A good example of this are the United States after World War II; Europe and Japan required over a decade to rebuild their infrastructure. The material infrastructure of the United States was still intact, just like their population (human infrastructure) was largely whole and unaffected. Because the 1950s lacked any real, global competition, it was the golden age for American industry. In the 1960s international competition emerged again and heated up in the 1970s and beyond. Immediately after World War II the U.S. had a dominant share of worldwide innovation of 75 percent. This share fell to less than 50 percent in the late 1960s. Alarmists believed that it was a sign of the fall of U.S. innovation prowess, but it was inevitable once the European and Japanese material infrastructures were rebuilt and re-established. The U.S. share of worldwide innovation should continue to drop as the infrastructure of developing countries continues to grow.

  Venture capital (financial infrastructure) also has a great influence on innovation. In the U.S. when the level of venture capital available was high, innovations have flourished. The level of innovation within a country appears to be directly linked with the ease in which capital is mad available.

  Governments form a major portion of a country's institutional infrastructure. Generally believed is that massive government regulation has a negative effect on innovation. The U.S. pharmaceutical industry is a classical example of governmental overregulation. The FDA (Food and Drug Administration), in attempting to guarantee zero adverse reactions, severely impacts the innovation capabilities of American drug companies. Because of this, diffusion often follows years behind the innovation/diffusion cycle enjoyed by foreign competitors. Government bureaucracy also has negative influence on

innovation. Often regulatory agencies then to be risk averse by nature and do not stimulate new technologies. Dunphy et al. [1996] conclude that "The greater the central government's bureaucracy and regulatory powers, the less inclination there is for innovation within the country" and "The type and power of the government also plays a strong role in determining the innovativeness of the society".

The legal infrastructure also has its influence on innovation. New innovations have more uncertainties then the mature ones and are more risky for the innovators. The newer the technology, the greater the scientific uncertainty about its risks and benefits. This predisposition inherently tilts the liability system against newness. To survive against the legal system and the regulatory system requires a large stable of lobbyists and liability lawyers, which only larger companies can afford.

- *Industry structure*

  The degree of regulation in the industry impacts on the industry's capacity for innovation. An example is the transport industry: before deregulation this industry was severely controlled by location, service and price. In this period few new participants entered this industry, afterward, many new ventures appeared. During the period of regulation, innovation stagnated, afterward competition was fierce and innovation thrived. Protective regulations to protect industries to external competition reduce the incentives to innovate. At first it seems that the protected industry is done a favour, but in the end technological obsolescence will occur. The degree of competitiveness also contributes to the need to innovate. As large company are mostly followers, smaller companies are usually the innovators.

Next to this four macro discriminators Dunphy et al. also introduce three micro level discriminators. These will only be reviewed in short because they will only serve as an addition to information from the Stage-Gate systems research and the Technology maps research.

- *Firm size*

  Large firms with high R&D expenditures should have a disproportional number of innovations. This theory is however, in contrast with reality, where larger firms do not even have a close to proportional share. Large firms suffer from grossly inefficient R&D budgets, unfocused marketing, and diverse approaches to new product development. Unfortunately, smaller firms often do not survive a innovation failure. A study of Ettlie and Rubenstein [1987] showed how the firm size correlated to the innovativeness of a company. When a company has less than 1.000 employees, no relationship between size and the number of innovations is shown. Companies with 1.200 to 11.000 employees have a positive relationship between size and the number of innovations. This relationship becomes less significant when a company has more than 11.000, but less than 40.000 employees. Companies with over 45.000 employees have a only a slight chance of developing a radical new product. Though firm size in sheer number of employees in not the whole truth. Successful, large, and innovative companies like 3M and Hewlett Packard had there own method to make innovations more successful. These companies form units that are entrepreneurial start-ups in almost every respect. So it is not the size of the firm that counts, but the size of the decision-making unit and its degree of autonomy and flexibility that determines the company's innovativeness potential.

- *Management Attitudes*

  Innovation brings change, so management must be capable of managing these changes. Sunk investments in technology have to be overcome if they inhibit new innovations. Management should be intimately involved with significant innovations. When attention is delegated downward, the firm may lose its competitive edge and much of its innovativeness.

- *Standards*

  A technological innovation does not imply a commercial success. Setting a standard is a major force in expediting the diffusion of the innovation. Standards help to remove user uncertainties about a new technology. Standards also have a downside: when a standard is set, the innovation should defuse more widely, helping the maturation process. However, future innovations may be slowed by the very factor that propelled the initial innovation: the new standards now in place.

This third and last view completes the innovation management view on three levels. In the next paragraph the gaps and overlap of these different levels are discussed.

8.5 Gaps and overlap

The three researches, discussed in paragraphs 8.2, 8.3, and 8.4 are analysed for gaps and overlap in this paragraph. To show the relations between the three researches an adapted funnel model is used, which is introduced in *Figure 8-4*.

Often when talking about innovation management the most subjects are on company or product level. Global/national level is a rare subject within innovation management, but as the research of Dunphy et al. [1996] shows many interesting factors are found on this level. Culture, being one of these factors, is often very

underestimated in its influence on innovativeness. The research shows its importance through a number of great examples of cultural influence on innovativeness.

