TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MUHAMMAD SAJID HAROON

A COMMUNICATION MODULE FOR CAPTURING EVENTS IN ORDER TO MONITOR A SERVICE-BASED AUTOMATED PRODUCTION LINE

MASTER OF SCIENCE THESIS

Examiner: Prof. José Luis Martínez Lastra

Examiner and topic approved in the Automation, Mechanical and Materials Engineering Faculty Council Meeting on 05.06.2013

PREFACE

The research work presented in this thesis was carried out at Factory Automation Systems and Technology (FAST) Laboratory in Department of Production Engineering at Tampere University of Engineering and Technology (TUT).

The funding for the research work came from eSONIA: Embedded Service Oriented Monitoring, Diagnostics and Control: Towards Asset-Aware and Self-Recovery Factory.

I am highly thankful to Director FAST, Prof. José Luis Martínez Lastra for providing me the opportunity to work in the FAST lab, and for the guidance to achieve the research targets during my stay at FAST. Secondly, I would like to thank Bin Zhang, Hector Garcia, Luis Enrique and all staff members at FAST lab for their constant support and helping me out to accomplish my goals.

I would like to pay many thanks to Dr. Corina Postelnicu for her guidance, supervision and support at the time whenever I needed.

I would also like to thank all my friends in Tampere and back home for their best wishes, backing and assistance.

Finally, I would like to acknowledge and pay my deepest gratitude to my parents and all my family members for their prayers, care and kind support which helped me to achieve all my goals at every stage of my life.

Tampere, July 18th 2013
Muhammad Sajid Haroon

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master of Science Degree Programme in Machine Automation
HAROON, MUHAMMAD SAJID: A Communication Module for capturing Events in order to Monitor a Service-based Automated Production Line
Master of Science Thesis, 64 pages, 7 Appendix pages
July 2013
Major: Factory Automation
Examiner: Prof. José Luis Martínez Lastra
Supervisor: Dr. Corina Postelnicu

Keywords: Real-time Monitoring, SCADA, 3D Visualization, Unity3D, SOA, DWR, SPRING, WEB Applications, Web Services

The efficiency, reliability and on time maintenance of a manufacturing process largely relies on a highly efficient and rapidly responsive monitoring system. The increasing demand of uninterrupted continuation of a production process emphasises the need of an efficient real time monitoring mechanism of the process. The rapid advancements of modern technology especially in the communication field have largely affected every field of daily life as well as the industrial sector. The rise of wireless communication technology has made it possible to develop wireless sensors for industrial monitoring applications and revolutionize the monitoring techniques to a greater extent.

The work researches a web based monitoring approach for real time monitoring of service-oriented production assembly with 3D visualization. The implementation deals with the design and implementation of a communication framework for receiving, processing and publishing events information of a service oriented assembly line. The processed information is then linked and simulated with a 3D replica of the actual process over the web in real time.

The work demonstrates the usefulness of versatile features of 3D visualization in industrial monitoring applications. The online accessibility of the monitoring application enables all concerned individuals to access and monitor the manufacturing process in real time from any remote location. The developed web application can also be simulated for a given set of historical data. Currently, the research work focuses on capturing and simulating only two types of shop floor messages (Pallet activity notification message and Robot activity equipment change state message), but can be enhanced to include more features of the robotic assembly line in future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AJAX	Asynchronous JavaScript and XML
ANSI	American National Standards Institute
AOP	Aspect Oriented Programming
APE	AJAX Push Engine
API	Application Programming Interface
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CAMX	Computer Aided Manufacturing using XML
CATIA	Computer Aided Three-dimensional Interactive Application
CICS	Customer Information Control System
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
DCOM	Distributed Component Object Model
DOM	Document Object Model
DPWS	Devices Profile for Web Services
DWR	Direct Web Remoting
EJB	Enterprise JavaBeans
ERP	Enterprise resource planning
FIS	Factory Information System
FTP	File Transfer Protocol
HMI	Human Machine Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technologies
IMS	IP Multimedia Subsystem
J2EE	Java 2 Enterprise Edition
JAXB	Java Architecture for XML Binding
JAXP	Java API for XML Processing
JDOM	Java-based document object model
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JSF	Java Server Faces
JSON	JavaScript Object Notation
MES	Manufacturing Execution Systems
MIME	Multipurpose Internet Mail Extensions
MOM	Management Object Model
MVC	Model–View–Controller
RMI	Remote Method Invocation
ROM	Resource Object Model
SCADA	Supervisory Control & Data Acquisition
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SOM	Service Oriented Model
TCP	Transmission Control Protocol

URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WS-DD	Web Services Dynamic Discovery
WSDL	Web Services Description Language
WS-I	Web Services Interoperability
XML	Extensible Markup Language
XOM	XML Object Model
XPATH	XML Path Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

1. INTRODUCTION

1.1 Background

Monitoring of an assembly line or a production system has always been an important aspect in an efficient manufacturing system in order to achieve better productive outcome and enhanced performance.

An industrial unit with adequate, accurate and efficient monitoring mechanism results in improved production and supports the optimization of resource consumption. Monitoring not only helps to acquire understanding of normal production activity but also facilitates the identification and resolution of possible defects in order to improve efficiency of the production unit. Permanent monitoring of resources, processes and components is always required to avoid problematic situation.

For real time monitoring of the events at the shop floor, it is essential to have highly reliable and efficient data acquisition and some form of visualization facility to support the analysis of the gathered data. Most of the production industries deploy large HMI displays of the whole production unit, as in traditional SCADA systems.

Traditional monitoring systems are comprised of wired communication networks spread all over the industrial vicinity to provide fast and efficient communication between field devices and central monitoring & control (M&C) station. Field device I/O’s continuously send the information to the central control station through high speed communication networks for monitoring, assessment and control of the production activity in the field.

To ensure the amount of information collectible from the sensors is fully handled; significant focus has been granted in automation field to the remote visualization of the process data from field sensors. Conventional monitoring schemes like SCADA HMIs use 2D graphs and figures to manipulate the information which have limited graphical and interaction possibilities.

A monitoring system capable of representing production phenomena as real time 3D animation is strong support for the identification and localization of the production flow, while providing highly interactive and reliable supervisory control.

This thesis discusses an approach to real time web based monitoring and visualization, applied to assembly line production activity. The proposed solution relies on 3D graphical animation using Java, Unity 3D game engine and web tools.

1.2 PROBLEM DEFINITION

1.2.1 Problem description

The task is to model a production assembly line supported by web service based controllers via 3D animation and to demonstrate remote web based real time monitoring of the events occurring during production.

1.2.2 Justification of the work

“Powerful market trends of increasing product complexity, increased liability and warranty costs, and escalating competitive cost pressures all drive the need to streamline and improve the process of design verification and validation” [3].

Verification and validation indicates whether the design meets the specified requirements, or respectively if the right thing is being built. With the introduction of 3D mechanical drawings and CAD models along traditional 2D models, electrical schematics and programming soft wares, V&V processes have been largely improved. 3D virtual models results in faster and cost effective development of assembly products. Early detection of product imperfections while testing with 3D models ensures that design defects can be dealt with before it is too late, thus saving unnecessary overheads of time and cost. Potential benefits include increased product quality with lesser warranty expenditure, optimum prototyping costs and reduced design cycle times [3].

3D visualization largely enhances the impact and influence of any design scheme. Well documented 3D visualization and presentation of goods, services for the customers are rapidly becoming successful and favourable for companies. A well-designed, highly interactive and good quality animated model of a product or a product feature helps companies to easily market their product and making it more appealing for the customers.

Many companies usually approach their clients by presenting their product through 3D animations and virtual simulations, thus emphasizing its vitality and effectiveness in the practical usage that how this can be useful for the client. Moreover companies can help their clients by providing animation and visualization of the processes with specification and in order required by the client.

Nowadays, many 2D graphical representational techniques are commonly found in practice for representing various kinds of process data in industrial manufacturing systems. These include line charts, bar graphs, 2D diagrams and tables in Human Machine Interface (HMI). However, the increasing demand for vital information representation and efficient monitoring, these kinds of traditional visualization tools and techniques are unable to address the required needs thereby impairing fast error detection [1]. A 3D graphical model or animation model is much more intuitive and explanatory, capable of presenting greater insight information thereby facilitating faster and better information understanding of the situation at hand. Moreover, 3D visualization makes it possible to better the tremendous amount of input data, including spatial and chronological information.

Unexpected situations that might affect the production process are usually imposing time and financial costs to predict and track possible causes of the problems, by investigating huge historical data records of the production operation. In a production system, it is difficult for the operators or company personnel to continuously monitor the production operation outside the premises of the site because of limitations of traditionally available resources. Furthermore, conventional monitoring methods are not sufficient to provide real time information remotely located user, via the internet.

3D real time visualization can be helpful not only for rendering of different process functions in testing mode but for real world applications by introducing a sufficient communication means between the real world and the simulation of the 3D model. 3D model real time visualization can be even more useful if the designed model visualization is accessible instantly via internet from any location in the world.

The solution proposed in this thesis is addressing the above-mentioned issues. The adopted monitoring approach is web based therefore allowing the users to access the application remotely from any location without installing unnecessary supporting software utilities.

1.3 WORK DESCRIPTION

1.3.1 Objectives

The main objectives of the research work are as follows:

1. To design and implement a communication framework for receiving events from a service oriented robotic assembly line, processing the received information and exposing the results on the web.

2. To utilize the received information for animating the 3D model replica of the real world test bed setup on the web in real-time.

1.3.1 Methodology

The approach adopted to achieve the research objectives and goals is stated as below:

Review and discussion of real time 3D visualization based monitoring techniques especially focusing on web based 3D visualization techniques and application.
Perform an extensive study of possible communicating methods applicable to the test bed setting of choice through JAVA Eclipse®, receiving and parsing of information and for making it available to web applications.

Study of data acquisition techniques in web browser to access data from JAVA applications while forwarding the accessed information to Unity 3D web player for simulating the data obtained.

