FInest – **F**uture **In**ternet **e**nabled optimi**s**ation of **t**ransport and logistics networks

D6.1

Requirements Analysis and Selection of Technology Baseline for Event Processing Component

Project Acronym	FInest	
Project Title	Future Internet enabled optimisation of transport and logistics networks	
Project Number	285598	
Workpackage	#WP6 Proactive Event-Driven Monitoring	
Lead Beneficiary	IBM	
Editor(s)	Yagil Engel	IBM
Contributors	Guy Sharon	IBM
	Fabiana Fournier	IBM
	Åsmund Tjora	MRTK
	Michael Zahlmann	KN
	Tor Thunem Knutsen	ARH
	Evert-Jan Van Harten	AFKL
Reviewer	Metin Turkay	KOC
Reviewer	Michael Stollberg	SAP

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 285598

Dissemination Level	PU
Contractual Delivery Date	30.9.2011
Actual Delivery Date	30.9.2011
Version 1.0	

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 285598

Abstract

This report presents the first Deliverable of work package WP6 "Proactive Event Driven Monitoring". Work Package 6 is responsible for the Event Processing Module (EPM), one of the four core technical components of FInest. The role of this module within FInest is to provide event-driven monitoring across domain and transport modality, with three goals: obtaining end-to-end visibility of logistics processes, monitoring the achievement of contract terms, and triggering replanning of the scenario when needed.

The document describes initial results of the "requirements analysis and selection of technology baseline for the event processing component" (T6.1); the task includes: identifying the relevant business requirements provided by domain experts, an initial conceptual architecture which is stating the boundaries of the event processing component within the overall archietecture of FInest, and a functional and technical requirements anaylsis. In addition, the task includes a state of the art analysis of event driven technology from which a technology baseline is selected.

The document refers to additional work done in this deliverable: the initial identification of generic enablers, which is part of T6.3, "Technological Alignment with FI PPP Core Platform". The actual results of this work are provided at D3.1 along with the rest of the project.

This document has two main facets: functional and technical. The functional aspect, as described above, deals with the functionality provided by the EPM within Finest. The document provides a functional overview of the EPM, review of some existing software products that provide event-driven monitoring functionality for logistics processes, and mainly a list of initial functional requirements of the EPM.

The technical aspect is the event-driven technology that will be used to provide this functionality. The document provides an overview of what event-driven technology is, a list of requirements from a generic event processing technical component, and an assessment of existing systems that provide generic event processing functionality, including a selection of technology baseline for FInest EPM.

Document History

Version	Date	Comments
1.0	30 Sept. 2011	Released

Table of Contents

List of Tables

List of Figures

Acronyms

Acronym	Explanation
BCM	Business Collaboration Module
ECM	E-Contracting Module
EPM	Event Processing Module
KPI	Key Performance Indicator
SLA	Service Level Agreement
AIS	Automatic Identification System
IFP	Information Flow Processing
TPM	Transport Planning Module
TP	Transport Plan
CDMP	Cargo Data Management Portal
AMIT	Active Middleware Technology
DSMS	Data Stream Management System
CEP	Complex Event Processing

1. Introduction and Functional Overview

Logistics Processes are complex and affected by many external factors. Whereas a logistic freight scenario requires careful and optimal planning, the resulting plan is often subject to change as a result of unplanned events that occur during the execution of the plan. The ability to monitor these events and respond to them as promptly as possible is essential for maintaining successful and efficient supply chain.

1.1. Business Background

The need for intelligent event-driven monitoring stems primarily from the inherent complexity of the domain. The main challenge is to provide end-to-end visibility of a shipment, and facilitate instantaneous response to exceptions.

1.1.1. End-to-end Shipment Monitoring

A shipment is typically transported with various transportation modes and transferred between several carriers. In a typical scenario, the shipment is picked up at a customer's location, taken by truck to airport, seaport, or freight train warehouse, where it is handed over to a different carrier (the air, sea, or rail carrier) that is responsible for the shipment until handed over to another trucking company at a different port or station. Finally, the shipment is trucked over to its destination.

This highlights the need for the involved parties to have end-to-end visibility over the shipment process, and get constant updates with respect to its status. For example, a fish delivery from Norway to Holland might go awry (for example, due to time delay, or faulty refrigeration conditions of the containers); if the source of the delivery has visibility into the process in real time, the source can quickly get ready to send a replacement. In addition, it lets the end receiver of the shipment to prepare for the potential lack of supply. Furthermore, the visibility can point out the reason for the damage and allow players to protect themselves from unjustified responsibility. As another example, consider a forwarder who ships critical parts from Maastricht to Barcelona, through a flight from Brussels; an event indicating severe traffic congestion on the way to Brussels' airport, and as a result an indication that the shipment might not reach the airport in time for the flight, will allow the forwarder to find an alternative route (for example a flight from Dusseldorf) on time. The main section of requirements (4.1.3) deals with end-to-end monitoring and detection of unexpected events in a multi-modal transportation scenario.

1.1.2. Air and Rail Cargo Issues

In addition to the multi-modal, multi-player nature of the domain, there are many complexities that are particular to air, sea, and rail cargo transport. The air cargo industry adopted the Cargo 2000 standard [11], which provides a summary of the stages of a cargo process by defining seven milestones (some of which might repeat several times) from which a freight scenario consists. The milestones are:

- Receive electronic airway bill (AWB)
- Receive shipment at origin

- Shipment departure (by truck or flight) from (origin or transit) station
- Shipment arrival to next (transit or final) station
- Shipment received at (transit of final) station
- Customer pickup notification
- Delivery to customer

Note that three of the milestones are repeated for different segments of the transport chain: departure, arrival, and receipt at station.

In addition to setting the milestones, the standard covers the possible exceptions that might occur during the shipment scenario. These are divided to several groups:

- Booking discrepancies : difference between the details of the order and the actual shipment
- Documentation issues: missing documents, wrong documents
- Government or authority issues: customs, security
- Physical issue with the shipment: loading and unloading of freight, temperature control, issues with warehouse equipment, etc.
- Commercial issues: overbooking, change for optimization reasons, capacity reduction.
- Flight or truck operation: late flight, late truck arrival to flight, etc…
- IT failure: message loss
- Force majeure: weather, terror alert, etc.

These exceptions come from different sources; some can be received from various electronic systems, and some must be entered manually. It is crucial that these events will be processed by a single logical component that is capable of analyzing their combined effect and forward it in order to enable the various players to respond in time, and possibly to generate a new shipping plan.