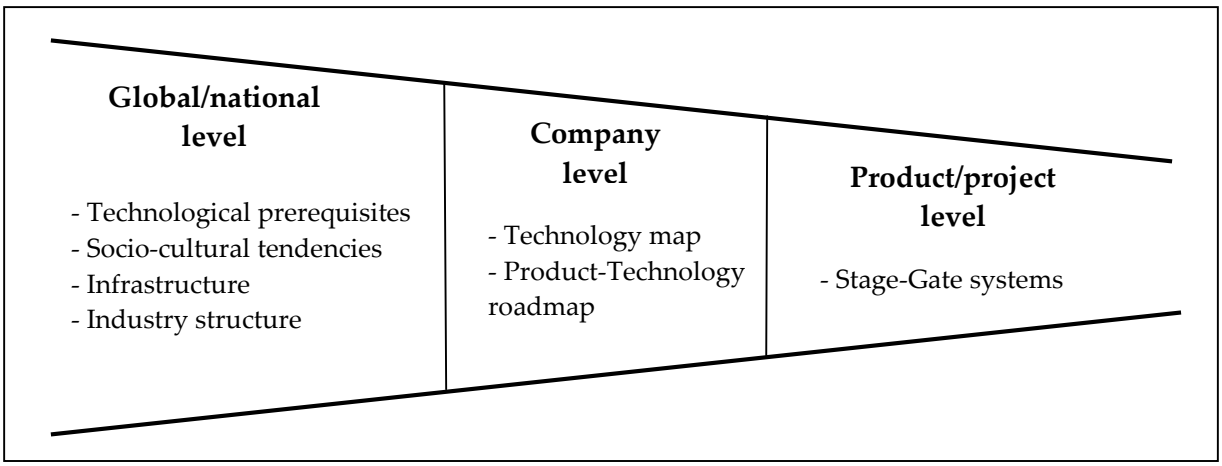


Figure 8-4: Innovation management funnel model

A very nice red line through these three researches is (market) need. On global level this is discussed in the technological prerequisites, where Dunphy et al. [1996] conclude that technological prerequisites should be met in order to do a feasible innovation and that necessity often accelerates innovation. On company level this red line is clearly present in the composed technology map where 'the market' is the first part of this model. Also in the second part of the model (the product) the market is clearly present in the form of 'product drivers', which are the main aspects for potential customers when considering to buy a new product. In the Stage-Gate systems research, on product/project level, Cooper [1990] introduces many market related operations already in the first stage (Preliminary Assessment). In his research he also emphasizes that 'doing the homework' is very important for a successful innovation. A great technological innovation is one thing, but to make it a commercial success the market need should never be underestimated.

Technology is often related with innovation, which is also the case in this three level view, as technology forms the second overlap in this combined model. On the global level technology is present in the first part 'technological prerequisites', as becomes

clear from the example of Babbage design of an analytical engine, which consisted of many parts that modern computers contain. Though it design was very good, it was doomed to failure due to the lack of the right technology in the 1840s. On company level technology is present in the whole model, but especially in the technology section, which acts as the central point of the roadmap. In the Stage-Gate systems the technology aspect is also part of the first stage. In this stage the technical aspects of the innovation are inspected and a technical preliminary assessment is conducted to assess development and manufacturing feasibility. Dunphy et al. [1996] add another nice item to this technological red line; the fact that companies that have invested a lot in a particular technology (which are called 'sunk-cost') will not easy change to a new technology, which gives them an arrear to companies that are working with new innovative technologies. So the technological leaders of today can be overtaken by the innovative new companies of tomorrow.

The two red lines show that there is enough overlap in the combined model. But what about gaps? Are there missing constructs in one of these models which do not permit seamless integration? On the global/national level Dunphy et al. [1996] discuss infrastructure and industry structure which, among other things, includes legal issues and governmental regulation. Nothing of this is used in the company level model, while these are very important factors that have great influence on innovation. Another small gap that is worth mentioning is the absence of any "what to sell" decision in the Stage-Gate systems on product/project level that Mohr [2005] has included in the technology map. The "what to sell" decision is not a pure financial aspect, which are used in the entire Stage-Gate model, but focuses on the question whether to sell, for example, an end product to customers or maybe a license of the technology behind the product and let other companies produce the read end products based on that technology.

A possible explanation for this gaps can be found when looking to the different stakeholders at the different levels. On global level governments (including politicians) are the main stakeholders, on company level high management are the main stakeholders and on product/project level lower management and technicians are the main stakeholders. The two gaps between the three levels can be explained through the differences between the stakeholders of these levels. The big differences between the stakeholders of global/national level and the company level result in the larger gap between these two levels. The differences between the stakeholders from the company level and the product/project level are much smaller, resulting in a smaller gap between these levels.

8.6 Innovation management at Cordys

This paragraph gives an insight about innovation management at a medium sized, High-tech company like Cordys. First the interview methodology is explained and after that the findings of the interview are discussed.

8.6.1 Methodology

To learn more about innovation management at Cordys there are to two interviews held. An interview with a Senior Architect and with the CTO (Chief Technical Officer) is planned because these two persons are directly involved with innovation management at Cordys.

An interview is used because of the direct communication with the interviewed and the possibility to change directions in the questions if necessary. Also open questions in an interview will most of the times be answered more extensively than for example when a questionnaire is used.

The focus of this interview is starting at the global/national level and then moves down, toward the company level and finally to the product/project level.

The open questions in the face-to-face interview are intended to provide structure, but also leave room for unforeseen direction change. In Appendix C the questions of the interview are provided.

8.6.2 Interview

A large share of the development of the Cordys product is done at their big R&D establishment in India. India's culture is very different to the culture here in the Netherlands, and a common thought is that the innovativeness of India is lacking behind that of the Western European countries, like the Netherlands. The CTO of Cordys does not agree with this common thought and states that it is more a difference in attitude than in culture. To back up this opinion he mentioned a real innovative project that comes from India: a new stunning user interface.

Staying on the global/national level, subsidy is acquired by joining projects from the JACQUARD program, a research program that focuses on software engineering. Joining such projects helps innovation of software engineering techniques and also gives some recognition.

Another important factor on this level is the government. While some subsidies are provided the government can help innovative companies more when acting as a reference customer. The CTO of Cordys states that a reference customer is very important for a successful innovation and it is very hard to get these reference customers, so the government could be a big help in this when taking this reference role into account.

