

SPEEDD – **Scalable Proactive Event-Driven Decision-making**



|  |
| --- |
| D6.1 |
| Architecture Design |

|  |  |  |  |
| --- | --- | --- | --- |
| Project Acronym | SPEEDD | | |
| Project Title | Scalable Proactive Event-Driven Decision-making | | |
| Project Number | 619435 | | |
| Workpackage | #WP6 | Scalability and System Integration | |
| Lead Beneficiary | IBM | | |
| Editor | Alexander Kofman | | IBM |
| Contributors | Inna Skarbovsky | | IBM |
|  | Fabiana Fournier | | IBM |
|  | TBD | | TBD |
| Reviewers | Alexander Artikis | | NCSR |
|  | TBD | | TBD |
| Dissemination Level | Public | | |
| Contractual Delivery Date | 31/09/2014 | | |
| Actual Delivery Date | 31/09/2014 | | |
| Version | V1.0 | | |

Abstract

This report presents TBD

Document History

|  |  |  |
| --- | --- | --- |
| Version | Date | Comments |
| V0.1 | tbd | First draft |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[Abstract 2](#_Toc399660665)

[Document History 3](#_Toc399660666)

[Table of Contents 4](#_Toc399660667)

[List of Tables 5](#_Toc399660668)

[List of Figures 5](#_Toc399660669)

[Acronyms 6](#_Toc399660670)

[1. Introduction 7](#_Toc399660671)

[2. System Requirements 7](#_Toc399660672)

[3. Approach 7](#_Toc399660673)

[4. Conceptual Architecture 8](#_Toc399660674)

[5. SPEEDD Runtime Architecture 10](#_Toc399660675)

[5.1. Event Bus 12](#_Toc399660676)

[5.2. Event/Data Providers 12](#_Toc399660677)

[5.3. Action Consumption – Actuators/Connectors 12](#_Toc399660678)

[5.4. Complex Event Processor 12](#_Toc399660679)

[5.5. Decision Management 12](#_Toc399660680)

[5.6. Dashboard application 12](#_Toc399660681)

[6. Build-Time Architecture 12](#_Toc399660682)

[6.1. Event Pattern Mining 12](#_Toc399660683)

[6.2. Authoring of CEP Rules 12](#_Toc399660684)

[6.3. Decision Management – the Offline Part 13](#_Toc399660685)

[7. Integration – APIs and Data Formats 13](#_Toc399660686)

[8. Non-Functional Aspects 13](#_Toc399660687)

[8.1. Scalability 13](#_Toc399660688)

[8.2. Fault Tolerance 13](#_Toc399660689)

[8.3. Testability 13](#_Toc399660690)

[9. Summary 13](#_Toc399660691)

[10. Appendix A – Technology Evaluation 13](#_Toc399660692)

[10.1. Stream Processing – requirements and evaluation criteria 14](#_Toc399660693)

[10.2. Storm 14](#_Toc399660694)

[10.3. Akka 14](#_Toc399660695)

[10.4. Spark Streaming 14](#_Toc399660696)

[10.5. Choice of the Messaging Platform 14](#_Toc399660697)

List of Tables

**No table of figures entries found.**

List of Figures

[Figure ‎3.1 - SPEEDD design architecture approach 8](#_Toc399660698)

[Figure ‎4.1 - Conceptual Architecture of SPEEDD Prototype 9](#_Toc399660699)

[Figure ‎5.1 - SPEEDD - Event-Driven Architecture 10](#_Toc399660700)

[Figure ‎5.2 - SPEEDD Runtime - Event-Driven Architecture (Traffic Use Case) 11](#_Toc399660701)

[Figure ‎5.3 - SPEEDD Runtime - Event-Driven Architecture (Credit Card Fraud Use Case) 11](#_Toc399660702)

Acronyms

|  |  |
| --- | --- |
| Acronym | Explanation |
| BCM | Business Collaboration Module |
| CEP | Complex Event Processing |
| EPA | Event Processing Agent |
| EPM | Event Processing Module |
| EPN | Event Processing Network |
| GE | Generic Enabler |
| FInest | Future Internet enabled optimisation of transport and logistics networks |
| JMS | Java Message Service |
| JSON | JavaScript Object Notation |
| IoT | Internet of Things |
| TCP | Transport Chain Plan |
| TEP | Transport Execution Plan |
| TPM | Transport Planning Module |
| WP | Work Package |

# Introduction

Describe the goal and the structure of this document.

# System Requirements

This section lays out the main requirements for SPEEDD prototype. The requirements are based on the requirements documents provided by the use cases.