The issues with freight rail are similar in nature; the milestones above can be adapted with minor changes for monitoring a rail cargo scenario. Furthermore, most of the exceptions (booking discrepancies, customs, commercial issues, operation, IT, force majeure…) can occur in any modality. Section 4.1.4 covers requirements that involve handing a shipment to a freight carrier, and are not mentioned in the shipment monitoring section. In the case of sea freight there are additional requirements that rise from the complex operation of a sea port, and the high dependency of the freight vessels on the services of the port. These are described below and covered in Section 4.1.5.

1.1.3. Sea Freight Issues

The central scenario that is handled with respect to sea freight is a *vessel call*, a term referring to the arrival and departure of a vessel to / from a port. A smooth vessel call scenario requires a series of timely operations, mainly of booking various resources in the port (pilot, quay, tugboat, mooring crew, ground equipment, cargo workers, electric and water supply, storage space); it is also highly dependent on weather conditions. Moreover, the operation also depends on various unanticipated port events (overbooking, the unannounced arrival of other ships, etc.). Failure in any of these aspects may cause a delay in the vessel call completion, which entails a delay in the particular shipment scenario of each container on the vessel.

The port authority tracks the scenario progress through its booking systems and with the help of the marine tracking system called AIS (Automatic Identification System). A failure or difficulty in one step of one vessel call may affect the vessel's schedule, and sometimes the vessel call process of other vessels. For example, weather conditions can delay the vessel's entrance to the port, respectively delaying the use of its booked resources, and as a result delaying the use of these resources by the next vessels. An effective handling of such events can help to mitigate these changes: port resource usage can be optimized with respect to new constraints, relevant parties can be notified on delay of shipment arrival, and in extreme situations vessels can be rerouted between ports.

1.2. EPM Functional Overview

The functionality of the EPM can be described at three levels.

1. On the surface level, event processing provides visibility into the current status of the logistics process: the location of a shipment, whether it is on a carrier or in a warehouse, whether or not it was customs-cleared, etc. This means that EPM collects tracking events sent by various systems employed for different transportation modes, and provide this information to user interface.
2. Beyond the functionality of merely track-and-notify, event-processing employs rules that encapsulate specific logic applied on events. The basic functionality of rules is to indicate whether or not the logistics process progresses as it should, or whether something has gone wrong. This is done by constantly keeping track of planned route (according to TP), notifying the BCM and / or front-end when a plan seems to fail, or when a requirement specified in the transport plan might be in breach.
3. At a deeper level, events potentially provide insights regarding parts of the scenario that have not yet been reached; for example, stormy weather near a seaport may indicate that a ship carrying the managed containers will be delayed in entering the port. Security alerts at an airport may imply flight delays. Detecting those events relevant to the scenario at an early stage allows the system to respond to events *before* they occur, and thus to surface *proactive event-driven computing* functionality [4].

Event processing engines detect specific situations by identification of event patterns in real-time data, and react to these situations according to predefined rules. These rules will be defined for specific deployments and for specific scenarios. The EPM will provide a parameterized set of rules from which users will be able to configure the engine for their needs.

The EPM interact with BCM as follows: BCM send EPM the relevant parameters from the TP. These includes specific locations, times, and shipment conditions. It also indicates the status changes and conditions under which the BCM or front-end should be notified. All these are translated to parameter values to the event processing rules.

The functionality above is specified by the functional requirements in this document, according to the following grouping:

1. Availability of Events: many of the events required for the above functionality are external, and it is necessary to ensure that they become available to the FInest system, and in format the the EPM can read.
2. Configurability: a set of predefined parameterized rules from which users can select.
3. Shipment Monitoring: according to the three levels above: end-to-end visibility, identification of deviations, and detection of possible future deviations.

In addition, two scenarios are isolated in order to allow their particular requirements to surface:

4. Cargo Carrier Monitoring: monitoring of a particular cargo carrier (mainly rail and air; sea vessels are covered in the next group).
5. Vessel Call: the process of a vessel arrival to port, cargo handling and departure.

2. Initial Conceptual Architecture of the EPM Component within Finest

In this section we overview the conceptual architecture of the EPM and its role within FInest. First, we provide high level background on the conceptual model of event processing; next we discuss the role of the component within FInest, and finally lay out an initial conceptual architecture for the EPM.

2.1. The conceptual model of event processing

Event driven architectures support applications that are reactive in nature, ones in which processing is triggered in response to events; this is in contrast to traditional responsive applications, in which processing is in response to an explicit request [5]. Furthermore, reactions are often based on situations identified by more than one event over periods of time and therefore processing events and inferring situations is required capability in an event-driven architecture. Complex Event Processing (CEP) is such a capability, often used in scenarios of environment monitoring, business process automation, and control systems [3]; it has two main characteristics:

(i) Situation Detection: The role of the system is the detection of complex events in streams of incoming raw events; complex events are characterized as *patterns* of atomic events. The semantics of a pattern represents a particular situation that is of interest to the system.

(ii) Decoupling: An event source, or *producer*, does not depend on a particular processing or course of action being taken by an event *consumer* (the *sink*, or the entity that is notified on events or situation detected). Moreover, an event consumer does not depend on processing performed by the producer other than the production of the event itself [5].

Therefore, the CEP system interacts with a large number of decoupled and heterogeneous sources and sinks, while responsible to detect a large number of different situations. This suggests a distributed architecture, organized as a set of *processing units*, connected in an overlay network, called the *event-processing network* [3,5], which executes particular forwarding and routing policies between the units. Each processing unit is responsible for a particular operation; the detection of an event pattern (usually some sequence or collection of raw events) or a particular aggregation operation over its incoming events (such as the summation over the value of a particular event attribute). These operations must be given a specific semantic context: what is the *time window* in which a pattern needs to appear (in order for it to indicate the situation occurred) or in which a value is aggregated, and whether the pattern or aggregation is done according to a particular segmentation of the incoming events (for example, by geographic regions, or by ID of a particular entity), or according to indicators in the state of the system (*state variables*) [7].

2.2. Role in FInest

The architecture of FInest is comprised of three main technical components in addition to the EPM. The *Transport Planning Module (TPM)* allows end users to construct a *Transport Plan (TP)* involving various carriers; the TP includes the information that will allow the EPM to track the execution: what are the transport segments, ETAs, and various plan requirements (when to report status changes, which conditions of the shipment to monitor, etc.). The *Business Collaboration Manager (BCM)* introduces secure end-to-end networks between transport and logistics partners. This module interacts with the EPM during the scenario execution: the BCM provides the EPM with all the relevant information from the TP, and the EPM reports the BCM on status changes and when various flags are raised (according to the requirements indicated in the TP). The third module is *Electronic Contract Manager (ECM)*, which will facilitate the negotiation and contracting with logistic partners towards the creation of a transport chain; the ECM interacts with the TPM in the *preparation phase* of the logistic process, whereas the EPM and BCM operate in the *execution phase*. The initial requirement analysis for the BCM, TPM, and ECM are provided in deliverables D5.1, D7.1, and D8.1 respectively.