1.4 Thesis Outline

The thesis work consists of 5 chapters. Chapter 2 focuses on 3D model design and web services technologies, on one hand and software tools used in this research work, on the other. Chapter 3 introduces the test bed setup, detailing its major components and their roles. Chapter 4 discusses implementation details. Chapter 5 presents the conclusive results of the research work, overall system overview and outlines future work possibilities.

2. TECHNOLOGY OVERVIEW

This section is organized as follows. Section 2.1 presents a brief overview of some of the available 3D visualization tools, their importance and application domains. Section 2.2 highlights and compares commonly used 3D design technologies including CAD, CATIA and the game development engine Unity 3D. Section 2.3 gives an overview about 3D monitoring systems, tools and applications. Sections 2.4, 2.5, 2.6, 2.7 and 2.8 include a detail review of SOA and web services technologies including SOA design principles, WS architecture related details and properties. Section 2.9 discusses CAMX standards and other details. Sections 2.10 and 2.11 give an overview of use of Java tools and applications especially focusing on Spring frame work. Section 2.12 reviews some of the web tools including HTML, AJAX, XML and JavaScript. Sections 2.13 and 2.14 present a review about server data push and available data push technologies.

2.1. 3D Visualization Techniques

This section is an overview of some of the applications that have been adopted so far using 3D modelling and visualization techniques.

2.1.1. Virtual Simulation & Testing

3D design and simulation enables manufacturing engineers to create various components of the manufacturing process (e.g. work cells, robots, machine tools etc.) based on digital definitions of product. Machinery can be programmed offline and their behaviour simulated even for highly complex machine operations, before deployment of the machine in the actual production environment [2]. Figure 1 shows virtual 3D simulation of a robot operation.

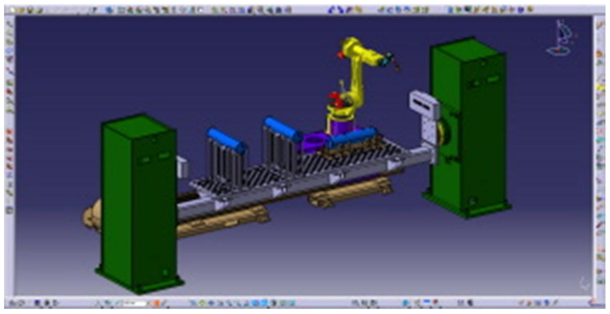
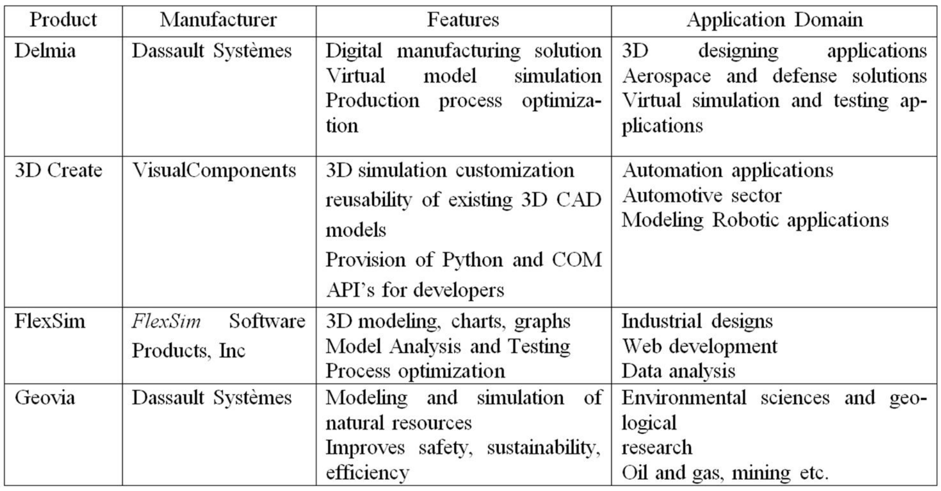


Figure 1: Virtual 3D simulation of a Robot Operation [2]

Some examples of simulation tools include DELMIA and Geovia by Dassault Systems, 3D Create by Visual Components, FlexSim by FlexSim Softwares etc. Table 1 shows some of the commonly available virtual simulation and modelling techniques is shown as under.

Table 1: Commonly used simulation softwares



Product	Manufacturer	Features	Application Domain
Delmia	Dassault Systèmes	Digital manufacturing solution Virtual model simulation Production process optimization	3D designing applications Aerospace and defense solutions Virtual simulation and testing applications
3D Create	VisualComponents	3D simulation customization reusability of existing 3D CAD models Provision of Python and COM API's for developers	Automation applications Automotive sector Modeling Robotic applications
FlexSim	*FlexSim* Software Products, Inc	3D modeling, charts, graphs Model Analysis and Testing Process optimization	Industrial designs Web development Data analysis
Geovia	Dassault Systèmes	Modeling and simulation of natural resources Improves safety, sustainability, efficiency	Environmental sciences and geological research Oil and gas, mining etc.

These tools proved to be very useful for simulating precisely an individual machine component, robot operation or even complete assembly line comprehensively and efficient. Figure 2 shows 3D model arrangement of robots designed in DELMIA.

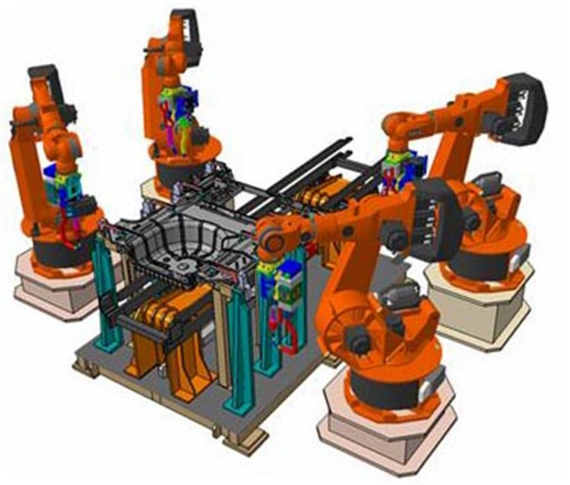


Figure 2: Robot simulation using DELMIA [4]

2.2. 3D Modelling

3D modelling is the representation of a physical object in mathematical form through 3D computer graphics soft wares as a result of combination of several points in 3D space, curved surfaces, triangles, circles and other geometrical shapes. Figure 3 represents a 3D design of a robot vehicle in 3d-canvas.

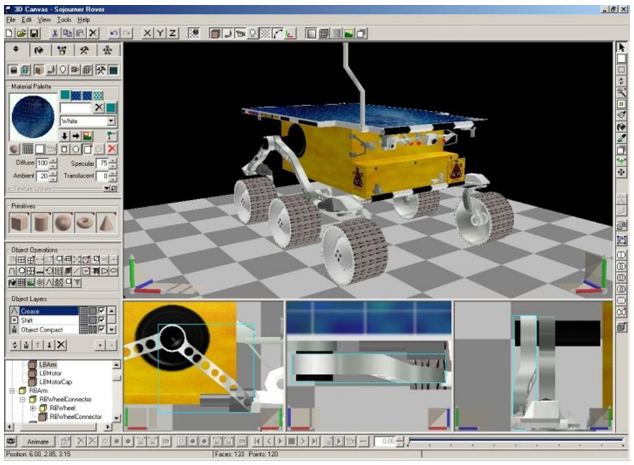


Figure 3: A 3D model of robot vehicle in 3d-canvas [53]

3D models and designs nowadays are widely used in a number of fields including medical research, aerospace and engineering research and development areas, entertainment industries, computer application, video games and largely in industrial design and automation applications. Some of the commonly used 3D modelling tools are briefly presented as following.

1) CAD

CAD or (Computer Aided Design) is creation of technical drawings with the help of computer design soft wares. CAD can be used to create, modify, analyse or optimize computer designs. CAD has found application in many areas including architectural, automotive, industrial and aerospace engineering and many others.

Unlike just designing or drafting drawings, a CAD design holds additional information regarding part dimension, material, tolerance level and many other application specific details. Together with animation, CAD is used to add special effects in multiple applications like advertisement sector, movies & games, entertainment and in technical drawings and models. Figure 4 shows CAD model of a robot operation.

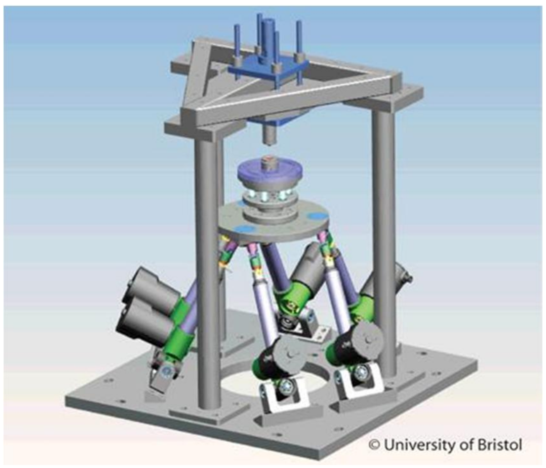


Figure 4: CAD model of a robotic operation [54]

2) CATIA

CATIA (Computer Aided Three-dimensional Interactive Application) is a commercial software suite developed by French aircraft manufacturer Dassault Systems. It’s a multi-platform supporting multiple phases of product development e.g. CAD (Computer-aided Design), CAM (Computer-aided Manufacturing) and CAE (Computer-aided Engineering).

CATIA provides designing and modelling solution from creation of 3D machine parts, tools to a fully functional production assembly line. It also facilitates to create, alter and validate even complex models and shapes with ease. In addition to drawing mechanical design and shapes, CATIA also has the capability to draw electrical, electronics and fluid systems efficiently. Figure 5 demonstrates CATIA model of Toledo Jeep Assembly Plant.