# Approach

The design of the system architecture for a prototype like SPEEDD is an iterative process that starts with the beginning of the project and continuously evolves, as requirements of the different components are better understood and insights are gained. Therefore, a close iterative and collaborative process was carried out between the architecture team in WP6 “Scalability and System Integration” lead by IBM, and the technical teams of the SPEEDD prototype, specifically the teams of the real-time event recognition and forecasting (WP3), real-time decision making (WP4), real-time visual analytics (WP5), scalability (WP6), and the technical teams from the use cases (WP7 and WP8).

To this end we followed the steps below, as illustrated in Figure 1:

1. Iterative biweekly virtual meetings that included representatives of all partners involved. A very draft architecture presented at M3 of the project has been frequently updated and refined based on input and feedback to the current architecture (showed in Figure XXX).
2. On a case-by-case basis, bilateral virtual meetings with a specific partner to elaborate on a specific issue (e.g., specific API).
3. Face-to-face meetings during the project meetings in May and September 2014.



Figure ‎3.1 - SPEEDD design architecture approach

# Conceptual Architecture

This section provides a high-level overview of SPEEDD prototype. The goal is to introduce the main concepts, high-level components and information flow without getting into implementation and technological details.

Figure ‎3.1illustrates the conceptual architecture of SPEEDD prototype. We separate between the design time and the run time. The products of the design time activities are event processing definitions and decision management algorithms and configurations that will be deployed and executed at the runtime.



Figure ‎4.1 - Conceptual Architecture of SPEEDD Prototype

Historic data used at design time contains raw events reported during the observed period along with annotations provided by domain experts. These annotations mark important situations that have been observed in past and should be detected automatically in future. Visualization tooling is used to sift through historic data to gain insights and create annotations. Domain experts apply tools and methodologies provided by SPEEDD authoring toolkit to extract complex event pattern definitions from the annotated event history. This is a semi-automatic process involving applying machine learning tools to extract initial set of patterns which is further enhanced and translated with help of the domain experts into deployable CEP artefacts.

The runtime part is composed of the CEP component, the automatic decision management component, and visual decision support tooling. SPEEDD runtime receives raw events emitted by the various event sources (e.g. traffic sensors, transactional systems, etc., - depending on the use case) and emits actions that are consumed by the actuators connected to the operational systems or simulators.

The CEP component is capable of detecting and forecasting complex event patterns under uncertainty. It processes raw as well as derived (detected and forecasted) events to detect and forecast higher-level events, or situations. These serve as triggers for the decision management component, which uses domain-specific algorithms to suggest the next best action to resolve or prevent an undesired situation.

The visualization component (further called the dashboard) facilitates decision making process for business users by providing easily comprehensible visualization of detected or forecasted situations along with output of the automatic decision making component – a list of suggested actions to deal with the situation. The SPEEDD system can be run in either open or closed loop mode. In case of the open loop, the user can approve, reject, or modify the action proposed by the automatic decision maker. The closed loop operation does not require user’s approval, - the action is performed automatically[[1]](#footnote-1). A hybrid mode where some types of actions are taken automatically while other types require human attention is also supported; moreover, we believe that this mode is the most realistic one.

# SPEEDD Runtime Architecture

The architecture of the runtime part of SPEEDD follows the “Event-Driven Architecture” paradigm. Every component functions as an event consumer, or an event producer, or a combination of both. The event bus plays a central role in facilitating inter-component communication which is done via events. Figure ‎6.1 provides a refinement of the conceptual architecture described above where the runtime part is represented as a group of loosely-coupled components interacting through events. The event bus plays a central role as it provides the communication and integration platform for an event-driven system.