When looking at company level Cordys is working with business requirements (BR) to define important improvements and innovations to their products. These BR's are coming mostly from (big) customers and R&D at the moment. In the ideal situation they should come mostly from the market but that is not yet the case at this moment. From these BR's a prove of concept (POC) is created. Then the Product Committee

(PC) proposes these POC's to the Product Management Board (PMB) that has to approve them.

While technology was the original drive for Cordys, it is now the market that becomes more and more important. At this moment they are in the middle of this transformation process, and present customers are their main drive. This is also noticeable in the decisions about which BR's are developed further to a real improvement / innovation; the (big) customers are the driving force at the moment, and money should come in.

Arriving at the lowest level: product/project level, it is interesting to see how Cordys structures the projects that are evolved from the BR's / POC's. The teams work in sprints that last four weeks. After each sprint there is a sprint review with the program managers that have to make sure that the project is still running according to plan. The sprints are development phases, before development definition and retrieving more information is done in the BR / POC phase.

8.7 Summary and Conclusions

While the combined model of the three levels of innovation management shows some gaps, it can be concluded that it gives a very broad, well connected view on innovation management on three different levels. An innovation can be a real breakthrough, but if the necessary technologies are not available and there is no real need for the product then the chance of an equal commercial success is minor. The two redlines, technology and the market, support this.

The interviews, held at Cordys, have shown that there is definitely resemblance between the theory and practice, but also show some differences, like the culture aspect. The technology part that was one of the theory red lines is clearly present within Cordys. The second theory red line: the market is also present but is becoming more important while Cordys is transforming from being technology driven to becoming more market driven.

The three levels are clearly visible within Cordys: their sprint model can be mapped to the Stage-Gate model from the product/project level. The BR/POC phase, with its involved management boards, is linked closely to the technology maps model from the company level. Factors from the last level, global/national level, return in the international oriented R&D of Cordys, but also in the joining of subsidized research groups like JACQUARD.

Cordys is in the process of changing from technology driven to market driven, and in this moment their strategy is maybe best described as customer driven. It is very important that they do not loose sight in this process and stay too much customer driven in stead of market driven. At this moment the (big) customers are the main driving force and the pressure to perform is very high. This pressure to perform can prohibit innovation management investments on short term, which include: people, time, and management capacity. However, short term urgency can really hurt long term benefits and efficiency. It is therefore very important to always keep track of the long term impact that decisions have.

While the difference in stakeholders are a possible explanation for the gaps in the integrated model, further research is necessary to check this and maybe come up with other explanations or even better: find a solution to fill the gaps and provide an even more smooth transition between the different levels. In further research also the reference customer should be taken into account, as these are not present in the researched models, and, according to the CTO of Cordys, are very important for a high-tech company like Cordys, also segmenting innovations is very important to make sure that the chance that an innovation becomes a commercial success is maximized.

References

[Albright 2003] Richard E. Albright and Thomas A. Kappel, "Roadmapping in the Corporation", Research Technology Management, March-April 2003

[BEAS 2006] BEA Systems, "Extending the Business Value of SOA Through Business Process Management", http://www.bpm.com/IndustryResearchRO.asp?IndustryResearchid=15

[Bitpipe n.d.] TechTarget library of white papers, http://www.bitpipe.com/tlist/Business-Process-Management.html

[ComplexEvents n.d.] David Luckham et. al, "Complex Event Processing", http://www.complexevents.com

[Cooper 1990] Robert G. Cooper, "Stage-Gate Systems: A New Tool for Managing New Producs", Business Horizons, May-June 1990

[Coopers 1985] Coopers & Lybrand Consulting Group, "Business Planning in the Eighties: The New Marketing Shape of North American Corporations".

[DataDirect n.d.] DataDirect Technologies, "Event-Driven Architectures (EDA)", http://www.neonsys.com/Solutions/event-driven_architecture.asp#

[Dunphy 1996] Steven M. Dunphy, Paul R. Herbig, and Mary E. Howes, "The Innovation Funnel], Elsevier Science Inc.

[Elemental 2006] Brenda Michelson, "Event-Driven Architecture Overview", Elemental Links, http://elementallinks.typepad.com/bmichelson/2006/02/eventdriven_arc.html

[Esper n.d.] Thomas Bernhardt, Esper, http://esper.codehaus.org

[Esper benchmark n.d.] Esper VWAP benchmark, http://www.espertech.com/news/20070814_performance.php

[EsperTech n.d.] Thomas Bernhardt (CEO), EsperTech, http://www.espertech.com

[Ettlie 1987] John E. Ettlie and Albert H. Rubenstein, "Firm Size and Product Innovation", Journal of Product Innovation and Management, June, 89-108 (1987).

[Hopkins 1980] D.S. Hopkins, "New Products Winners and Losers", The Conference Board, Report no. 773, New York, 1980.

[Luckham 2002] David Luckham, "The Power of Events – An Introduction to Complex Event Processing in Distributed Enterprise Systems", Addison-Wesley 2002

[Maccoby 1990] Michael Maccoby, "The American Character: The Organization Man", Current 326, 4-9 (1990).

[Mohr 2005] Jakki Mohr, Sanjit Sengupta, and Stanley Slater, "Marketing of High-Technology Producs and Innovations – second edition", "Technology Map", pg 204-211.

[Rothwell 1986] Roy Rothwell and Hans Wissemann, "Technology, Culture, and Public Policy", Technovation 4, 91-115 (1986).

[ruleCore n.d.] Marco Seiriö, MS Analog Software kb, ruleCore, http://www.rulecore.com

[SearchSMB 2006] SearchSMB, "BAM definition", http://searchsmb.techtarget.com/sDefinition/0,290660,sid44_gci1224512,00.html

[SearchWebServices 2006] SearchWebServices, "EAI definition", http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci213523,00.html

[Snoop 1993] Sharma Chakravarthy and Deepak Mishra, "Snoop: An Expressive Event Specification Language For Active Databases".