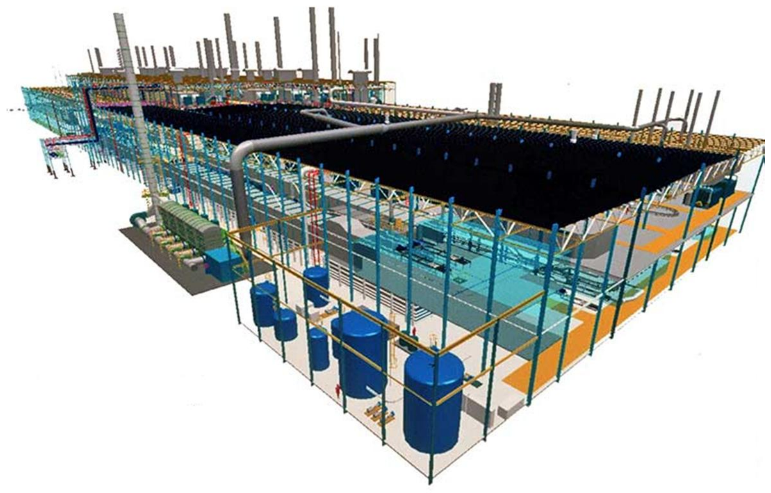
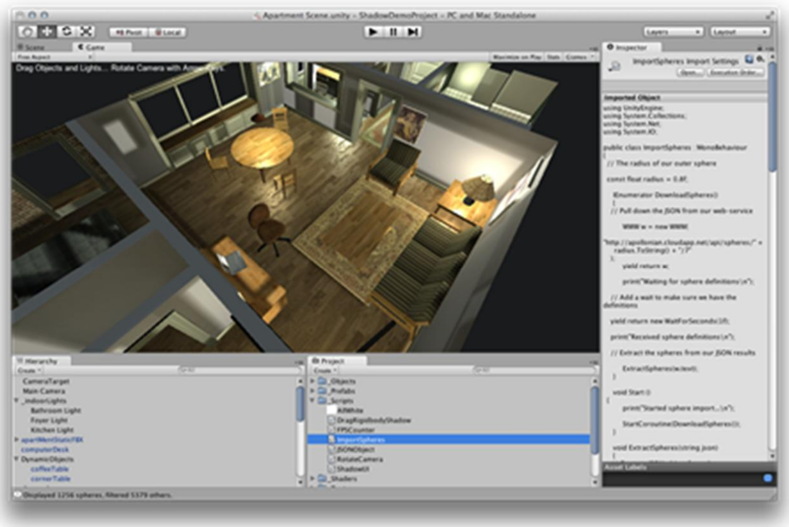


Figure 5: CATIA model of an assembly plant [55]

3) Unity3D GameEngine

Unity3D is a 3D game development tool capable of performing several other multi functions under the same platform. Unity presents an integrated designing tool for a high quality interactive real time architectural visualization with the compatibility of execution on almost all traditionally available operating systems and platforms. A typical Unity 3D scene is depicted by Figure 6.



```
using UnityEngine;
using System.Collections;
using System.Net;
using System.IO;

public class ImportSpheres : MonoBehaviour
{
  // The radius of our outer sphere

  const float radius = 0.8f;

    IEnumerator DownloadSpheres()
    {
   // Pull down the JSON from our web-service

        WWW w = new WWW(
"http://apollonian.cloudapp.net/api/spheres/" +
     radius.ToString() + "/3"
    );
        yield return w;

        print("Waiting for sphere definitions\n");

    // Add a wait to make sure we have the
definitions

    yield return new WaitForSeconds(1f);

    print("Received sphere definitions\n");

    // Extract the spheres from our JSON results

        ExtractSpheres(w.text);
    }

    void Start ()
    {
        print("Started sphere import...\n");

        StartCoroutine(DownloadSpheres());
    }

    void ExtractSpheres(string json)
    {
```

Figure 6: Typical example of a scene in Unity 3D [56]

Unity development editor is a virtual building tool, featuring an asset-aware architecture aimed (containing detailed descriptions and characteristics of available resources) to give a better development environment for building applications. A Unity application instance starts with a creation of scene in 3D space while objects are added to it by defining their size and specification. The objects behaviours and also the dynamics of the scene in a unity application are then managed through programming using Unity Scripts i.e. using JavaScript, Boo or C++ programming language.

CAD softwares have traditionally been used for a long time due to their easy to use design interface. But the demand for higher accuracy and multiple functions such as animation capability gave rise to the use of other 3D softwares e.g. CATIA, Pro/E etc. However the issue of supporting all design formats and simulation capability on commonly used user platforms still remained. Unity has a great advantage of importing and using 3D models and designs from other drawing soft wares like CAD soft wares, CATIA etc. without requiring any complex conversion mechanisms and any loss of information in the original design. Another unique feature of Unity 3D is Unity web player which enables application developers to run applications in the web browser, thereby facilitating inclusion of web features and functions in the applications.

2.3. 3D monitoring Systems

3D monitoring systems offers a wide range of advantages over 2D visualization based systems. There are on-going efforts to have 3D visualization based monitoring systems that are not only effective on factory floor but are also available remotely for all users via the internet.

Java3D is one of the tools that have been successfully used in applications to achieve real time web based monitoring with 3D visualization. The experimental approach presented by [9], uses Java3D API as development tool for an extensive variety of 3D graphical application while justifies its importance and benefit for the development of real time web based monitoring. Java 3D is also finding its applications in various applications such as designing 3D visualization and virtual environments [9]. An example of 3D modelling using Java3D is shown in figure 7.

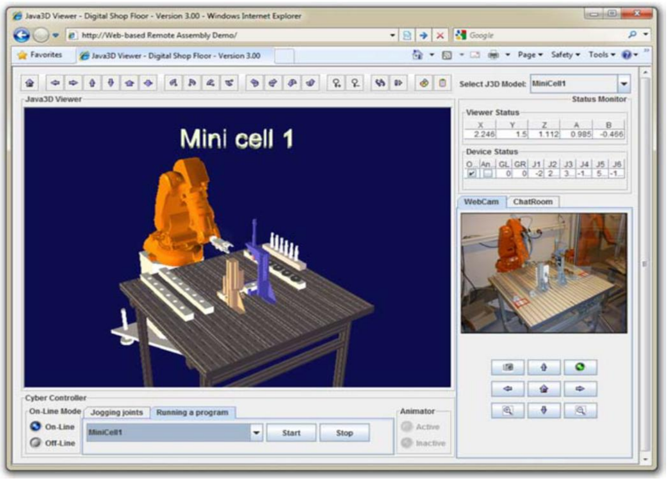


Figure 7: Remote assembly facility developed in Java3D [9]

Another 3D real time monitoring approach was implemented by [10]; to develop an event based 3D monitoring system to visualize shop floor-level state variations in real time while taking advantage of SOA (Service Oriented Architecture). The experiment uses mainly two tools for development in building the 3D based environment including Ogre (open source 3D graphics engine) and Autodesk 3ds Max, whereas DPWS (Device Profile for Web Services) is used as an infrastructure technology to address the tasks related to SOA and web services. Figure 8 demonstrates the 3D model of aforementioned monitoring approach.

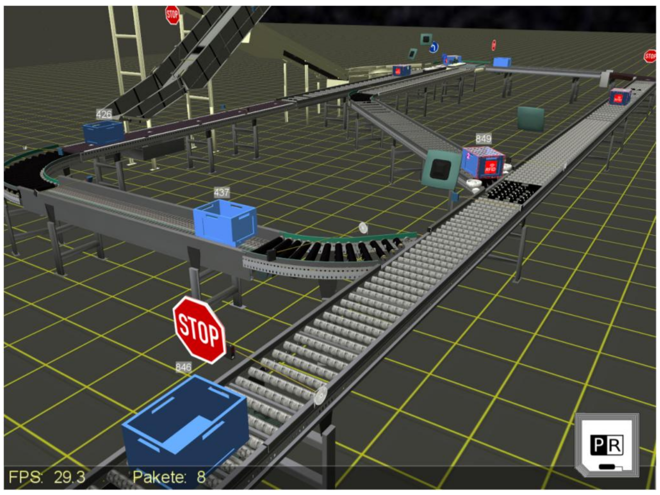


Figure 8: Event-based 3D-Monitoring of Material Flow [10]

2.4. Service Oriented Architecture (SOA)

The rapidly increasing advancements of modern technologies have revolutionized every field of daily life, has left a greater impact in the field of automated manufacturing as well. Several new tools and methods have been evolved that have largely modernized the traditional monitoring and control techniques in automation field. SOA’s have been implemented via Web Services for many years effectively offering several benefits such as autonomy, reusability, statelessness with fully documented interfaces. Due to their effectiveness and efficiency, Web Services are taken into account in the field of automation to improve the monitoring and control technique.

Service Oriented Architecture (SOA) is covering all aspects of building and using business process applications by furnishing an IT platform, allowing multiple applications to interact and share information in a business processes despite of their different operating systems and programming languages [5]. Although the idea of Service Oriented Architecture (SOA) is not new in the field of IT and communication technology, its flexibility to work with different execution systems extended its viability and application in multiple environments, providing users with system selection and tying them together under a consistent architecture. Adoption of SOA in fields like business and industrial sector gives companies a competitive advantage over other contenders, enabling them to react more quickly and efficiently to varying business and industrial needs.

In industrial and manufacturing level scale, SOA is an evolution of distributed control based on the request/response design model for synchronous and asynchronous applications.

2.4.1. SOA Design Principles

Following are the key design principles involved in SOA.

Abstraction	Service abstraction binds together many aspects of the services. Essentially, the stated principle underlines concealing as much detail of a service as possible. Doing so not only enable loosely coupled nature of services but also plays its part to conserve it. Abstraction level results in various forms of metadata during evaluation. The degree of abstraction applied may have an influence on service contract granularity leading a direct impact on ultimate expense and effort required for hosting the service.
Autonomy	Services have total control over the logic encapsulated within the service and each service is fully autonomous and independent of other nearby services in surrounding. The need of considerable control over its resources and environment during a logic implementation for a particular solution enables services to achieve their capabilities constantly and reliably. Service autonomy involves two distinct types of autonomy, design-time and run-time. Design-time autonomy directs towards autonomy of services evolution without effecting service consumers. Run-time autonomy reflects the control limit a service can have on its solution logic.