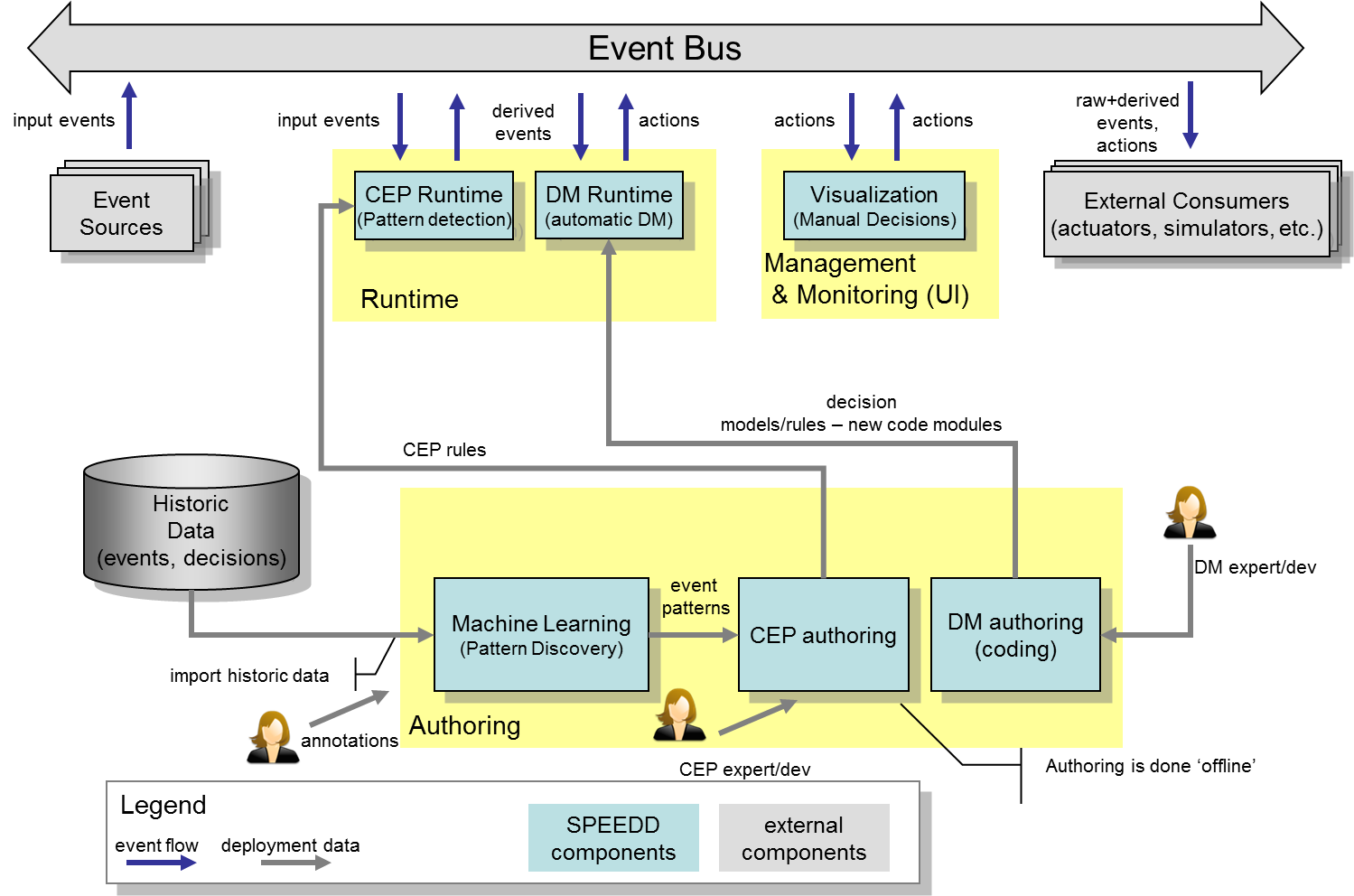


Figure ‎5.1 - SPEEDD - Event-Driven Architecture

In the following subsections we are describe the individual architecture of all the components as well as discuss the technology used to implement them.

Figure ‎6.1 and Figure ‎6.2 illustrate the architecture for the traffic and credit card fraud use cases respectively.