2.3. Initial Conceptual Architecture of EPM

The Event Processing Module (EPM) includes the following components, which are identified in Figure 1:

- Rules Definition – Allowing the design of a set of rules, according to the requirements of which situations are needed to be monitored or detected. In FInest, there is a global parameterized set of rules. Rules can either be permanent (for any scenario) or instantiated by the user for a specific scenario. Specific requirements of a transport plan set the rules parameters (e.g., ETA for individual transport legs). Rule instances are employed by the runtime engine.
- Events Sources – Sources of events can be various related systems (in FInest examples for such sources include AIS, Airport Systems, and existing tracking systems employed by freight forwarders). Events emitted from sources are processed by the runtime engine.
- Runtime Engine – The run-time engine is a collection of processing units, each executes a rule or a set of rules to determine situations and to send status updates.
- Detected Situations – When rules are triggered by patterns of input events, they indicate particular situations. In FInest, the results of event processing can be directed either to the human interface, and / or to BCM, in order to allow the user to respond in a timely manner.

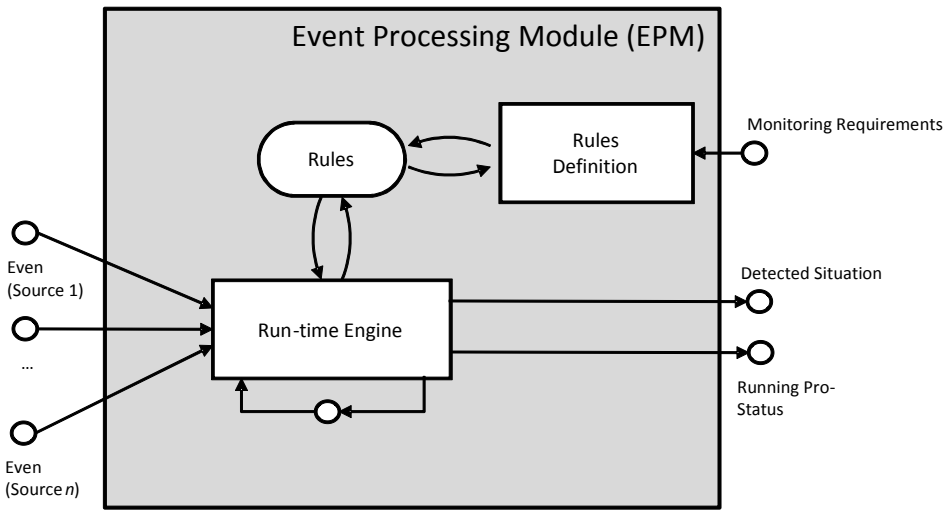

Figure 1: Event Processing Module Conceptual Architecture

This flexible rule based structure is suitable to support the business needs raised in Section 1. First, it supports the basic functionality of detecting combinations of incoming events to track the current status of the shipment. Further, it detects the absence of events to realize that a shipment is late. Next, rules can be added for any requirement posed by the TP; for example, if the plan requires that stakeholders are notified when the anticipated delay is above a specific time frame, a rule can indicate that based on its calculated ETA. Technically, some processing units update the ETA according to events such as status change or lack of arrival, and another processing unit will compare the ETA with the original schedule and will emit an event to BCM when the difference exceeds the specified time frame.

To summarize, the technical requirements for the EPM are grouped as follows:

- Events and event sources: define a general high-level structure of events (*meta-data*), providing a basic level of uniformity to which producers and consumers adhere.
- Event processing network: rule-based processing units, overlay network, and routing policies that comprise the runtime engine.
- Operators: event patterns and functional aggregations that express the logic of rules.
- Event context: temporal windows, segmentations, and combination thereof; this determines the association of events to specific processing units.

3. Existing Systems for Event Monitoring in the Logistics Domain

The logistic freight industry is wide and diverse; its various players employ a wide range of tools, most of which are proprietary, specific to a segment of the industry or particular mode of transportation, or even specific to a particular player. The survey below covers three representatives, each one from a different segment, chosen due to their employment by FInest domain partners. The list is by no means exhaustive, but the conclusions (Section 3.2) are likely to be fairly general to the range of tools employed by the industry today.

3.1.1. AIS

The *Automatic Identification System (AIS)* [1] is a VHF-based system for broadcasting a vessel's information, such as the vessel's position and speed, activity, status, and identification number. The main purpose of the system is navigation safety. Often, the data received from the *AIS* system is plotted into electronic charts, giving a graphical overview of the traffic situation in the area.

According to the international convention for Safety Of Life At Sea (SOLAS), all ships above 500 gross tons, all ships above 300 gross tons engaged in international voyages, and all passenger ships shall be fitted with AIS. Many other vessels also carry AIS.

While the original intent of the system is navigation safety, the information broadcasted by the system is also used for other purposes, like traffic monitoring and control, tracking of vessels, improving estimates on the ship's time to arrival, signalling when the ship enters a specific area (e.g., near the port), and logging movements within a harbour area for various purposes (e.g., monitoring the use of berths).

As the system is based on terrestrial VHF, the system has a limited range (due to the Earth's curvature), and for good coverage in larger areas, an extensive reception infrastructure may be needed. For the same reason, land-based reception is only possible when the ships are relatively close to the coast. Recently, there also have been tests with satellite-based reception of AIS signals [2]; satellite reception will enable a much wider coverage.

AIS provides automated tracking information, which are then entered manually into other port control systems. FInest EPM is not expected to replace AIS, but rather provide a sophisticated outlet for the tracking information AIS emits. Currently those events are monitored manually; with FInest, AIS events may be employed for vessel call monitoring (Section 3.1.3) and for shipment monitoring (Section 3.1.2).

3.1.2. Traxon CDMP

Traxon's Cargo Data Management Portal (CDMP) is a web-based application and an intelligent tool for the performance management of air cargo processes. CDMP is currently used by the air cargo industry (though can potentially be applied to other transport modes) and enables airlines to measure and improve performance in handling consignments by providing two main functionalities:

Process visibility: indicating consistency of consignment and status between logistic partners. This is done through real-time web-based reporting; Traxon CDMP collates and evaluates performance data collected along the air freight transport chain, so that airlines receive accurate information and identify deviations from planned service level.

Performance measurement: comparison of real-time data with KPIs. Traxon compares planned route with actual events and identifies deviations, and creates exception record for each deviation. The aim is to reduce mistakes and dwell time on the ground.