Loose Coupling	Loosely coupled design principle endures minimizing dependencies between services, service consumers and underlying service logic. The principle core emphasis is to promote independent design while maintaining interoperability with consumers of utilizing service capabilities. A number of couplings can be thought of during the designing of a service that can have an impact on service contract content and its granularity. SOA provides built in mechanisms to facilitate loose couplings between services and ensures that the service is decoupled from other components in protocol, time and location.
Reusability	An underlying solution’s logic is divided into services in order to maximize reusability. Reusability has vital importance within the service orientation and is considered as a core part in service analysis and design process. The remarkable advancement in the field of service technology has largely facilitated to achieve maximum gain out of reusability of multi-purpose logic on a distinctive level [6]. The actual idea of reuse is to employ services for multiple automated business applications instead of referring towards the frequent usage of services. The reusability feature makes it possible to use single service that can be used by several applications while avoiding the creation of new service for each application.
Granularity	Service granularity determines the overall quantity of functionality encapsulated by a service. A specific feature to determine a service’s granularity is its functional context which is often derived from one of three common service models. Services with larger quantity of functional context will have respective coarser service granularity. On the contrary, services with better and modest level of service granularity exhibit narrow or targeted functional contexts.
Composability	The ability for services to achieve service-oriented goals lies in their efficient and well-organized composition. Service compos ability enables service composers to solve big problems efficiently by distributing the task into conveniently handled smaller chunks. With the rising demand for sophistication in service-oriented solutions gave rise to the complexity of service composition. Service compos ability results in several useful outcomes specially design configuration which empowers a service’s reuse during the service composition.
Statelessness	Services tend to minimize resource consumption by being remained as stateless while stateful only when required. Like autonomy, statelessness is another nice feature favouring scalability and reusability. Ideally, a service is temporarily

stateful while receiving or processing a request message in certain application. Figure 9 presents a simple comparison of stateless and stateful services.

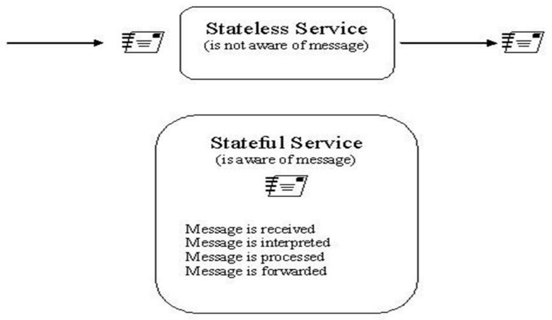


Figure 9: Stateless and Stateful services [58]

Service availability for many consumers at a time becomes difficult if it retains a state for a longer period of time.

Interoperability

Interoperability is a unique feature while discussing services referring towards its effectiveness and versatility. Service's interoperability helps to provide easy application integration. Interoperability provides scheme for wrapping existing applications which enables developers as well as consumers to access services through standard scripting languages and communication protocols. With this property, a service created using Java platform can be called up by any client application running on another platform, e.g. .Net. Since the said property has a number of advantages but at the same time requires huge amount of effort to resolve interoperability issues by going through frequent development and testing procedures to achieve the ultimate goals. For this purpose, Web (WS-I) was formed to address issues such as coordination, cooperation and other related subjects regarding interoperability.

Discoverability

Services are equipped with communicative informational metadata which helps them in discoverability and interpretation [6].In general, discoverability points to the process for searching web services for a given task. A web service description language (WSDL) and a web service endpoint are two fundamental requirements for web service consumers to discover access and interpret the information contained by the service. A common service discovery mechanism also known as Web Services Dynamic Discovery (WS-DD) uses a multicast communication protocol for discovering services over a communication network.

2.5. Potential Benefits of implementing SOA

SOA has presented numerous amounts of benefits in various fields of daily life. Some of the potential advantages are briefly discussed as under.

1) Architecture Flexibility

Service Oriented Architecture (SOA) and object-oriented design are quite similar in many ways. Both of these technologies implement same kind of functional implementation by communicating with functions, capable of performing certain functionalities. The architecture provides the flexibility for an application to act independently and stand on its own. Additionally, the architecture also takes into account the requirements of both service providers and consumers.

2) Agility

Since the variations in an industrial or business environment occur vigorously and rapidly, therefore requires an agile infrastructure to respond timely and rapidly. Generally an industrial production setup may require adopting new configuration based on product demands. With an SOA based approach, it’s easy to manage and reorganize the processes involved with minimal time and cost consumption. In a traditional industrial environment, applications are tightly coupled and largely dependent on various other entities while SOA is based on a decentralized working methodology. The so-called “loosely coupled” nature of SOA enables the services to be independently responsible for the services they offer which increases their ability to respond quickly and efficiently no matter what organizational changes occur in the system.

3) Cost effectiveness

Cost benefit is another significant outcome achieved together with agility and control through implementation of SOA in businesses. In situation where business using services, is required to be adjusted based on demands of new product, with services architecture can be rapidly and efficiently realized with merely involving business-level knowledge of the services. This ultimately reduces time, cost and effort for adopting new settings of the business process. Another benefit is the elimination of effort involved required in complex programming collaboration, technicalities and manpower utilization which is mainly replaced by available services knowledge and their functionalities in a business process.

4) Technological Aspects

Implementation of SOA offers a wide range of advantages from technological point of view. This includes businesses services flexibility across multiple platforms thereby increasing overall efficiency and bypassing requirement of compatibility schemes among platforms. Since SOA empowers services with location autonomy, hence enabling services to be independent of residing on any specific network or domain. The loosely coupled feature makes services in designing applications that are workable with multiple hardware types and operating systems. Additional benefits achieved through SOA implementation are services discoverability and dynamic connectivity with other services within surrounding or running on the same network.

5) Business Aspects

SOA has also a lot to offer from business point of view as well. With the inclusion of SOA in businesses especially in industrial sectors facilitates companies to meet customer's demands quickly and on time. Similar, it requires comparatively lesser amount of investments and resources while adoption or maintenance of technology. Another unique advantage that SOA exhibits is empowering existing technology resources for implementation, while at the same time minimizes reliance on expensive custom development

6) Supplementary Benefits

Some of the supplementary benefits of SOA can be regarded as interim and long-term context. The short-term advantages include improved reliability, lesser hardware acquisition cost, facilitation of operational and communication bridging among incompatible platforms [7].

The benefits that SOA serves in the longer run include composite applications development capability, an agile and self-healing infrastructure, ability to build real-time self-sufficient applications, providing a set of uniform classification of information across an organization, its customers and collaborators [7].

2.6. Web Services (WS)

As defined by World Wide Web Consortium (W3C), web services are described as:
"A web service is a software system to design to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically WSDL)" [18].

Web-services are platform and programming languages independent because they use XML language as standard. Most of the web services use HTTP as transfer protocol, since it is one of the most common, popular and widely used protocol. Each web service contains an associated file which holds the description/functionality of the relevant web service; also know a web services description language (WSDL).

2.6.1. WS Architecture

The basic web service architecture consists of combination of various technologies that builds up the so-called web services "stack". A detailed architecture further expresses the details of stack or implementation subsets in a broader perspective.
The main features depicted in WS-architecture include [18].

- Information exchange
- Web services description
- Web services descriptions discoverability and publishing

Following are some of the essential actions that should to take place for web services in a service-oriented environment [19]:

- The invocation methods and interface of a web service must be defined before its creation

- A web service should be published to one or more communications networks and prospective users must be able to locate it.
- A web service must respond when invoked by a user for certain function.
- When availability or usage is not required, the service may need to be un-published.

The three main participants in a WS implementation with their functionality are:

- Service Provider- the host implementing a particular service
- Service Requester- the entity requires accessing a service hosted by service provider
- Service Broker- the mechanism that supports the service publishing and discovery

Figure 10 shows a simplified form of Web Services (WS) Architecture.

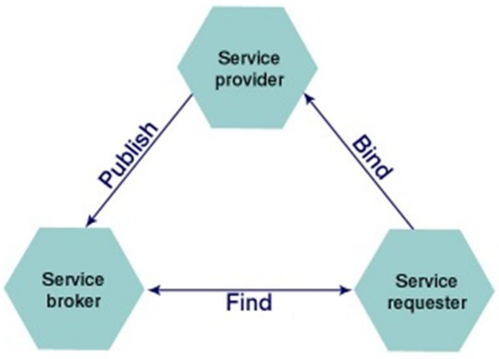


Figure 10: WS basic architecture [19]

The above said fundamental players of the architecture in reality represent software programmes also known as software agents. The basic web service architecture characterizes the interaction between those software agents associated with each entity of the architecture. All the information exchange involves three main functions find, bind and publish. These roles and operations act upon the web service artifacts: the web service software module and its description [18]. The service provider keeps the service's description and publishes it to a service broker (also known as service registry or service discovery) or service requester. The service requester finds the requested service description from service provider or service broker by using find operation, while interacts with the service provider and web service through bind operation.

The major elements of the basic architecture are in general defined by XML language (using XML standard data structuring and data types) whereas HTTP is used as standard protocol for information transportation.

2.7. WS Architecture models

According to World Wide Web Consortium (w3.org), the web services architecture can be characterized by four distinct models [18]. These architecture models are:

- Message-Oriented Model
- Service-Oriented Model
- Resource Model
- Policy Model

Message-Oriented Model

Message-Oriented model (MOM) highlights the main factors involving messages, message transport and structure of the messages. A simplified message oriented model is depicted by figure 11.

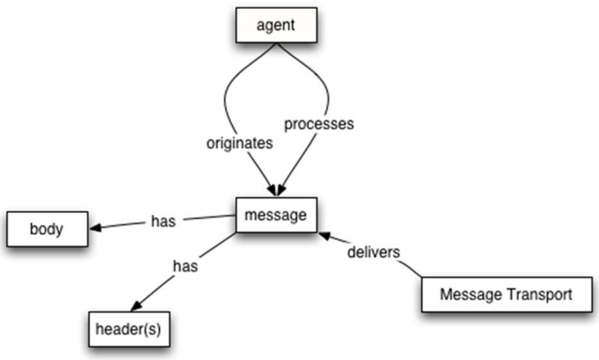


Figure 11: Simplified Message Oriented Model [18]

As illustrated from the figure above, the model represents agents capable of receiving and sending of messages, message transportation and message structure components (message body and headers). The main focus following this model is to deal with messages and their processing without taking into account their interaction with other messages or considering any semantic importance related to message contents.