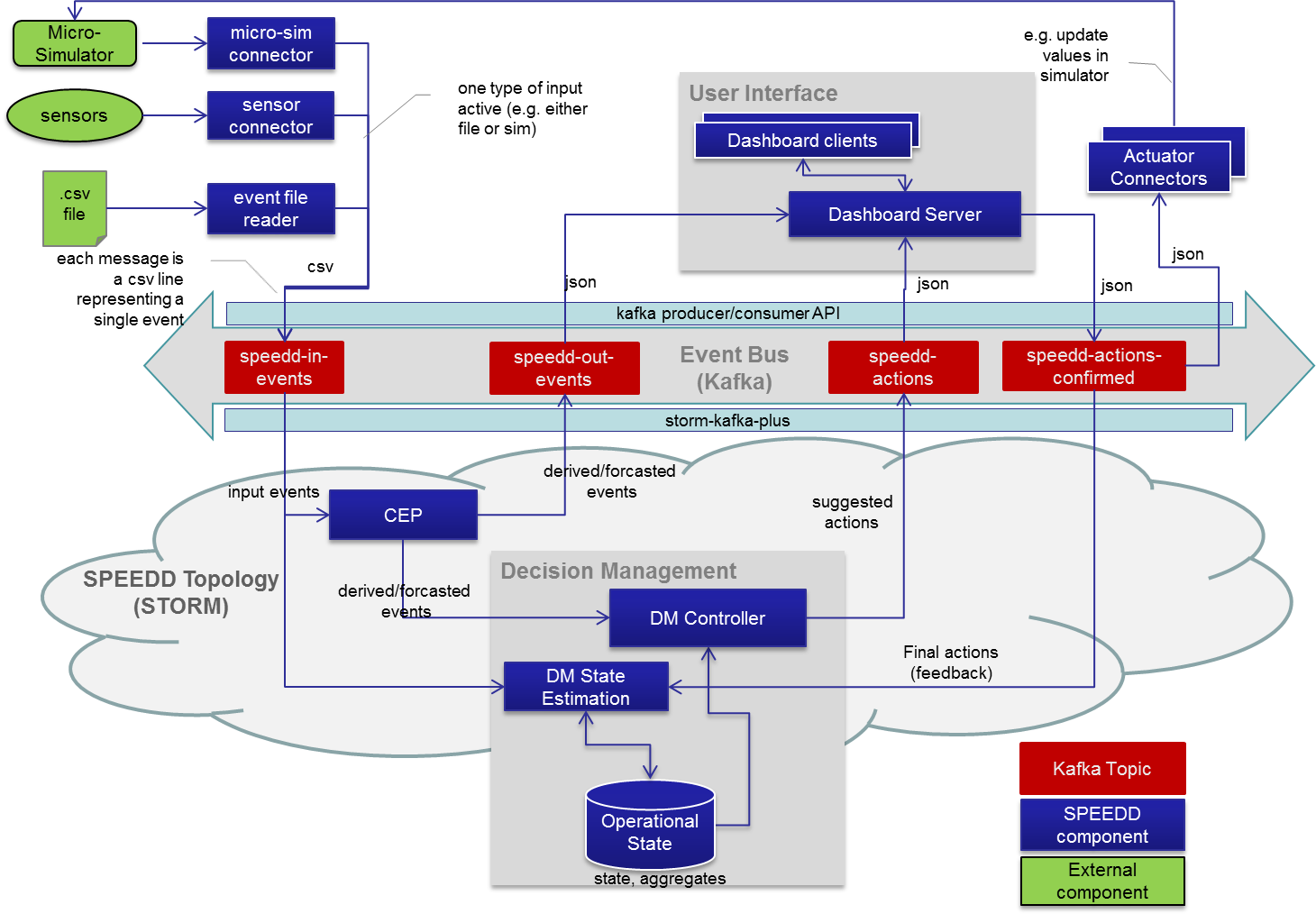


Figure ‎5.2 - SPEEDD Runtime - Event-Driven Architecture (Traffic Use Case)