Traxon provides air cargo companies the two main functionalities that are required from the EPM at FInest: visibility and performance tracking. However, it has several disadvantages: first, its application is currently limited to the air freight industry. Furthermore, most of the events it tracks are entered manually by system operators; at FInest, the intention is to use Future Internet infrastructure to get automatic event data. Additional drawbacks are mentioned in Section 3.2.

3.1.3. KN Login

The *KN Login* tool is a web based, open, proprietary system developed by Kuehne + Nagel to provide both multi-level planning and visibility functionality relating to the movements of shipments between a consignor and a consignee. Accessible by using any of the leading internet web browsers, KN Login is both easy to use and always up-to-date being fully integrated with Kuehne + Nagel's underlying suite of applications containing Warehouse, Distribution and Freight Management systems (*Ciel)*. These base applications provide real-time, event driven shipment updates as well as full end of day bulk updates.

Using KN Login each client can search, view and then assess information relating to the status, contents and progress of their shipments. The tool may be used simply as the basis for general shipment status and progress reporting or as part of a more structured, planning function within which the shipment timings and modes of delivery can be pre-planned according to expected lead-times and shipment details.

KN Login allows users to drilldown to individual details per shipment, which cover:

- Progress details, such as Route, Location and associated Date/Time.
- Shipping details relating to the Bill of Lading and product descriptions.
- Address details of the various Shipping and Receiving agents or Notify parties along the route.
- Customer references for the shipment.
- Associated documents, such as the Bill of Lading, attached as a series of images.

In support of the shipment visibility function, clients may run their own specific analyses and data extracts. These include a consolidated dashboard view, which shows the total number of shipments In Transit by location, the number of orders by mode by Vendor or Agent and a shipment forecast again by mode, but showing the number of shipments expected over the next day, 2 days, 3 days etc. A performance function similarly allows the user to track individual shipment progress against the expected route milestones and dates.

KN Login provides extensive visibility into an end-to-end multi-modal shipment process, with sophisticated search and drill-down capabilities. However, it lacks most of the other type of functionality required from EPM: measurement and comparison of online data with contract terms and KPIs.

3.2. Summary: Disadvantages of Current Systems

This is a summary of disadvantages of current systems in comparison to the planned event-driven functionality of FInest; some of the items are mentioned for specific tools at their respective sections, and some are additional and common to all.

- Most tools (KN Login is the exception) are limited to either a specific mode of transportation (sea or air), and are not capable of combining sources of data
- Most tools (Traxon is the exception) do not provide explicit comparison of online tracking information with: (i) planned route, and (ii) SLA KPIs, along with the ability to detect current and possible future deviations.
- FInest is a project that is designed to provide a comprehensive solution for the control and automation of logistic freight and supply chain processes; it therefore integrates the

event monitoring functionality with other aspects of the process, namely collaboration, contracting, and replanning. This integration is not available for current systems.

- Current systems are limited in the kind of input they handle; FInest EPM will be designed to allow dynamic definition of event sources and event types, and hence better allow exploiting Future Internet infrastructure.
- Whereas current systems are specific to the domain of logistic freight, requirements for FInest EPM are specified with the goal of pushing generic event processing capabilities into the core platform. To that end, *functional requirements*, which are specific to the freight logistics domain, are distinguished from the *technical requirements*, which are generic requirements for an event-processing engine.
- Finally, a key requirement of FInest is to be deployed on cloud; this will boost performance, the dynamic updates of the system, and allow companies to connect to it without the need to install local systems. As far as we know, existing systems mentioned above do not facilitate cloud deployment.

4. Initial Requirements Analysis

This section provides the initial analysis of requirements for the EPM. The functional requirements were gathered by an iterative process in collaboration with domain partners. First, a general input form was sent to partners (Appendix A), asking for input on event handling scenarios. With that input, IBM built a specific questionnaire for each domain partner, asking for further details on the specific information provided by that partner. In addition, several general questions were sent to all partners (Appendix B). Based on the responses to these questionnaires (which are not included for confidentiality concerns) IBM generated requirements draft, that was further refined as a result of a face to face meeting with the partners.

The two other types of requirements ("technical" and "non-functional") are both technical, however those under the group "technical" refer to functionality of event processing as a technological platform, according to the high level conceptual design outlined in Section 2; those under the group "non-functional" refer to general computing capabilities.

The technical and non-functional requirements are based on years of experience with event-driven computing within IBM HRL, and in particular on a recent book that summarizes this accumulated knowledge [5].

4.1. Functional Requirements

The role of the functional requirements is to define the required functionality of a proactive event-processing component for monitoring logistics scenarios. The requirements are divided into five groups.

The group *availability of events* covers the requirement of this WP to ensure that all required input events are available; this is not a requirement of the event processing software component per se, and might be delegated to the core platform (cf. Section 6). For this reason it is grouped separately. The second group deals with *configurability* of the set of rules employed by the system.

The main group is the *shipment monitoring*, which covers the requirements for end-to-end visibility of shipment scenarios. These scenarios are potentially multi-modal, with ground, air, and

sea freight segments. It includes various events and settings that were identified by FInest team this far; however, there is also a requirement to allow adding additional types of events. The group *cargo carrier monitoring* covers specific requirements from particular modes of transportation, mostly air and rail, coming from the Cargo 2000 standard [11]. The group *vessel call monitoring* concerns additional requirements that are related to the vessel call process in a port.

4.1.1. Availability of Events

Id	Name	Description
R101	External Events	Availability of external events relevant to shipment: traffic in route segment, weather affecting freight vessels, security alerts, etc.
R102	Internal Events	Availability of events indicating progress of shipment, either through integration with existing systems or entered manually (order, pickup, loading…).
R103	Sea Freight Events	Availability of events from sea vessels, sea ports, and AIS.
R104	Air Cargo Events	Availability of events related to airport operation and air cargo status within airport (flight status, cargo loading and unloading, cargo inspected by customs…), either through integration with airport systems or entered manually.
R105	Ground Freight Events	Availability of events from ground carriers (loading, unloading, location information, damage).

4.1.2. Configurability

Id	Name	Description
R201	Global Rule Set (Template)	A set of rules from which users can select which to activate in specific deployment / scenario
R202	Parameterized Rules	The available rules are parameterized, and receive specific values per parameter from a transport plan (e.g., how long is significantly late, what is a critical container temperature).