1) Service Oriented Model

Service Oriented Model (SOM) is the most complex model of all of models that underlines the characteristics of services, actions and the messages enclosed within the services. Figure 12 presents a simplified service oriented model.

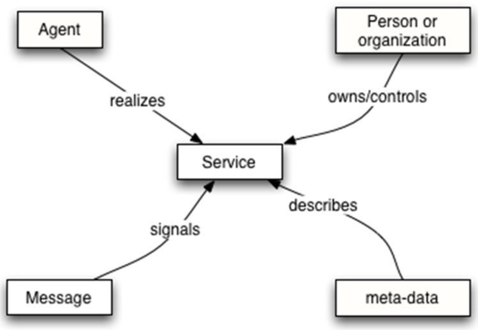


Figure 12: Simplified Service Oriented Model [18]

The above mentioned model depicts the features encircling services in a SOM. Services are realized and used by agents, by exchanging messages between requester and provider agents. In addition, services are meant to be used and provide their functionality to the real world (person or organizations). The meta-data in SOM represents the policy restrictions, transport binding, interface details and the semantics of the services.

2) Resource Oriented Model

The Resource Oriented Model (ROM) focuses on those aspects of the architecture that relate to resources, and are a fundamental concept that underpins much of the Web and much of Web services; for example, a Web service is a particular kind of resource that is important to this architecture [18]. A simplified resource oriented model is shown by figure 13 as below.

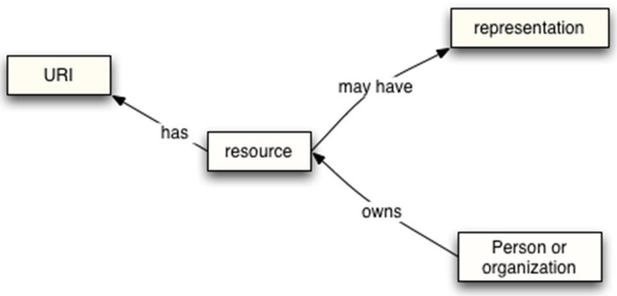


Figure 13: Simplified Resource Oriented Model [18]

The ROM highlights the key features of the resources, regardless of the functions performed by the specific resource in web services implementation. For example, a web service is a particular resource with its own significance within the model.

3) Policy Model

According W3C (w3.org) definition, “the Policy Model focuses on constraints on the behaviour of agents and services since policies can apply equally to documents (such as descriptions of services) as well as active computational resources” [18]. Figure 14 presents a simplified policy model.

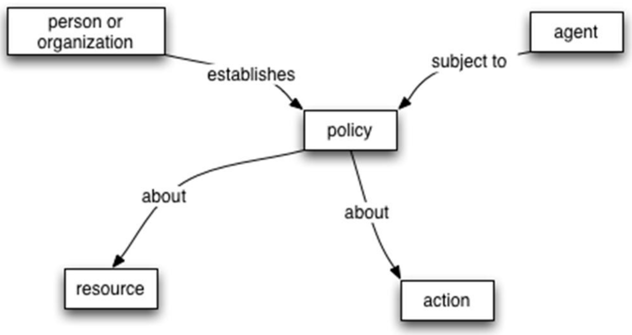


Figure 14: Simplified Policy Model [18]

The model obviously suggests that policies are mainly concerned with resources, for example agents may subscribe to the resources which are deployed and maintained by persons or organization. Policies on next level, address additional feature like security and quality of services. Security mainly focuses on constraints related to functional behaviours of actions and accessing resources, while quality of service deals with constraints on services.

2.8. Web-Services Architecture Stack

Web services architecture stack as shown by the figure 15, represents various layers of interrelated technologies. Each of the constituting layers, consist of a number implementation technologies families and their functions in a hierarchical manner. Web Services

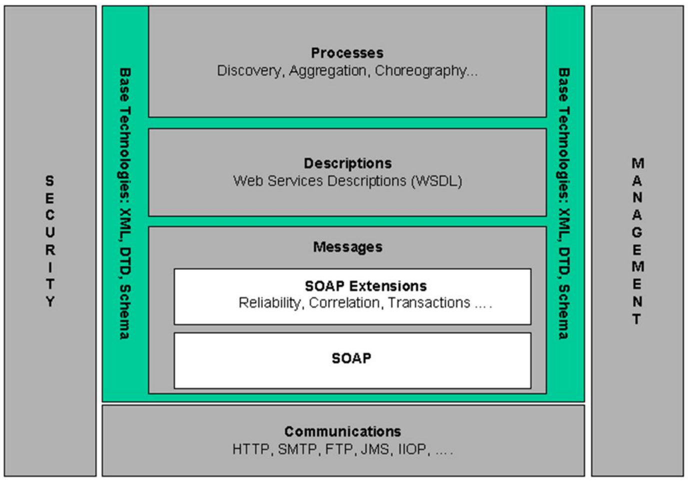

S E C U R I T Y
Base Technologies: XML, DTD, Schema
Processes
Discovery, Aggregation, Choreography...
Descriptions
Web Services Descriptions (WSDL)
Messages
SOAP Extensions
Reliability, Correlation, Transactions ....
SOAP
Base Technologies: XML, DTD, Schema
M A N A G E M E N T
Communications
HTTP, SMTP, FTP, JMS, IIOP, ...

Figure 15: Web Services Architecture Stack [18]

The technologies depicted in the stack furnish information regarding designing, building and deploying web services. The significant technologies which build the most critical part of the stack are messages i.e. Service-Oriented Architecture Protocol (SOAP), web services description i.e. Web Services Definition Language (WSDL) and standard formatting language, i.e. Extensible Mark-Up Language (XML). A brief overview of aforementioned tools is expressed in the following section.

2.8.1. XML

Extensible Mark-up Language (XML) derived from SGML (ISO 8879), is a simple, flexible and self-describing text format for encoding data for information interchange both for machines and humans.

XML provides a mean of programming documents in a way which is conveniently easy to understand both for humans and machines.XML proved to be a highly flexible programming format for representing and storing information, successfully implemented in various IT and business applications. The use of XML technology results in wide range of benefits including, an efficiently management of web information system and organizing digital resources, easy and dynamic processing of user's complex information requests and information extraction, virtual, supports interoperability and integration between multiple information systems and applications [22].

2.8.2. SOAP

Definition of Simple Object Access Protocol (SOAP) according to W3C organization states: "SOAP 1.2 provides a standard, extensible, composable framework for packaging and exchanging XML messages" [18]. The basic aim for designing SOAP is to find an alternate to traditionally used remote communication methods like CORBA, DCOM and RMI [26].

SOAP uses XML for messaging whereas HTTP or SMTP for communication. SOAP provides the protocol information and message framework in building process of web services. Although SOAP provides a number of advantages like interoperability and universality while at the same reduces its application for larger applications due to its considerable functional limitations [24]. Below given figure 16 presents a typical SOAP message envelope structure.

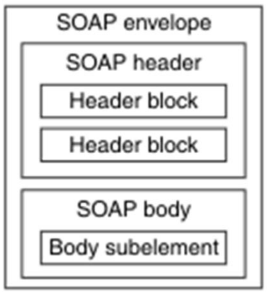


Figure 16: SOAP message Envelop [59]

A SOAP message envelope consists of two major parts, SOAP header and SOAP body. SOAP header contains XML structured application related information that is processed by the message provider. SOAP body is also XML formed application specific data that effects the processing of application specific information.

One of the main advantages of SOAP is interoperability. SOAP is considered as a versatile protocol capable of supporting various protocols. Although the SOAP standard stack utilizes HTTP for transportation, while at the same time provides operational usability of JMS and SMTP as well.

2.8.3. WSDL

According to World Wide Web Consortium (w3.org); "A WSDL document defines services as collections of network endpoints, or ports". AWSDL document contains abstract definitions of messages and endpoints separated from concrete service and data format binding. This kind of classification allows reusability of abstract definitions like messages (information interchange description) and ports types (abstract collections of operations).Similarly, the concrete protocol and data format specification for a certain port type results in binding reusability [25].

A WSDL is essentially an XML based language having the description and access detail of web services. A WSDL document for convenience is divided into two major sections, ‘abstract’ and ‘concrete’ as shown in figure 17. Specifying the two sections further enhances the flexibility and reusability of web services definition.

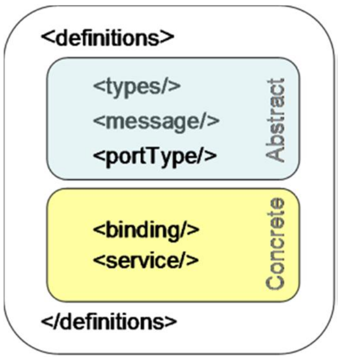


Figure 17: A general WSDL structure [60]

In a Service-Oriented environment, the WSDL portrays the functional features of a service indicating what type of messages can be send and receive by a service. The service’s point of view explains this kind of message exchange [27].

The six critical building elements in a WSDL document with a brief functional explanation are revealed as below.

Types	Describes the data type definitions defined in XML schema definition (XSD)
Message	Contains an abstract definition of informational data being communicated
PortTypes	Set of operations supported by some endpoints
Ports	Shows a binding address, referring to a particular communication endpoint
Operation	Description of the action performed by web service
Binding	Holds the protocol and data format information for operational detail (operation, messages) specified by portType
Service	A set of associated endpoints

A single WSDL contains the functional descriptions of a service and focuses only the same service. By grouping messages into operations, a WSDL tells what messages a service can sense and receive and how they are related to one another [27].

2.9. CAMX Standard

"CAMX (Computer Aided Manufacturing using XML) refers to set of standards that specify web-based communication protocols for the electronics production industry, intended to provide factory-wide communications for information-intensive manufacturing systems" [28].