[SOAWorld 2004] SOA World Magazine "Web Services Integration Brokers and Enterprise Application Integration", SYS-CON Media, http://webservices.sys-con.com/read/45523.htm

[StreamCruncher n.d.] Ashwin Jayaprakash, StreamCruncher, http://www.streamcruncher.com

[Tidd 2005] Joe Tidd, John Bessant, and Keith Pavitt, "Managing Innovation – third edition", John Wiley & Sons 2005

[WebMethods 2006] webMethods, "BAM – the New Face of BPM", http://www1.webmethods.com/PDF/whitepapers/BAM-The_New_Face_of_BPM.pdf

List of Acronyms

BAM	Business Activity Monitoring
BI	Business intelligence
BPM	Business Process Management
BPMG	Business Process Management Group
CEP	Complex Event Processing
CRM	Customer Relationship Management
DSL	Domain Specific Language
EDA	Event-Driven Architecture
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ESP	Event Stream Processing
EQL	Event Query Language
FDA	Food and Drug Administration
IT	Information Technology
PUB/SUB	Publish / Subscribe
SCM	Supply Chain Management
SEDA	Staged Event-Driven Architecture
SOA	Service-Oriented Architecture
SQL	Structured Query Language
VWAP	Volume Weighted Average (Price)
XML	Extensible Markup Language

Appendix

A. Research results six CEP cases.

Case 1:

Esper

```
every a=orderPackaged -> (orderCancelled(orderID=a.orderID) and
newOrder(productID=a.productID))"
```

StreamCruncher

```
select newOrderStream.orderId, orderPackStream.orderId,
        orderPackStream.productId
from orderPack (partition store last 40 minutes) as orderPackStream,
        orderCancel (partition store last 40 minutes) as orderCancelStream,
        newOrder (partition store latest 1000) as newOrderStream
where newOrderStream.$row_status is new
        and orderPackStream.$row_status is not dead
        and orderCancelStream.$row_status is not dead
        and newOrderStream.productId = orderPackStream.productId
        and orderPackStream.orderId = orderCancelStream.orderId;
```

ruleCore

```
<view>
  <match>
    <value>
      <event>$xpath("event-def[@eventType="orderPackaged"]")</event>
      <field>$xpath("EventBody/orderID")</field>
    </value>
    <value>
      <event>$xpath("event-def[@eventType="orderCancelled"]")</event>
      <field>$xpath("EventBody/orderID")</field>
    </value>
  </match>
  <match>
    <value>
      <event>$xpath("event-def[@eventType="newOrder"]")</event>
      <field>$xpath("EventBody/productID")</field>
    </value>
    <value>
      <event>$xpath("event-def[@eventType="orderPackaged"]")</event>
```

```
        <field>$xpath("EventBody/productID")</field>
      </value>
    </match>
</view>
<detector>
    <and>
      <event-pickup>$xpath("view/event[@type="orderPackaged"]")</event-pickup>
      <event-pickup>$xpath("view/event[@type="orderCancelled"]")</event-pickup>
      <event-pickup>$xpath("view/event[@type="newOrder"]")</event-pickup>
    </and>
</detector>
```

Case 2:

Esper

"every a=orderPackaged(totalPrice>5000) -> not orderApproved (orderID=a.orderID)"

StreamCruncher

```
select unapprovedOrderId, price
from
   alert orderPackStream.orderId as unapprovedOrderId,
         orderPackStream.totalPrice as price
   using orderPack (partition store latest 1000 where totalPrice > 5000) as
         orderPackStream correlate on orderId, orderApprove (partition store last 40
         minutes) as orderApproveStream correlate on orderId
   when present(orderPackStream and not orderApproveStream);
```

ruleCore

```
<view>
   <assert>
      <event>$xpath("event-def[@eventType="orderPackaged"]")</event>
      <expression>$xpath("EventBody/totalPrice > 5000")</expression>
   </assert>
   <match>
      <value>
         <event>$xpath("event-def[@eventType="orderPackaged"]")</event>
         <field>$xpath("EventBody/orderID")</field>
      </value>
      <value>
```

```
        <event>$xpath("event-def[@eventType="orderApproved"]")</event>
        <field>$xpath("EventBody/orderID")</field>
      </value>
    </match>
</view>
<detector>
    <and>
      <event-pickup>$xpath("view/event[@type="orderPackaged"]")</event-pickup>
      <not>
        <event-pickup>$xpath("view/event[@type="orderApproved"]")</event-
                                                                    pickup>
      </not>
    </and>
</detector>
```

Case 3:

Esper

"every a=cashWithdrawal ->
(cashWithdrawal(bankcardNumber=a.bankcardNumber,
accountNumber=a.accountNumber, cityID!=a.cityID) where timer:within(120 sec))"

StreamCruncher

```
select cashWithdrawalStreamX.accountNumber,
   cashWithdrawalStreamX.cardNumber, cashWithdrawalStreamX.cityId as City1,
   cashWithdrawalStreamX.cityId as City2
from cashWithdrawal (partition by accountNumber store 120 seconds) as
   cashWithdrawalStreamX, self#cashWithdrawalStreamX as
   cashWithdrawalStreamY
where cashWithdrawalStreamX.$row_status is new
   and cashWithdrawalStreamY.$row_status is not dead
   and cashWithdrawalStreamX.accountNumber =
         cashWithdrawalStreamY.accountNumber
   and cashWithdrawalStreamX.cardNumber =
          cashWithdrawalStreamY.cardNumber
   and cashWithdrawalStreamX.cityId != cashWithdrawalStreamY.cityId;
```

ruleCore

```
<view>
   <age>
      <max>0000-00-00-00-02</max>
```

```
  </age>
  <match>
    <value>
      <event>$xpath("event-def[@eventType="cashWithdrawal"]")</event>
      <field>$xpath("EventBody/bankcardNumber")</field>
      <field>$xpath("EventBody/accountNumber")</field>
    </value>
    <value>
      <event>$xpath("event-def[@eventType="cashWithdrawal"]")</event>
      <field>$xpath("EventBody/bankcardNumber")</field>
      <field>$xpath("EventBody/accountNumber")</field>
    </value>
  </match>
</view>
<detector>
  <and>
    <event-pickup>$xpath("view/event[@type="cashWithdrawal"]")</event-pickup>
    <event-pickup>$xpath("view/event[@type="cashWithdrawal"]")</event-pickup>
  </and>
</detector>
```