4.1.3. Shipment Monitoring

Id	Name	Description

R301	Delivery Monitoring	Provide tracking of current deliveries associated with an open contract, by monitoring internal events and external events, and comparing with contract terms (cf. Requirements R302-R315).
R302	ETA Tracking	Maintain and update ETA according to the current location, current time, and events known regarding the expected route (cf. Requirements R303-R312, R401, R401-R402).
R303	Shipment Location	Monitor physical location of delivery, according to location events.
R304	Order Event	Receive order event and initiate shipment monitoring for the order, using TP requirements (which are driven by SLA from the associated contract).
R305	Pickup Event	Receive pickup event and update shipment status. Compare with contract / order pickup time and notify on discrepancy. Update ETA.
R306	Shipment Status Event	Handle various events related to shipment status, and update status and ETA accordingly (loading and unloading to / from a carrier, in terminal or off terminal, etc.
R307	Delivery Event	Receive delivery event, terminate shipment monitoring, notify BCM and front-end.
R308	Monitor Schedule	Compare scheduled time of events R205-R208 with actual time, also notify if scheduled time passed significantly and event did not take place.
R309	Customs Delay	Receive events of delay due to customs inspection (either automatically or entered manually).
R310	Other Exception Handling	Receive manually entered exception events (booking discrepancies, missing documentation, cargo loading problem…) and notify front-end and / or BCM.
R311	Weather Events	Identify weather events that affect the route of a vessel carrying a shipment. Notify front-end, and calculate effect on ETA if possible.
R312	Traffic Events	Process events indicating traffic problems on current delivery routes. Notify front-end, and calculate effect on ETA if possible.
R313	Damage Monitoring	Collect events that indicate possible damage to cargo, either entered manually or inferred, when possible, from other events such as weather conditions. If possible to calculate specific damage figures, compare with SLA if indicated in TP.
R314	Adding Events	Allow the addition of new types of events and rules that handle them (also after deployment), beyond those that are known at the moment and that are classified above.

R315	Hierarchical Transport Plan	Support hierarchical decomposition of TP. This means that there might be a segmentation of input and output events according to hierarchy level; the status events sent to BCM and front-end will carry this level Id to ensure that the right player is notified on the event.

4.1.4. Monitoring Freight Carrier

Id	Name	Description
R401	Cargo 2000 seven mile-stones	Monitor the seven process milestones defined by the Cargo 2000 standard; detect the lack of milestone achievement on time. Update ETA of associated deliveries when needed.
R402	Cargo 2000 Exception	Detect exceptions according to the Cargo 2000 standard – notify BPM and update ETA.

4.1.5. Vessel Call Monitoring

Id	Name	Description
R501	Vessel Moni-toring	Provide tracking of vessels from their initial call to port, until the vessel leaves the port. Update ETA of all shipments related to vessel when needed.
R502	Track Booking Status	Track progress of port related bookings done by vessel (pilot, quay, tugboat, mooring crew, equipment, cargo workers, water and elec-tricity, storage and supply), including the receipt of feedback from port system to the booking request.
R503	Receive AIS Notifications	Arrival to port, arrival to quay, departure from quay, departure from port district
R504	Notify on Delay	When a serious delay in vessel call processing is incurred, send noti-fication regarding affected vessels and shipments.
R505	Notify on Full Booking	When port resources are expected to be fully booked in a future time frame (for example, after ships arrived without announcing in ad-vance), send notification regarding affected vessels and shipments.

4.2. Technical Requirements

4.2.1. Events and Events Sources

Id	Name	Description
Rt101	Event Definition	Define event as an entity with type, temporal dimension (occurrence time and detection time), payload (attributes and values), and indication of event source.
Rt102	Calendar and Scheduled Events	Automatically emitting calendar events (e.g., holidays) and scheduled physical events (e.g., major sports event).
Rt103	Dynamic Event Types and Event Sources	Allow dynamic definition of data sources with their respective event types and structure; this applies to sources which are either external systems or internal components.
Rt104	Emitting Output Events	The system should be able to send output events to specific clients, preferably via a pub / sub system.

4.2.2. Event Processing Network

Id	Name	Description
Rt201	Event Processing Units	Distributed processing of events, obtained by supporting standalone-processing units. A unit receives only specific types of events, in a specific context (Section 4.2.3), and performs a well-defined operation (Section 4.2.4).
Rt202	Event Processing Network (EPN)	Allow the specification of an overlay routing network, which prescribes how events are routed between the processing units.
Rt203	Event Channel	Support the automatic routing of events between Event Processing Units, according to EPN, event types and context (Section 4.2.3).
Rt204	Dynamic Rule Changing	Allow the refinement and changing of rules when the system is online.
Rt205	Global State	Support of shared memory: knowledge base and global state. Event processing units (Section 4.2.2) can write global variables for others to use.

4.2.3. Processing Operators

Id	Name	Description
Rt401	Basic Event Processing	Perform filtering, transformation, and derivation of events
Rt402	Join Operators	Operators that detect complex events which are joins of raw events: sequences, collections (with conjunctions, disjunctions, negations), and selection of subsets (e.g., k events with highest value of an attribute).
Rt403	Assertions	Any operator may include various assertions on values of attributes, possibly with comparison of attribute values of different events.
Rt404	Information Aggregation	Functions defined on attribute values of sets of events (summation, average, max / min, …)

4.2.4. Event Context

Id	Name	Description
Rt301	Temporal Context	Define time windows in which processing operators match or aggregate events.
Rt302	Segmentation Context	Allow segmentation of event processing units according to values of attributes of input events.
Rt303	Context by State	Allow segmentation and selection of event processing units according to values of state variables (Global State).
Rt304	Composite Context	Allow combination of multiple temporal, segmentation, and state contexts.

4.3. Non-Functional Requirements

4.3.1. Cloud Hosting

Id	Name	Description
Rn101	Distribution on Cloud	Support distributed processing over a large number of machines, keeping the correct semantics. Automatic assignment to machines

		based on topology of Event Processing Network.
Rn102	Load Balancing	Support runtime load balancing.
Rn103	Parallel Execution	Support parallel execution of instances of Event Processing Units.

4.3.2. Robustness

Id	Name	Description
Rn201	Availability	Ensuring availability of event-processing system
Rn202	Recovery	Perform logging, maintain checkpoints for restart, and ensure persistence of state.
Rn203	Failover	Always have sufficient number of functioning machines in case of failure of some of the machines.

5. Technical State-of-the-Art Analysis

5.1. Existing Event-Processing Systems

Event processing is an emerging software engineering paradigm and computing field. It can be defined as follows [5]:

Event Processing is computing that performs operations on events. Common event processing operations include reading, creating, transforming, and deleting events.

The market of event processing software platforms is growing [8]. An event processing system is usually a software platform providing some or all of the following [5]:

1) A language for expressing event processing logic
2) Tools to design and test event processing logic
3) A runtime to execute event processing logic
4) An event distribution mechanism
5) Operational management tools

In Contrast to the domain specific systems surveyed in Section 3, an event processing platform leaves the power of creating the application flow in the hands of the system designer, rather than the software developer. This is done through a language or design tools through which an event processing application for a particular domain is designed (point 1 and 2 above). The other main part of an event processing system is a runtime engine (3 above); a deployment of the runtime engine executes the logic resulted from the design tools. The runtime engine receives streams of

events, processes them, and potentially creates new events (called *derived events* [5]); the derived events can be distributed to various event consumers (4 above). In addition, the platform may provide operational management tools that help users access, monitor, and intervene in the execution.