CAMX was initially developed by the International Electronics Manufacturing Initiative (iNEMI) and IPC along with collaboration of electronic production industries to address information exchange between manufacturing equipments and electronics manufacturing shop floor.

As the name suggests, CAMX utilizes XML data format for all of the IPC-25xx messages descriptions. CAMX consists of the IPC-25xx standards family, which are approved by IPC, Association Connecting Electronics Industries, an ANSI accredited standards body having thousands of member companies and individuals [31].

2.9.1. CAMX standard family

CAMX standard family defines four major types of standards.

IPC-2501	"The IPC-2501 standard establishes the governing semantics and an XML based syntax for shop floor communication between electronic assembly equipment and associated software applications" [32].
IPC-2541	The IPC-2541 standard sets up the information interchange requirements between Factory Information System (FIS) and electronic manufacturing software.
IPC-2546	The IPC-2546 standard sets up the information interchange requirements between Factory Information System (FIS) and shop floor assembly equipment.
IPC-2547	The IPC-2546 standard sets up the information interchange requirements between Factory Information System (FIS) and shop floor assembly equipment inspection.

2.9.2. IPC-2541

According to IPC — Association Connecting Electronics Industries, the IPC-2541 standard can be defined as "an XML encoding schema, which enables a detailed definition of electronics assembly, inspection, and test equipment, messages to be encoded at a level appropriate to facilitate plug-and-play characteristics in a factory's shop-floor information system" [33].

The IPC 2541 presents a general event message content and requires to be used along with IPC-2540 standard sectional documents, a CAMX Shop floor Equipment Communication standard series defining an XML based encoding format for comprehensive specification for electronic inspection and test equipment messages.

IPC-2541 standard enlists equipment state model to furnish important equipment/machine state information that determines the equipment's availability and overall utilization during its period of operation.

The CAMX IPC-2541 equipment state diagram is given by figure 18**.**

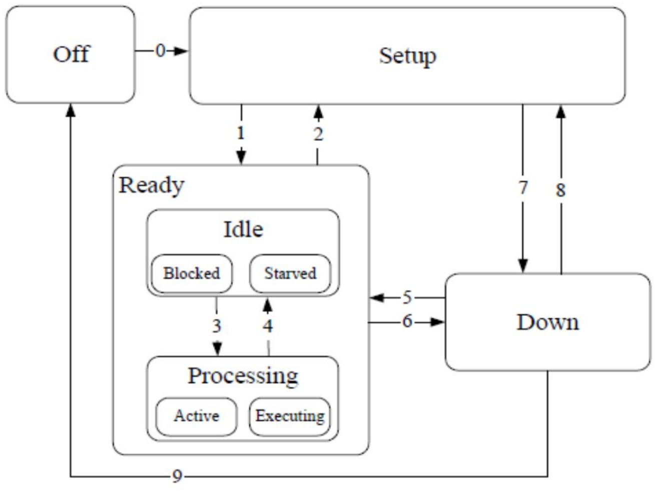
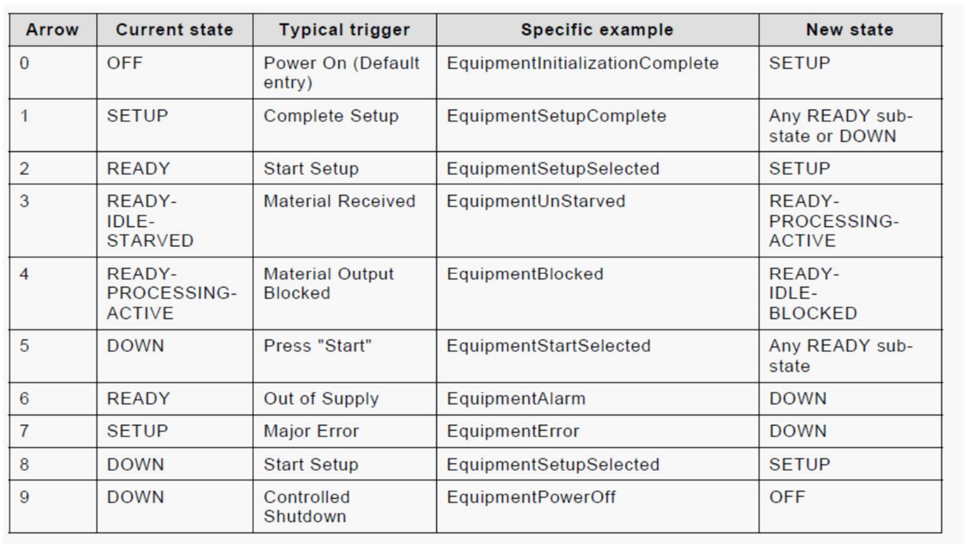


Figure 18: CAMX Equipment State Diagram [33]

The CAMX equipment state model aimed to address the following objectives [33]:

1. Presenting an equipment state model which is capable of defining states relevant to the test, inspection and electronic assembly industry.
2. An equipment state model having least number of states which are vital for monitoring and control purposes in a process.
3. A model with in advance declared states so as to avoid any alterations in basic states during implementation.

The equipment's state transitions are result of several triggering factor like alarm, system input or operator command. Figure 19 shows state transition table for equipment state model [33].



Arrow	Current state	Typical trigger	Specific example	New state
0	OFF	Power On (Default entry)	EquipmentInitializationComplete	SETUP
1	SETUP	Complete Setup	EquipmentSetupComplete	Any READY sub-state or DOWN
2	READY	Start Setup	EquipmentSetupSelected	SETUP
3	READY-IDLE-STARVED	Material Received	EquipmentUnStarved	READY-PROCESSING-ACTIVE
4	READY-PROCESSING-ACTIVE	Material Output Blocked	EquipmentBlocked	READY-IDLE-BLOCKED
5	DOWN	Press "Start"	EquipmentStartSelected	Any READY sub-state
6	READY	Out of Supply	EquipmentAlarm	DOWN
7	SETUP	Major Error	EquipmentError	DOWN
8	DOWN	Start Setup	EquipmentSetupSelected	SETUP
9	DOWN	Controlled Shutdown	EquipmentPowerOff	OFF

Figure 19 : State Transition Table for Equipment State Model [33]

2.10. JAVA Applications in Industrial Automation

Being an object-oriented platform independent language, Java quickly got appreciation in conventional enterprise application field as well as attention of researchers in manufacturing area [34].

Since Java by concept is an open platform, allows java applications to be executed over numerous hardware and operating systems. Although use of Java is already been in practice in various applications of higher levels of automation pyramid (Such as MES, SCADA or ERP) since requirements are analogous to conventional software applications, however at lower/field level the usage is still very rare [35].

Use of Java in industrial application is endorsed by various potential benefits offered by Java platform. Some of the noteworthy outcomes are:

- The so-called ‘Write Once, Run Anywhere’ (WORA) concept enables an application programmed on one platform, can run on other systems as well [35]**.**
- The elementary Java properties like polymorphism, inheritance and encapsulation empowers code reusability.
- Java provides the facility to easily build applications to be executed over internet enhancing its capability and versatility.
- The advanced field devices featuring alarming, configuration and maintenance are using Java applications that can work with a number of available communication protocols like HTTP, SMT and FTP [35].
- The built in features like automatic memory allocation, error recovery etc. adds up to efficient software development with Java.
- The wide-ranging and easily accessible networking capabilities of Java largely reduce the complexity of distributed systems since the vital requirements for developing those systems are already available in the language [35].

2.11. Spring Framework

“Spring is an open source framework created to address the complexity of enterprise application development [36]”. Spring framework was introduced to offer ease for programmers in application development in java platform. Spring framework is a multi-layer platform supporting both Java and Java Enterprise (J2EE) applications development. The framework uses java beans and is preferred for development because of being lightweight and having lower processing overheads. Figure 20 highlights salient features of Spring Framework.

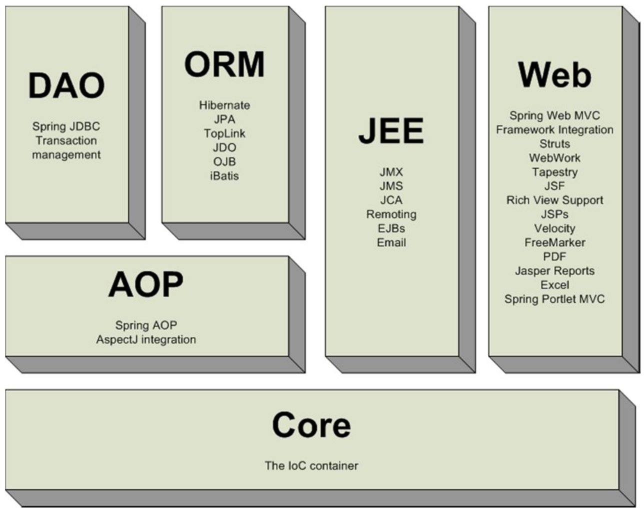


Figure 20: A brief overview of Spring Framework [61]

Spring framework contains several distinct modules that are built on aspect oriented programming and dependency injection, enriched with extensive amount of useful features required for enterprise application development [36].

2.11.1. Architectural benefits of Spring Framework

Following are some of the architectural benefits that Spring framework implementation offers as stated by [37]:

- Spring efficiently provides not only an ease for organizing middle layer objects but also configurationally management services at any architectural layer regardless of the runtime environment selection.
- Spring eradicates the propagation of singletons, which cause reduction in testability and object orientation.
- Spring facilitates in solving problems without using EJB and provides an alternative to EJB which is suitable for many applications, e.g. AOP (aspect-oriented programming).
- Spring uses JDBC or any other mapping product like Hibernate or a JDO implementation for offering a reliable data access framework.

- Spring provides an easy to program framework, reducing programming cost to a minimal level. In addition, the applications developed using Spring depend on few of its API's as possible
- Spring provides reliable and an easy to program model for many applications. Spring approach towards various API's including JDBC, JMS, JavaMail and JNDI confirms the said statement.

2.11.2. Spring WS (Web Service)

Spring web services, is an open source framework, designed to build document-driven web services. Spring Web Services aims to support contract-first SOAP services development, allowing flexible web services building using various techniques to manipulate XML payloads [38].