Case 4:

Esper

"every a=sellStock(amount>10000) -> (stockPriceChange(stockID=a.stockID, priceChangePercentage < -20) where timer:within(7 day))"

StreamCruncher

```
select firstSellStream.sellerId, firstSellStream.stockId, firstSellStream.amount,
  priceChangeStream.priceChange
from stockSell (partition by stockId store last 168 hours where amount > 10000) as
  firstSellStream, stockPriceChange (partition by stockId store latest 100 where
  priceChange < -20) as priceChangeStream
where priceChangeStream.$row_status is new
  /* This means that the Event is not New, but not Dead either, i.e Event
  arrived in an older cycle - before a Price change Event. */
  and firstSellStream.$row_status is not dead
  and not(firstSellStream.$row_status is new)
  and firstSellStream.stockId = priceChangeStream.stockId;
```

ruleCore

```
<view>
  <age>
    <max>0000-00-07</max>
  </age>
  <assert>
    <event>$xpath("event-def[@eventType="sellStock"]")</event>
    <expression>$xpath("EventBody/amount > 10000")</expression>
  </assert>
  <assert>
    <event>$xpath("event-def[@eventType="stockPriceChange"]")</event>
    <expression>$xpath("EventBody/ priceChangePercentage <
                                                    -20")</expression>
  </assert>
  <match>
    <value>
      <event>$xpath("event-def[@eventType="sellStock"]")</event>
      <field>$xpath("EventBody/stockID")</field>
    </value>
    <value>
      <event>$xpath("event-def[@eventType="stockPriceChange"]")</event>
      <field>$xpath("EventBody/stockID")</field>
    </value>
  </match>
</view>
<detector>
  <sequence>
    <event-pickup>$xpath("view/event[@type="sellStock"]")</event-pickup>
    <event-pickup>$xpath("view/event[@type="stockPriceChange"]")</event-
                                                              pickup>
  </sequence>
</detector>
```

Case 5:

Esper

"every a=newOrder -> (timer:interval(3 hour) and not orderPackaged(orderID=a.orderID))"

StreamCruncher

select missedOrderId

```
from
   alert newOrderStream.orderId as missedOrderId
   using newOrder (partition store last 3 hours) as newOrderStream correlate on
      orderId, orderPack (partition store latest 100) as orderPackStream correlate on
      orderId
   when present(newOrderStream and not orderPackStream);
```

ruleCore

```
<view>
   <age>
      <max>0000-00-00-03</max>
   </age>
   <match>
      <value>
         <event>$xpath("event-def[@eventType="newOrder"]")</event>
         <field>$xpath("EventBody/orderID")</field>
      </value>
      <value>
         <event>$xpath("event-def[@eventType="orderPackaged"]")</event>
         <field>$xpath("EventBody/orderID")</field>
      </value>
   </match>
</view>
<detector>
   <and>
      <event-pickup>$xpath("view/event[@type="newOrder"]")</event-pickup>
      <not>
         <event-pickup>$xpath("view/event[@type="orderPackaged"]")</event-
                                                                    pickup>
      </not>
   </and>
</detector>
```

Case 6:

Esper

"every a=serviceCalled -> ((serviceCalled(customerID=a.customerID) where timer:within(14 day)) -> (timer:interval(14 day) and not newOrder(customerID=a.customerID)))"

StreamCruncher

```
select customerId, serviceCallsMade
from serviceCalled (partition by customerId store last 336 hours with
   count(customerId) as serviceCallsMade) to (partition store last 336 hours where
   $row_status is new and serviceCallsMade >= 2) as serviceCalledStream
where serviceCalledStream.$row_status is dead
   and not exists (
      select customerId
      from newOrder (partition by customerId store last 336 hours) as
         newOrderStream
      where newOrderStream.$row_status is not dead
         and newOrderStream.customerId = serviceCalledStream.customerId
   );
```

ruleCore

```
<view>
   <age>
      <max>0000-00-14</max>
   </age>
   <match>
      <value>
         <event>$xpath("event-def[@eventType="serviceCalled"]")</event>
         <field>$xpath("EventBody/customerID")</field>
      </value>
      <value>
         <event>$xpath("event-def[@eventType="serviceCalled"]")</event>
         <field>$xpath("EventBody/customerID")</field>
      </value>
   </match>
</view>
```