Cugola and Margara [3] recently performed an exhaustive survey of existing systems, classified as *Information Flow Processing (IFP)* systems. Their survey is even more comprehensive then merely covering the existing *Complex Event Processing (CEP)* systems; they also include two other approaches to the general task that can be defined as *continuous and timely processing and responding to flow of incoming information*:

- *Active databases* employ the rules on a database whenever it changes; the rules can potentially include CEP logic [9].
- *Data Stream Management Systems (DSMS)* deal with abstract flows of data, not necessarily structured as events; these systems usually lack the tight semantics provided by CEP. The latter is a more general concept, however such systems lack the precise semantics usually provided by CEP systems.

We are not aware of relevant systems which are not covered by this survey; hence we will use their result in this analysis.

The systems are compared over the following properties:

Functional Model (Table 1)

Clock: The ability to generate scheduled events (as opposed to purely reacting to input events). This is essential in a practical business environment (see Requirement Rt102).

K-Base: The ability to access persistent storage, rather than purely information delivered by events. The ability to predict ETA given changing conditions depends on such capability to access resource information (see Requirement Rt205).

Seq: Refers to the sequence of events that match a rule body. For FInest, it is preferred that the length of the sequence does not need to be bounded in advance, hence it can match any number of events (sequences are mentioned in Requirement Rt402).

Recursion: Allow information items produced by triggered rules to re-enter the system. This is also important for FInest. For example, detecting a sequence of traffic events implies delay during a particular segment; this delay should be input to ETA calculation, which in turn serves as input to comparison with contract terms (Requirements Rt202 and Rt203).

Dynamic Rule Change: The ability to modify rules while the system operates. This is essential for any real world business application (Requirement Rt204).

Processing Model (Table 1)

The processing model refers to a policy which characterizes rule matching – how many times each rule is matched, which match is picked, can events be matched twice, and more. This part is not particularly important for FInest, and hence the criteria of this model and the respective part of Table 1 are ignored.

Data Model (Table 2)

Nature of Items: Input to FInest is assumed to be structured as events, rather than plain data streams or a static database (Requirement Rt101).

Format: Format of input items; not an important distinction at this point.

Uncertainty: At this point the input events to FInest are assumed to be precise rather than uncertain. Uncertain events and uncertain processing rules are considered in literature [4,10] but rarely implemented satisfactorily in practice.

Time Model (Table 2)

This section refers to the ability to order events on the time dimension (referred to in the table as "Absolute", or "Interval"). This is a must for FInest (and almost any other complex event processing system).

Rule Model (Table 2)

This section refers to various ways to represent and implement rules; the distinction is not crucial for FInest.

Language Model (Table 3)

Refers to the expressiveness provided to the rule base designer of an application on top of the event handling system.

Single Item: processing of a single item (event), either allowing **filtering** (selection of subset of the items), modifying the information on the event ("**Renaming**" and "**Projection**"), or transformation of the item according to state or fixed data. This is basic functionality usually provided by CEP engines (see Requirement Rt401).

Logic: matching a set of events according to some logic operator. FInest will requires the basic operators: conjunction (all of the events in a set were detected in the context window), disjunction (at least one of the events in a set were detected), and negation (none of the events in a set were detected). This is covered by Requirements Rt402, Rt403, Rt404.

Sequence: matching of a set of events according to a specific ordering constraint over time. This is also basic and required functionality (Rt402).

Iterations: selection of subset of events by defining an iteration condition. This is an advanced operation that does not seem to be necessary at the moment for any functional requirement of FInest.

Windows: this refers to what is called in this document the temporal context; how does the designer express the temporal window under which specific rules are evaluated. The concept of sliding window is a convenient representation that is appropriate for FInest (Rt301).

Flow Management: this is not relevant for CEP systems, where different event types are joined within logic operators.

The results on existing systems are presented in Tables 1-3, extracted from [3]. Two additional dimensions (Deployment and Interaction models) employed by the authors are omitted as they are less relevant for the purpose of this baseline selection.

Table 1: Functional and Processing Models of IFP Systems

	Functional Model					Processing Model		
Name	Clock	K. Base	Seq	Recursion	Dynamic Rule Change	Select. Policy	Consum. Policy	Load Shedding
HiPac	Present	Present	Bounded	Yes	Yes	Multiple	Zero	No
Ode	Absent	Present	Unbounded	Yes	Yes	Multiple	Zero	No
Samos	Present	Present	Unbounded	Yes	Yes	?	?	No
Snoop	Present	Present	Unbounded	Yes	Yes	Program.	Program.	No
TelegraphCQ	Present	Present	Unbounded	No	No	Multiple	Zero	Yes
NiagaraCQ	Present	Present	Unbounded	No	No	Multiple	Selected	Yes
OpenCQ	Present	Present	Unbounded	No	No	Multiple	Selected	No
Tribeca	Absent	Absent	Unbounded	No	Yes	Multiple	Zero	No
CQL / Stream	Absent	Present	Unbounded	No	No	Multiple	Zero	Yes
Aurora / Borealis	Absent	Present	Unbounded	No	No	Program.	Zero	Yes
Gigascope	Absent	Absent	Unbounded	No	No	Multiple	Zero	No
Stream Mill	Absent	Present	Unbounded	No	No	Program.	Program.	Yes
Traditional Pub-Sub	Absent	Absent	Single	No	No	Single	Single	No
Rapide	Absent	Present	Unbounded	Yes	No	Multiple	Zero	No
GEM	Present	Absent	Bounded	Yes	Yes	Multiple	Zero	No
Padres	Absent	Absent	Unbounded	No	No	Multiple	Zero	No
DistCED	Absent	Absent	Unbounded	No	No	Multiple	Zero	No
CEDR	Absent	Absent	Unbounded	Yes	No	Program.	Program.	No
Cayuga	Absent	Absent	Unbounded	Yes	No	Multiple	Zero	No
NextCEP	Absent	Absent	Unbounded	Yes	No	Multiple	Zero	No
PB-CED	Absent	Absent	Unbounded	No	No	Multiple	Zero	No
Raced	Present	Absent	Unbounded	Yes	No	Multiple	Zero	No
Amit	Present	Absent	Unbounded	Yes	No	Program.	Program.	No
Sase	Absent	Absent	Bounded	No	No	Multiple	Zero	No
Sase+	Absent	Absent	Unbounded	No	No	Program.	Zero	No
Peex	Absent	Absent	Unbounded	Yes	No	Multiple	Zero	No
TESLA/T-Rex	Present	Absent	Unbounded	Yes	No	Program.	Program.	No
Aleri SP	Present	Present	Unbounded	No	No	Multiple	Zero	?
Coral8 CEP	Present	Present	Unbounded	No	No	Program.	Program.	Yes
StreamBase	Present	Present	Unbounded	No	No	Multiple	Zero	?
Oracle CEP	Absent	Present	Unbounded	No	No	Program.	Program.	?
Esper	Present	Present	Unbounded	No	No	Program.	Program.	No
Tibco BE	Absent	Absent	Unbounded	Yes	No	Multiple	Zero	?
IBM System S	Present	Present	Unbounded	Yes	No	Program.	Program.	Yes