Web services creation mainly adopts either of the two development styles; contract first or contract last. In contract-first web services approach, a WSDL contract is created first while Java is used to implement the contract while on the contrary, in a contract-last approach, a Java code is written first that later on generates the WSDL[38].

Spring web services present both server and client side support, for creating and accessing contract-first web services. A server side support is achieved by creating a MessageDispatcher, which forwards the incoming messages (XML) to endpoints with configurable endpoint mappings. The endpoints, usually annotated as '@Endpoint', hold various handling methods and endpoint interceptors. These handling methods are annotated as '@PayloadRoot', each capable of handling certain parts of incoming XML messages and generating response if required. Figure 21 demonstrates the request processing workflow in Spring Web Services.

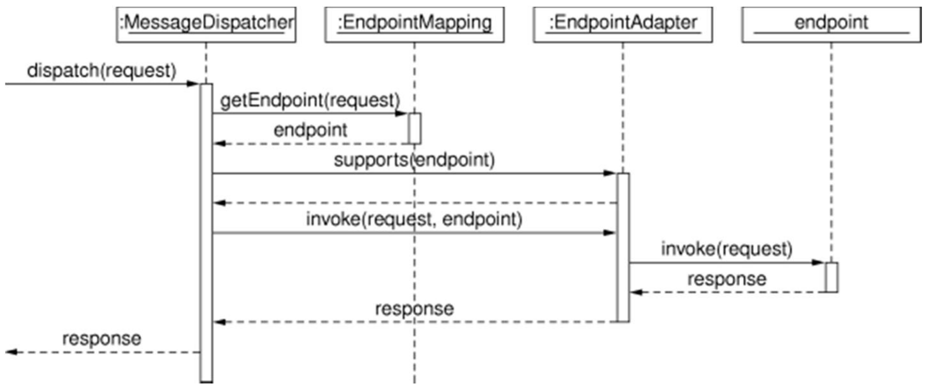


Figure 21: The request processing workflow in Spring Web Services [38]

Several XML handling and parsing alternatives are available with Spring WS. Spring WS supports a large number of XML handling libraries for endpoints including DOM family (W3C DOM, JDOM, dom4j, and XOM), XPath for message parsing, and marshalling techniques for converting XML to objects such as JAXB, Castor, XMLBeans, JiBX, or XStream [38].

2.11.3. Spring MVC

According to SpringSource [38] “the Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale and theme resolution as well as support for uploading files” [39].

Spring MVC follows typical MVC pattern to formulate an application into three distinct layers [40].

- Model: Objects signifying data in an application. The model includes the informational data about the service host or service client.
- View: Objects responsible for data presentation for the user. The view displays the outcomes of application into visual interface element, webpage, graph, list, table etc.
- Controller: Processes responding to events, usually user actions. A controller provides actions and means that can modify or customize the model object.

As like other web MVC frameworks, Spring's web MVC is essentially a request-driven framework, where a central servlet dispatches incoming requests to controllers and offers necessary functions and tools required for web applications development [40].

The DispatcherServlet is absolutely integrated with the Spring IoC container; versatile in functionality makes it possible to fully utilize almost all of the Spring features. Spring MVC involves three major components.

- Model & View: Holds the model and view objects of Spring MVC
- DispatcherServlet: Acts as a front controller, configured through an XML file (web.xml). All the desired URL patterns are mapped to DispatcherServlet.
- Controller: Acts as FormController. Shows the validation errors or success submissions on the form view.

The request process workflow in a Spring Web MVC application is shown by the figure 22. The DispatcherServlet acting as Front controller encounters the incoming request and directs it to the controller. The controller dispatches the request, processes the contents through certain commands and actions, creates a model object and forwards it again to the front controller. The front controller then passes on the model data to view template and receives the response. Finally, the front controller returns back the view response to the application requestor usually in the form of a webpage.

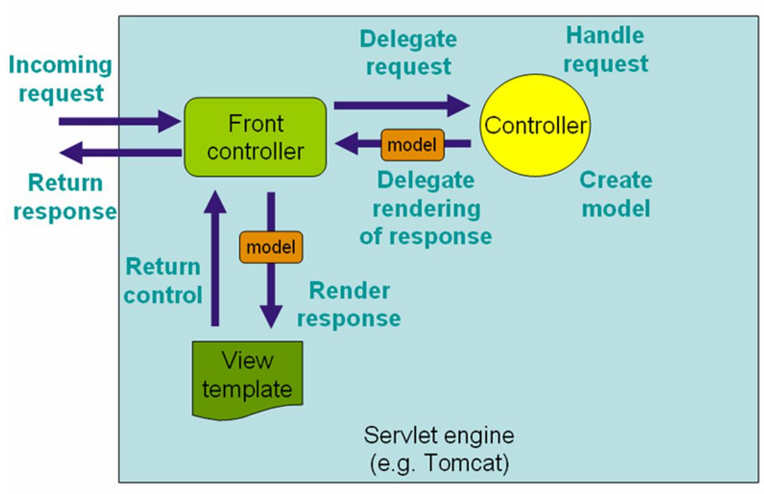


Figure 22: The request processing workflow in Spring Web MVC [39]

Spring MVC has the capability to integrate with other web frameworks like Struts, JSF (Java Server Faces) and WebWork. Recent developments have also made it possible to enhance the view technologies with the addition of new features like excel sheets and pdf documents.

2.12. Web Applications

Web applications, as the term expresses, are those applications which are located and accessible for remote users over the web. Perhaps one of the major advantages of building and deploying web applications is their availability of easy access for all users, irrespective of their operating platform or accessing methods (e.g. browsers types). This is because those applications are mainly created using HTML and JavaScript, which are supported by a wide variety of available web browsers.

Web applications are becoming more and more common in everyday life especially in e-commerce, online trading and hundreds of other fields, because of being convenient and an inexpensive way of providing detailed product information and online buying facilities to the online users [41].

The web is a rapidly growing technology that has heavily revolutionized communication and information access, whereas an increasing use of web is extended from individuals to thousands, industries and organizations [44].A java based web application consists of two technologies: communication protocol HTTP (Hype Text Transport Protocol) and Java API. HTTP is essentially a request/response based protocol while generally the communication takes place over TCP/IP connection. Generally, a client request to the server involves many factors including protocol version, request modifiers contained in a MIME-like message, resource URI, information about the client and possible body contents. Similarly, the server response also comprises some information such as an error or a success code, protocol version of the message, server information contained in MIME-like message and possible body content [42]. One of the major issues faced by web applications is of network traffic and the frequency of network interactions. The applications developed using Extensible Mark-up Language (XML) along with XSLT, have the capability to be more compact than HTML and largely reduce network traffic

The Java Servlet is an important API facilitating a simple framework in building applications that run on web servers [43]. Java Servlet API is a programming interface (API), aimed to expose HTTP to java platform and facilitate building web applications. Java Servlets are server-side java based entities that utilize HTTP interface to receive incoming client requests. A servlet engine captures the HTTP requests and sends the servlet output response back to the requester client.

2.12.1. HTML

HyperText Mark-up Language or HTML essentially a tag-based language is considered as the building block of web applications [44]. Rapid emergence of new features and introduction of novel tools in HTML has largely revolutionized the field of information and communication technologies (ICT).

The reason for HTML to be more common in web development is because of being globally acknowledged programming language, user friendly programming architecture, free to use and requires no plug-ins or any specific development platform, supported by all available web browsers, easy to understand and use, easier to identify errors and update webpages.

Recent inclusion of HTML5 in web development has largely facilitated web designers offering more powerful and unique features. The other web tools like JavaScript and Cascading Style Sheet (CSS) are also more efficiently integrated with HTML5.Presently combination of HTML 5 along with CSS is the most suitable technology for developing powerful web applications significantly enhancing web accessibility [44].

2.12.2. JavaScript

JavaScript is a lightweight object-oriented web scripting language and key element in web development. JavaScript is considered as the most prominent client-side programming language adding dynamic elements and interactivity to web pages [44].

In the past few years, JavaScript has emerged as a great web tool providing newer tools and method, enhancing framework simplicity and efficiency. In addition, JavaScript provides various multimedia features like allows playing visual and audio media files on webpages, textual and graphical animation and various data representation techniques. JavaScript make possible for the programmers to change web page dynamically at run time, thereby allowing more rapid and in time websites update

Generally a JavaScript environment involves three main active participants [45].

- JavaScript engine: Required for successful JavaScript code execution
- JavaScript context: Consists of all type of objects defined by JavaScript or JavaScript standard
- Host objects: The objects furnished by the host environment, e.g. DOM etc.

2.12.3. AJAX

Asynchronous JavaScript and XML (AJAX) is a client-side tool meant to communicate server-side script, in order to retrieve and use server-side data content. An
An AJAX Application commonly based on JavaScript, is a form of dynamic web application specified at a known URL presenting dynamically changing states to the user through UI events [46].

AJAX by nature is "asynchronous", which allows it to offer its functionality even without refreshing the webpage. AJAX itself is not a new technology rather it's a cluster of various other technology, each having its own specific functions and features. The technologies included within Ajax domain include JavaScript, XML & XSLT, XMLHttpRequest, XHTML & CSS and DOM. AJAX also offers to work with a variety of generally used data formats including text files, JSON, XML etc.
The general AJAX architecture is depicted by figure 23.

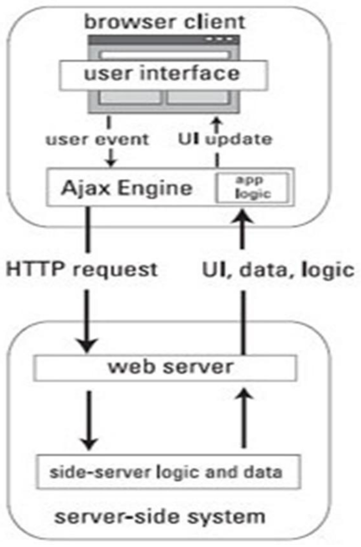


Figure 23: Ajax Architecture [62]

AJAX introduces a mediator between the client and server application, named as AJAX engine. AJAX engine, allows web application to run asynchronously without waiting for response from server. The browser loads the Ajax engine at the start of the session instead of loading the webpage. The engine is solely responsible for users' interface rendering and correspondence with the server. A JavaScript call to the Ajax engine is resulted every time the user action causes an HTTP request, whereas the engine handles user response by itself if does not require to send a reply to the server [47].