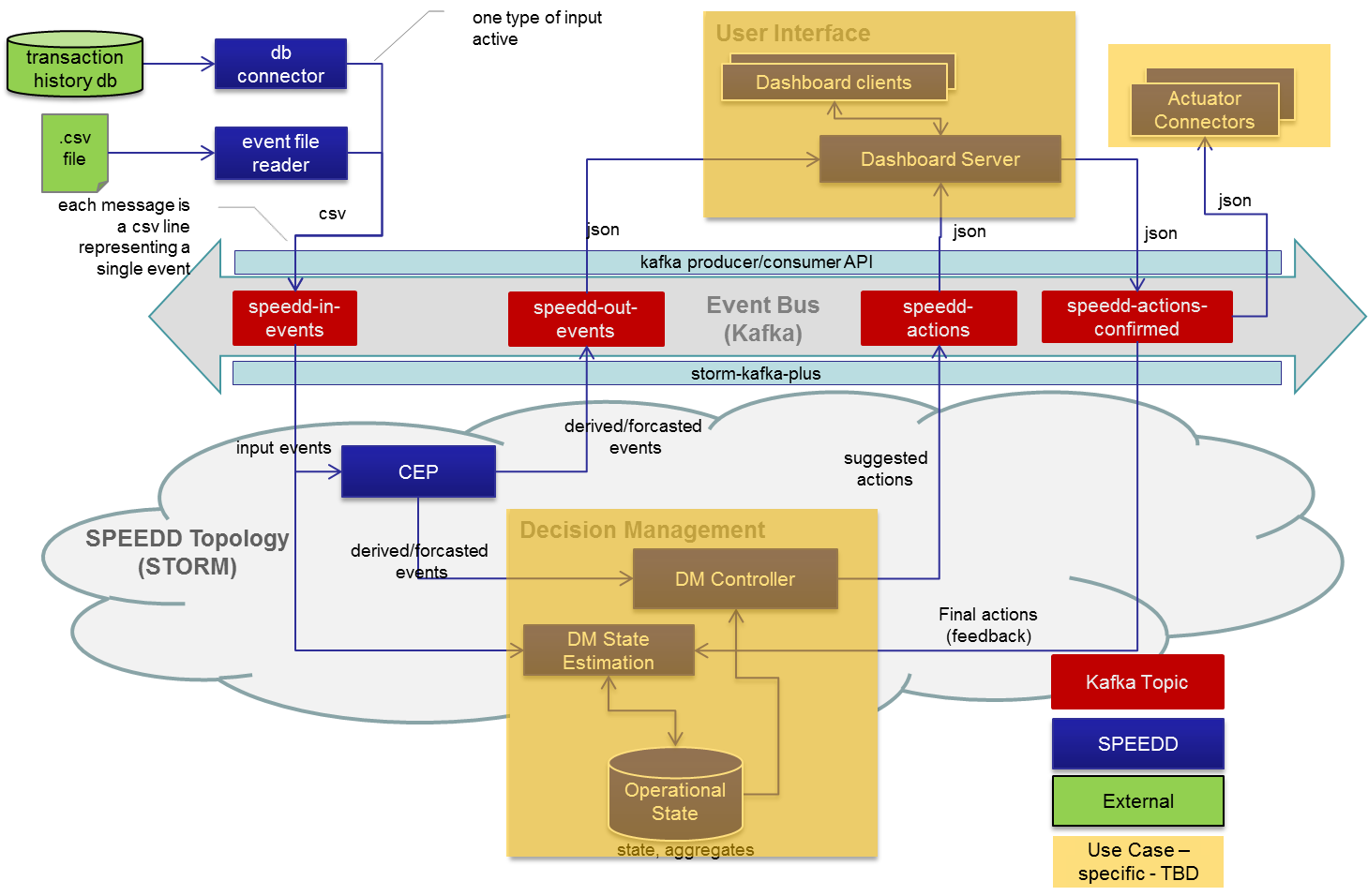


Figure ‎5.3 - SPEEDD Runtime - Event-Driven Architecture (Credit Card Fraud Use Case)

## Event Bus

Describe our implementation of the event bus using KAFKA technology.

## Event/Data Providers

Explain how we’re going to stream the data into the system, for each use case.

## Action Consumption – Actuators/Connectors

Describe who and how would consume action messages issued by the system. Per use case.

## Complex Event Processor

Describe the architecture of PROTON, including implementation on STORM.

## Decision Management

Architecture of the decision management component(s), approaches, issues, etc. Possibly discuss separately the design for every use case.

## Dashboard application

Describe the architecture of the dashboard application.

# Build-Time Architecture

Describe the build-time path and the architecture in details (based on the conceptual view presented earlier).

## Event Pattern Mining

Describe how machine learning approach is used to extract complex event patterns from annotated historic data.

## Authoring of CEP Rules

Describe the process, challenges, and approach to translation of the CEP patterns discovered in using machine learning into Proton EPN definition.

## Decision Management – the Offline Part

Should see if this is relevant. If it is, describe how decision management component is configured or adjusted based on the exploration of the historic data.

# Integration – APIs and Data Formats

Describe the integration mechanisms between different components and between the system and the outside world. List and describe the APIs and data formats in use.

# Non-Functional Aspects

## Scalability

Explain why is the proposed architecture is scalable. Describe how the system will scale up and out to match the load.

## Fault Tolerance

Explain what types of failures the system is designed to stand. Describe the designed behavior of the system in case of such failures.

## Testability

Describe the approach to testing the system. Address the functional testing as well as performance testing approach as designed.

# Summary

TBD – Summarize

Mention open issues and plans to evolve the architecture – refined to be submitted by M24

# Appendix A – Technology Evaluation

This could be one or more appendix parts. Here we’ll explain the approach, criteria, and the final choice of the technology stack that was made.

## Stream Processing – requirements and evaluation criteria

## Storm

## Akka

## Spark Streaming

## Choice of the Messaging Platform

1. Actuators are out of scope of SPEEDD prototype. Under automatic action we mean that the message representing the action type and parameters is emitted by SPEEDD, so that the actual operational system listening to action events is supposed to execute it. [↑](#footnote-ref-1)