Table 2: Data, Time, and Rule Models of IFP Systems

	Data Model				Time Model	Rule Model	
Name	Nature of Items	Format	Support for Uncert.	Nature of Flows	Time	Rule Type	Probab. Rules
HiPac	Events	Database and External Events	No	Heterogeneous	Absolute	Detecting	No
Ode	Events	Method Invocations	No	Heterogeneous	Absolute	Detecting	No
Samos	Events	Methon Invocation and External Events	No	Heterogeneous	Absolute	Detecting	No
Snoop	Events	Database and External Events	No	Heterogeneous	Absolute	Detecting	No
TelegraphCQ	Data	Tuples	No	Homogeneous	Absolute	Transforming	No
NiagaraCQ	Data	XML	No	Homogeneous	Stream-only	Transforming	No
OpenCQ	Data	Tuples	No	Heterogeneous	Stream-only	Transforming	No
Tribeca	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
CQL / Stream	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
Aurora / Borealis	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
Gigascope	Data	Tuples	No	Homogeneous	Causal	Transforming	No
Stream Mill	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
Traditional Pub-Sub	Events	System dependent	System dependent	Heterogeneous	System dependent	Detecting	System dependent
Rapide	Events	Records	No	Heterogeneous	Causal	Detecting	No
GEM	Events	Records	No	Heterogeneous	Interval	Detecting	No
Padres	Events	Records	No	Heterogeneous	Absolute	Detecting	No
DistCED	Events	Records	No	Heterogeneous	Interval	Detecting	No
CEDR	Events	Tuples	No	Homogeneous	Interval	Detecting	No
Cayuga	Events	Tuples	No	Homogeneous	Interval	Detecting	No
NextCEP	Events	Tuples	No	Homogeneous	Interval	Detecting	No
PB-CED	Events	Tuples	No	Heterogeneous	Interval	Detecting	No
Raced	Events	Records	No	Heterogeneous	Absolute	Detecting	No
Amit	Events	Records	No	Heterogeneous	Absolute	Detecting	No
Sase	Events	Records	No	Heterogeneous	Absolute	Detecting	No
Sase+	Events	Records	No	Heterogeneous	Absolute	Detecting	No
Peex	Events	Records	Yes	Heterogeneous	Absolute	Detecting	Yes
TESLA / T-Rex	Events	Tuples	No	Heterogeneous	Absolute	Detecting	No
Aleri SP	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
Coral8 CEP	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
StreamBase	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
Oracle CEP	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
Esper	Data	Tuples	No	Homogeneous	Stream-only	Transforming	No
Tibco BE	Events	Records	No	Heterogeneous	Interval	Detecting	No
IBM System S	Data	Various	No	Homogeneous	Stream-only	Transforming	No

Table 3: Language Models of IFP Systems

		Single-item			Logic						Windows						Flow Management									
Name	Type	Selection	Projection	Renaming	Conjunction	Disjunction	Repetition	Negation	Sequence	Iteration	Fixed	Landmark	Sliding	Pane	Tumble	User Defined	Join	Union	Except	Intersect	Remove Dup	Duplicate	Group By	Order By	Parameterization	Flow Creation	Detection Aggr	Production Aggr
HiPac	Det	X				X			X								X	X					X	X	X			X
Ode	Det	X			X	X	X	X	X	X															X			X
Samos	Det	X			X	X	X	X	X	X	X														X			X
Snoop	Det	X			X	X	X		X	X															X			X
TelegrahCQ	Decl	X	X	X							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
NiagaraCQ	Decl	X	X	X													X	X	X	X	X	X	X	X	X	X	X	X
OpenCQ	Decl	X	X	X	X	X		X									X	X	X	X	X	X	X	X	X	X	X	X
Tribeca	Imp	X	X		X	X		X	X				X				X	X	X	X	X	X	X	X	X	X	X	X
CQL/Stream	Decl	X	X	X							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Aurora/Borealis	Imp	X	X										X		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gigascope	Decl	X															X	X	X	X	X	X	X	X	X	X	X	X
Stream Mill	Decl	X	X								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Traditional Pub-Sub	Det	X																							X			X
Rapide	Det	X			X	X	X	X	X	X	X		X				X	X							X			X
GEM	Det	X			X	X		X	X			X	X												X		X	X
Padres	Det	X			X	X			X	X															X			X
DistCED	Det	X			X	X		X	X	X			X												X			X
CEDR	Det	X	X	X	X	X		X	X								X	X	X	X	X				X		X	X
Cayuga	Det	X	X	X	X	X		X	X	X							X								X		X	X
NextCEP	Det	X	X	X	X	X		X	X	X			X												X		X	X
PB-CED	Det	X			X	X		X	X		X														X			X
Raced	Det	X			X	X	X	X			X	X	X												X			X
Amit	Det	X			X	X	X	X	X	X			X												X		X	X
Sase	Det	X			X	X	X	X	X				X												X			X
Sase+	Det	X			X	X	X	X	X	X															X		X	X
Peex	Det	X			X	X	X	X	X		X														X			X
TESLA/T-Rex	Det	X						X	X				X												X			X
Aleri SP	Imp+Det	X	X	X	X	X	X	X	X	X			X	X	X		X	X							X			X
Coral8 CEP	Decl+Imp+Det	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
StreamBase	Decl+Det+Imp	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Oracle CEP	Decl+Det+Imp	X	X	X	X	X	X	X	X	X			X				X	X	X	X	X	X	X	X	X	X	X	X
Esper	Decl+Det	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Tibco BE	Det	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
IBM System S	Decl+Imp	X	X	X							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

5.2. Selection of Technology Baseline

The initial choice made in the design of FInest is to employ a complex event processing system. It is preferred to a DSMS, because the former are capable of supporting the complex semantics of handling freight related events and monitoring SLAs; this factor is much more significant than the importance of the superior scalability that can potentially be achieved by DSMS. Furthermore, CEP is preferred in this context to active database technology, because of the dynamic nature of a freight scenario; CEP is in general more appropriate for monitoring a process whereas active databases work well for static data with sparse changes.