<detector **emit "customerComplainingALot" event**>

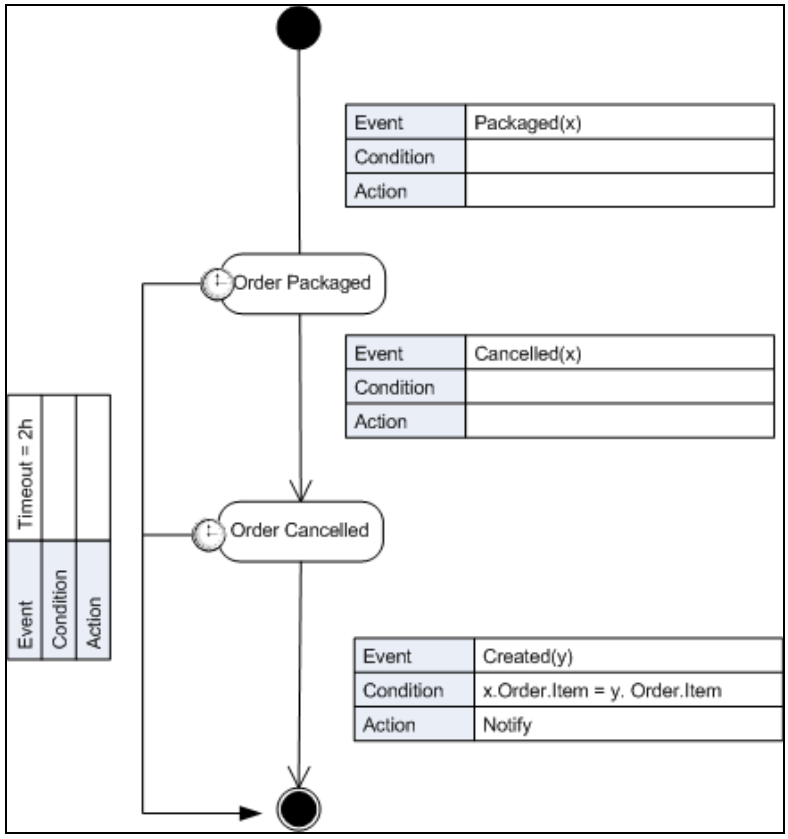
```
   <and>
      <event-pickup>$xpath("view/event[@type="serviceCalled"]")</event-pickup>
      <event-pickup>$xpath("view/event[@type="serviceCalled"]")</event-pickup>
   </and>
</detector>

<view>
   <age>
      <max>0000-00-14</max>
```

```
  </age>
  <match>
    <value>
      <event>$xpath("event-
                    def[@eventType="customerComplainingALot"]")</event>
      <field>$xpath("EventBody/customerID")</field>
    </value>
    <value>
      <event>$xpath("event-def[@eventType="newOrder"]")</event>
      <field>$xpath("EventBody/customerID")</field>
    </value>
  </match>
</view>
<detector>
  <and>
    <eventpickup>$xpath("view/event[@type=
                                  "customerComplainingALot"]")</event-pickup>
    <not>
      <event-pickup>$xpath("view/event[@type=
                                              "newOrder"]")</event-pickup>
    </not>
  </and>
</detector>
```

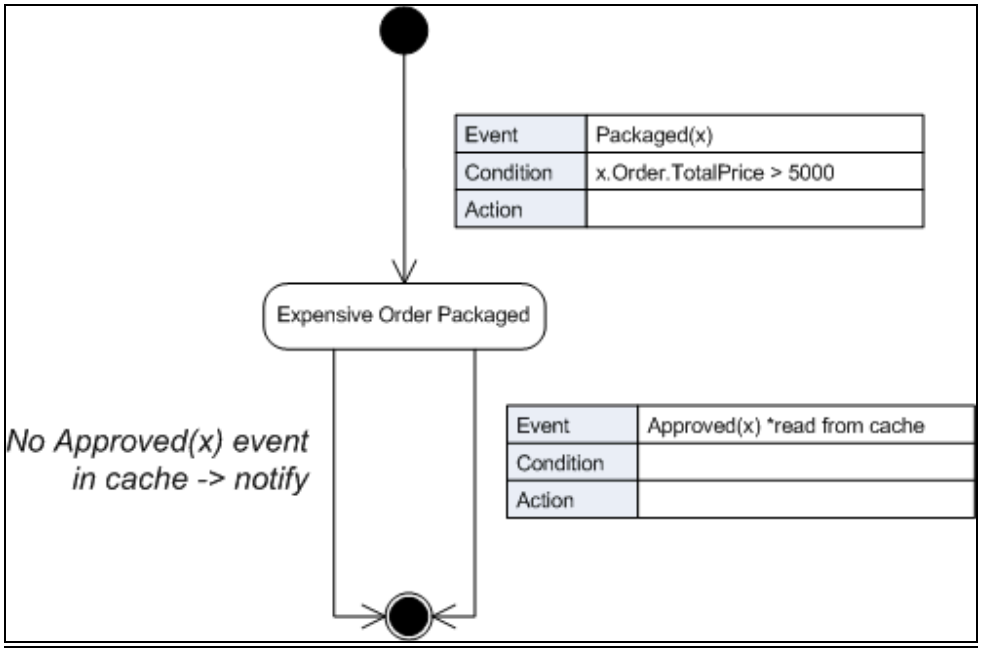
B. Research results six CEP cases.

Erik van de Ven, Senior Architect at Cordys, created non formal standard ECA (Event Condition Action) like diagrams for the six CEP cases.