In Tables 1-3, the rows corresponding to CEP systems are all those from “Rapide”, to “Tesla”. Our selection of this set is the system AMIT [6], whose immediate advantage is its availability: it is a research asset developed and maintained by IBM Haifa Research Lab.

Furthermore, AMIT is at least competitive with the rest of the systems. It includes almost all the functionality that is described above as significant for FInest, and also supports several requirements (in Section 4.2.4) which are not covered by the survey. The only important features which are missing according to [3] are:

Knowledge Base: Although AMIT has basic ability to access persistent storage, this has to be extended to allow smooth access to large resource databases. The only CEP system which provides this functionality today in a satisfactory manner is Rapide; however, the latter does not have the clock functionality which is also essential.

Dynamic rule change: The only CEP system that does provide this capability is GEM, which, on the other hand, does not have unbounded sequences. However, AMIT will have to be augmented with this functionality as well.

Projection and **Renaming**: In fact, AMIT has a simple way to obtain the same functionality, by allowing the event processing units to create new events based on the input events.

To conclude, as technology baseline we choose AMIT, an event processing engine which is available for WP6 team. It provides most of the basic underlying functionality that is mentioned under our technical requirements; AMIT will be augmented to support some enhancements.

6. Initial Identification of Potential Generic Enablers

The work done in this WP towards identification of generic enablers is provided within D3.1 (Section *Initial Generic Enablers Requests*), which summarizes this work from all of FInest WPs. Essentially, we envision the core platform will provide the technical event processing engine, and the technical requirements section above provides the requirements of FInest EPM from such engine. Section 5 of Deliverable D3.1 ("Initial Generic Enablers Requests") provides a complete list of requests for Generic Enables from all FInest work packages. As a result, we refer to that section in D3.1 and omit a precise description of Generic Enablers for the ECM. The complete list also can be accessed through the following Web site: **http://www.finest-ppp.eu/index.php/project-results/generic-enablers**

In essence, the requirement analysis of this work package results in three GE requests: (i) an event processing engine, which provides the capabilities described in the technical and non-functional requirements in this document; (ii) event gathering capability corresponding to the group of *availability of events* requirements in Section 4.1.1, and (iii) Enabler that provides dissemination of events to consumers, through publish / subscribe broker, mentioned in Requirement Rt104.

7. Conclusion and Next Steps

In this document we provide the initial set of technical, functional, and non-functional requirements for the event-processing component in FInest. This list of requirements is initial and is based on assessment of domain knowledge. As the project progresses and more domain knowledge is obtained the requirements will be expanded and refined. Beyond refinement of the requirements, the major next step in this WP is a conceptual architecture that will support this functionality; this will be provided in the next deliverable, *D6.2: Conceptual Design of the Event Processing Component*, to be submitted in month 12 of this project. The architecture will be an extension and refinement of the overview provided in this document.

References

[1] http://www.imo.org/OurWork/Safety/Navigation/Pages/AIS.aspx

[2] http://www.spacecentre.no/AISSat-1+%E2%80%93+the+facts.d25-TwlnW5j.ips

[3] Gianpaolo Cugola and Alessandro Margara, *Processing Flows of Information: From Data Stream to Complex Event Processing*, to appear in ACM Computing Surveys. Available through: http://home.dei.polimi.it/margara/papers/survey.pdf

[4] Yagil Engel and Opher Etzion, *Towards Proactive Event Driven Computing*, in Distributed Event-Based Systems (DEBS), 2011

[5] Opher Etzion and Peter Niblett, *Event Processing in Action*, Manning, 2010.

[6] Asaf Adi and Opher Etzion, *AMIT - the situation manager*. VLDB J. 13(2): 177-203 (2004)

[7] Opher Etzion, Yonit Magid, Ella Rabinovich, Inna Skarbovsky, and Nir Zolotorevsky, *Context aware computing and its utilization in event-based systems.* Distributed Event-Based Systems (DEBS), 2010.

[8] Gartner Report G00213721, *The Trend toward Intelligent Business Operations*, 24 June 2011.

[9] Detlef Zimmer and Rainer Unland, *On the Semantics of Complex Events in Active Database Management Systems*. ICDE 1999: 392-399

[10] Segev Wasserkrug, Avigdor Gal, and Opher Etzion, *A Model for Reasoning with Uncertain Rules in Event Composition Systems.* Uncertainty in Artificial Intelligence (UAI), 2005.

[11] http://www.iata.org/whatwedo/cargo/cargo2000/Pages/master-operating-plan.aspx

Appendix A - High level input on event processing component

This is a high level input template filled by FInest partner for the purpose of EPM requirement analysis. Partners were requested to provide three scenarios and answer the questions below.

Organization: KN

Written by:

Please choose 3 different shipment scenarios you could participate in. On each of these 3 different scenarios, please consider the types of events that can occur during the scenario and may require monitoring and/or response. Please fill out the following sections.

Scenario 1:

Name:

Description:

Please create a list of events that might occur during the scenario. We are interested in any event you can think of that might affect the scenario – this includes both events that are monitored today, and events that are not monitored but it would help if they were.

For each event, please provide the information requested below. Please copy the template below for each event in your list.

Event Description:

How can this event be detected? (Please elaborate: must the detection be manual, or can be automatic?)

Is this event monitored today? How?

Should this event be detected as part of a set or series of events?

What should be done if this event (or series of events) is detected?

Who needs to know about this event (e.g., port, agent, terminal)

Appendix B – Questionnaires for Partners

The second round of domain knowledge extraction was done with partner-specific questionnaires that are omitted from this report. In addition to the specific questions, the following question was posed to all the partners:

General questions about receiving and sending events

1. How closely do you need to monitor the progress of a vessel or a shipment? For example, do you follow the exact location of a vessel at sea (NCL)? Do you need to know the exact segment of the route in which a shipment currently is (KN)?
2. Each of your companies provides some kind of service to customers under contract (except perhaps Port of Alesund?). We wish to know more about the specific possible breaches of contract / SLA that we need to detect:
    a. Please list standard contractual obligations (e.g., time of shipment arrival), which are measured under SLA.
    b. How can each of these be breached?
3. In the previous questions we covered tracking with respect to the planned route and contractual obligations. Are there additional tracking requirements (which could be the results of contracting, collaboration, and planning)?
4. Which physical sources of data do you:
    a. Get data from today
    b. Envision for the future (e.g., does it make sense to get GPS information from trucks? Is it needed to have an RFID tag on each container?