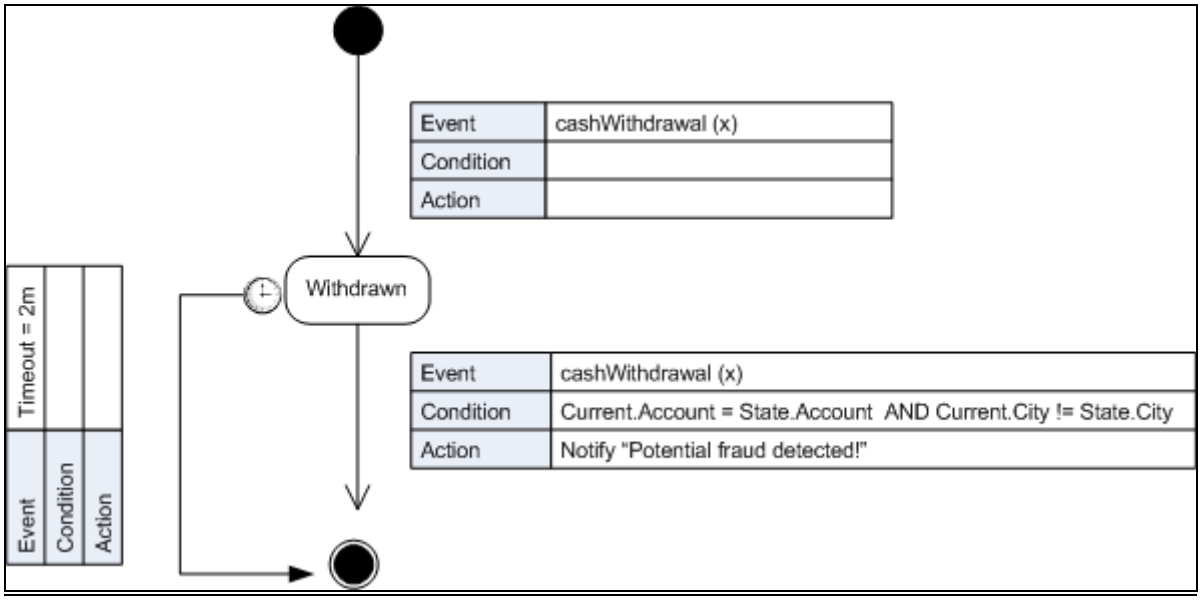
Case 1:


Event
Packaged(x)
Condition
Action
Order Packaged
Event
Cancelled(x)
Condition
Action
Timeout = 2h
Event
Condition
Action
Order Cancelled
Event
Created(y)
Condition
x.Order.Item = y. Order.Item
Action
Notify

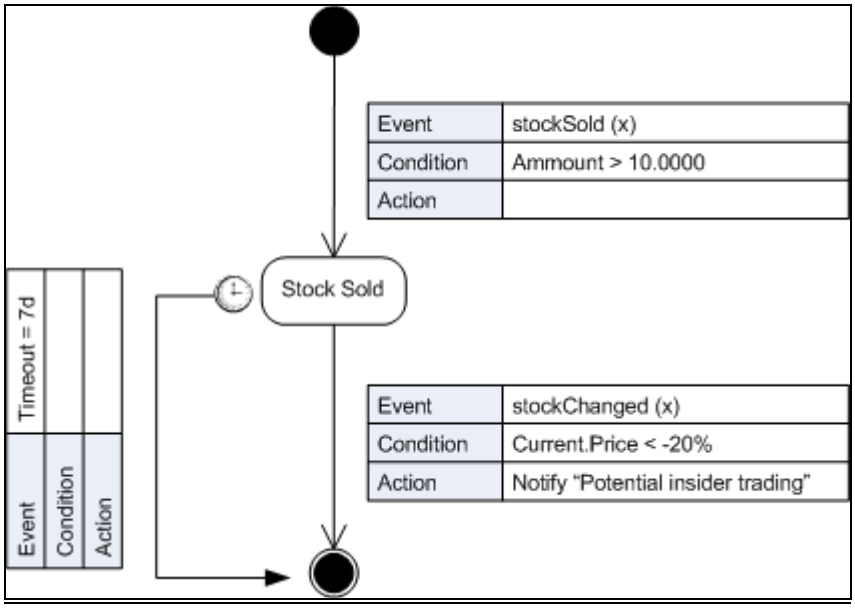
Case 2:


Event
Packaged(x)
Condition
x.Order.TotalPrice > 5000
Action
Expensive Order Packaged
No Approved(x) event
in cache -> notify
Event
Approved(x) *read from cache
Condition
Action

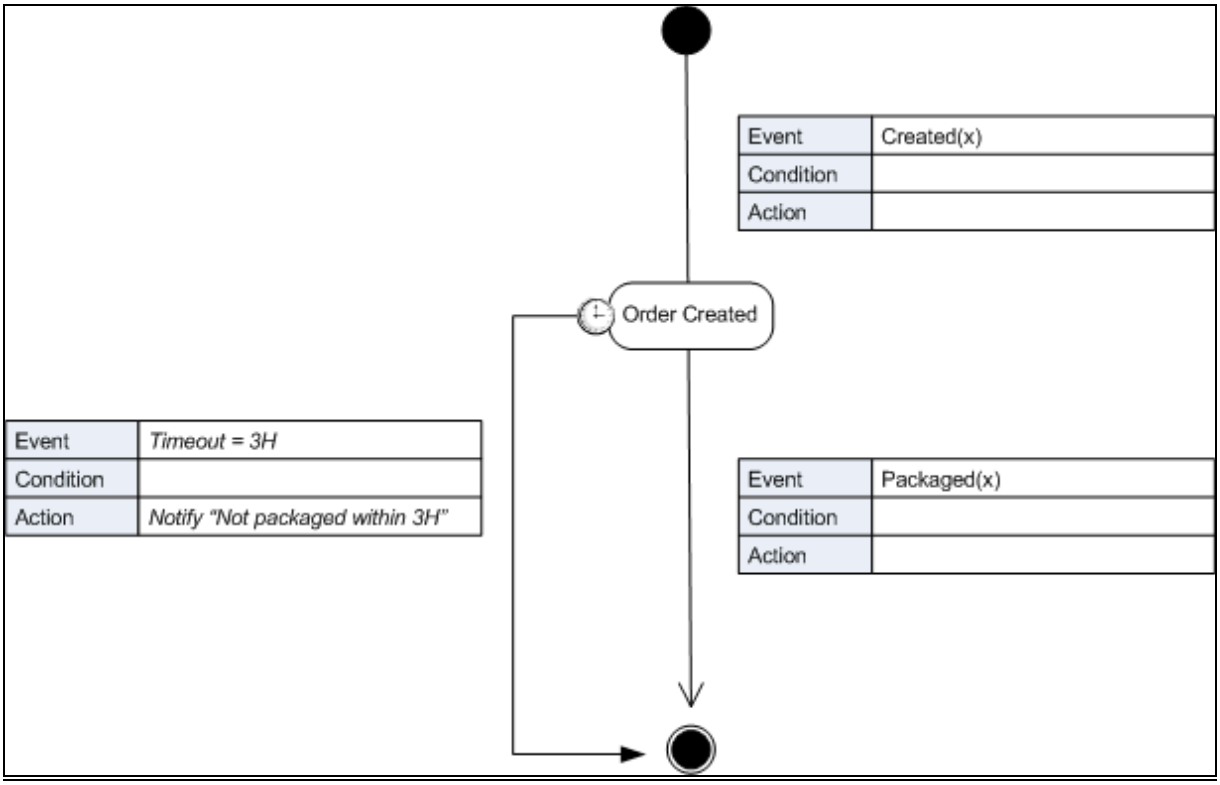
Case 3:


Event
cashWithdrawal (x)
Condition
Action
Withdrawn
Timeout = 2m
Event
Condition
Action
Event
cashWithdrawal (x)
Condition
Current.Account = State.Account AND Current.City != State.City
Action
Notify "Potential fraud detected!"

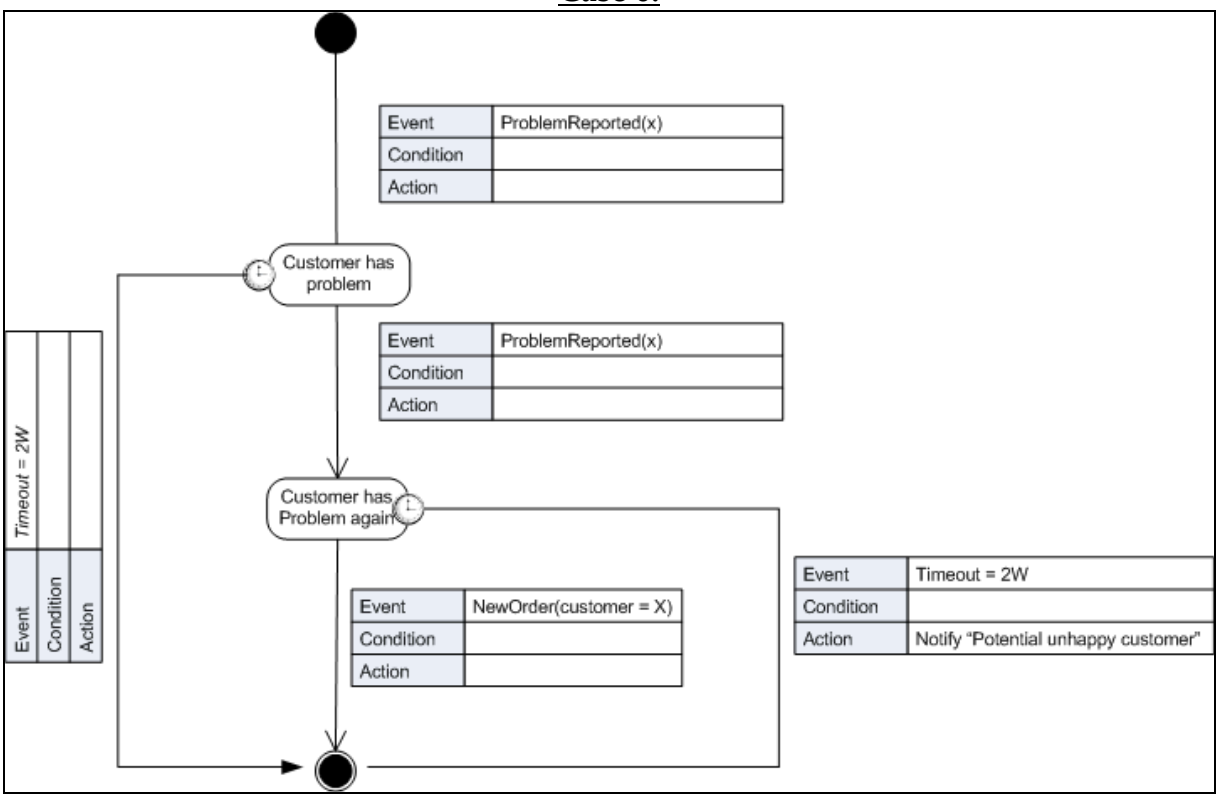
Case 4:


Event
stockSold (x)
Condition
Ammount > 10.0000
Action
Stock Sold
Timeout = 7d
Event
Condition
Action
Event
stockChanged (x)
Condition
Current.Price < -20%
Action
Notify "Potential insider trading"

Case 5:


Event | Created(x)
Condition
Action
Order Created
Event | Timeout = 3H
Condition
Action | Notify "Not packaged within 3H"
Event | Packaged(x)
Condition
Action

Case 6:


Event | ProblemReported(x)
Condition
Action
Customer has problem
Event | ProblemReported(x)
Condition
Action
Timeout = 2W
Event
Condition
Action
Customer has Problem again
Event | NewOrder(customer = X)
Condition
Action
Event | Timeout = 2W
Condition
Action | Notify "Potential unhappy customer"

C. Interview questions

The focus of this interview is starting at the global/national level and then moves down, toward the company level and finally to the product/project level. First a short introduction about innovation management is provided.

Global/national level:

- Which factors can play a role on global/national level, and why?
- Are there any concrete examples for Cordys, concerning these factors?
- Does Cordys also has partners that are used to obtain a better market share?
- What is the role of the government on this level?

Company level:

- Which factors can play a role on company level, and why?
- How is decided which innovations are important to Cordys?
- Is Cordys more technology or market driven, and why?
- What is your view on the relation between the size of a company and its innovativeness?

Product/project level:

- Which phases are known within an innovative project, and what happens roughly in these phases?
- How is decided which innovations are implemented and who decides that?
- Which decisions are made while a innovations progresses through the different phases?