2.12.4. XML

Extensible Mark-up Language (XML) is a tag based mark-up language for representing, storing and transporting data. The data format supported by XML is both human and machine-readable. XML offers simple, platform independent, easy to understand and parse data format. Its generality, flexible data structure and reusability make it an ideal candidate for web use.

The combination of XML with different web technologies results in rendering and processing data differently among users, tools and adaptive technology [44].

XML has found a great deal of applications over the internet. Its self-describing data structure behaviour has immensely attracted developers to employ XML in various web applications. Some of the considerable advantages achieved in XML web applications include [48]:

Data interchange	Use of XML has largely simplified the data interchange between applications. Since various applications are built with their own set of tools, this makes them difficult to communicate with other applications. The problem can be overcome by transforming the applications internal data formats into XML thereby facilitating the information interchange.
Smart Code	The simplified and self-descriptive nature of XML documents makes it easy to identify and extract any useful information, hence directs writing smart code for XML processing without human intervention [48].
Smart Searches	Another great advantage of XML based documents is their capability of easy searching and understanding. Unlike any other scripting or tag based language, data extraction and interpretation from an XML document saves time with minimum chances of erroneous results [48].

2.13. Server Data Push

Data pushing technologies are the ones which are used to push data from a server to client. This data push actually follows a publisher/subscriber communication model. Therefore, firstly it's essential for the subscriber to subscribe to the server for retrieval of any updated information. In the same way, whenever there is a new data content, server publishes it to all the subscribers who have accessed it. Sometimes it's also required to push data from server to client after certain period of time. Three commonly used data pushing modes are:

- Streaming
- Polling and
- Long polling

1) Data Streaming

Streaming is the most advanced data push technique in which a data transfer from server to clients takes place for undetermined interval. In data streaming scenarios, data should be pulled by end users rather than pushed to computational resources in the form of

streams of tuples, and processing is continuous over these streams as if data were always available from local storage.
A general data streaming working is shown by figure 24:

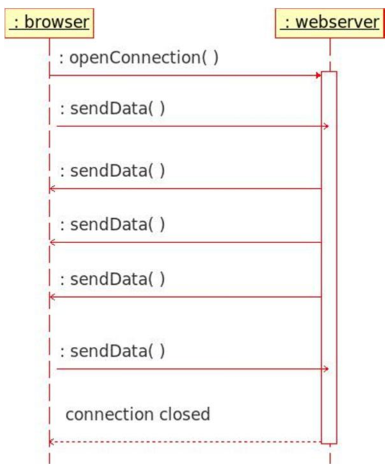


Figure 24: Data streaming [49]

Generally tuples are live and are considered as the most recent data packets received during the transmission process. Data streaming is often termed as forever response that takes place over HTTP. The statelessness of HTTP protocol causes the connection to close the connection between server and client after each response.

In order to evade data overflow where data processing is slower than data transmitted per tuple, the data transmission should be intermittent instead of continuous. The major advantage data streaming results is the availability of information without delay.

2) Polling

In polling, the connection between client and host is terminated each time the response from host ends. During data polling, the client application requests for data from server at specified fixed time intervals. Generally, a polling data flow is symbolized as shown by figure 25.

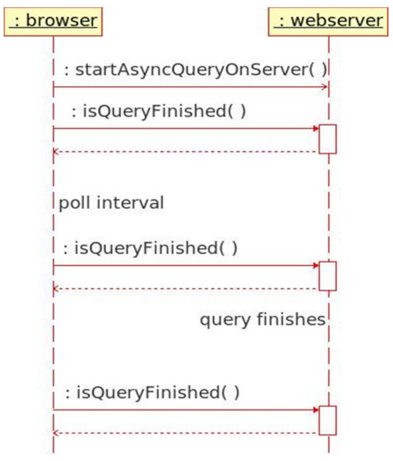


Figure 25: Data Polling [49]

Any updated information on the server side is received by the client, the next time when client makes request. The main advantage of polling is its simple implementation in web applications. Many applications employing data polling require only JavaScript and AJAX for implementation.

3) Long Polling

Long polling someway works in the same fashion as data streaming. In long polling, a client request is sent whenever the application is loaded, however the request remains suspended and response is received only when a data update on server side is available. After data receipt, the connection is closed while a new connection is established resulting in the connection being active most of the time. The request/response workflow in long polling is illustrated by Figure 26.

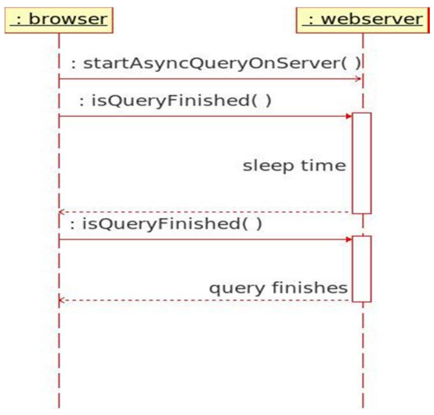


Figure 26: Long polling [49]

The sleep time is the duration in which the server holds the client query until there is a new data update to be sent. Usually, long polling is also named as Comet. The key benefit of long polling is immediate retrieval of updated information to client from server.

2.14. Data Push Technologies

Data push occurs when a server intends to publish the latest data available to all the potential clients which are already in contact with the server. Data push is opposite to data pull, where the request for data transfer is initiated by the subscriber client, whereas publisher or server initiates the data transmission in data push. There are several technologies available for data push from server, like

- APE (AJAX Push Engine)
- COMET
- DWR (Direct Web Remoting)

2.14.1. APE

AJAX Push Engine or APE, originally based on JavaScript, is an open source web tool introduced for AJAX data push APE comprising mainly AJAX technology features, is quite easy to implement tool excluding need of installing any special plugin software. Figure 27 gives an illustration of APE's working methodology [51].

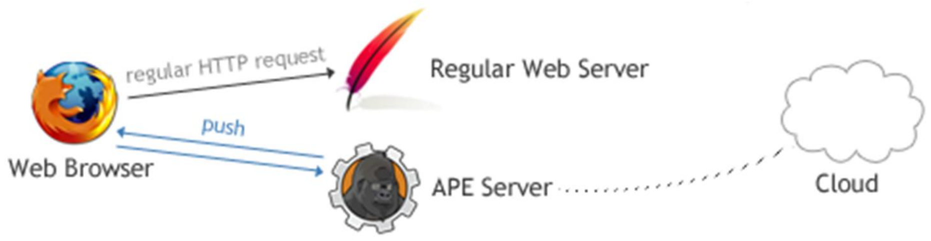


Figure 27: Ajax Push Engine (APE) working [51]

Information from various sources is routed towards an APE server which then waits for client application subscription. APE server starts pushing the real-time live data updates to all the web applications who have subscribed to the server. The APE supports all commonly used operating systems and mobile devices having generally available web browsers.

2.14.2. COMET

COMET is a data push technology, which maintains a long-lasting connection with a browser through an HTTP request, allowing real time data push without browser's need to request data. COMET eliminates the need of polling for real time data access by retaining a continual HTTP connection between the client and server.

In principle, COMET uses two major data push techniques in its implementation: long polling and data streaming. COMET in some applications makes use of one of the key AJAX tool XMLHTTPRequest (XHR) for server to browser communication.

2.14.3. DWR

Direct Web Remoting or DWR as defined by [52]: 'is a Java library that enables Java on the server and JavaScript in a browser to interact and call each other as simply as possible'.

DWR is a rather a novel emerging web tool aimed to facilitate the communication between a server side java application and a client side web application for real time data transmission. The working mechanism of DWR application is shown in Figure 28.

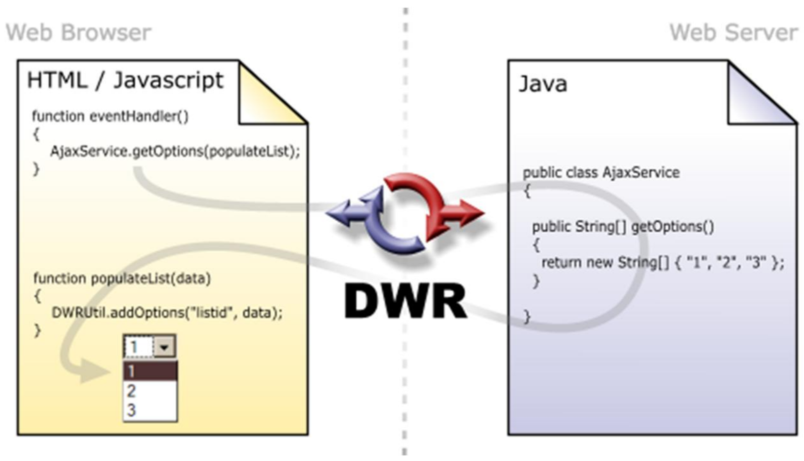

Web Browser
Web Server
HTML / Javascript
function eventHandler()
{
AjaxService.getOptions(populateList);
}
function populateList(data)
{
DWRUtil.addOptions("listid", data);
}
1
1
2
3
DWR
Java
public class AjaxService
{
public String[] getOptions()
{
return new String[] { "1", "2", "3" };
}
}

Figure 28: DWR working [52]

There are two main components that form a DWR web application: a Java Servlet and a JavaScript implementation function. The Java Servlet is the one which manages all the incoming web requests and related responses. The JavaScript function is the one which repeatedly asks for any data update from the java application while updates the webpage based on the information update received.

DWR provides the facility to retrieve and use the informational data on web from java classes by dynamically generating JavaScript's without requiring any supportive software or plug-in to make it functional. DWR also provides the opportunity to efficiently collaborate and integration with generally used server-side Java technologies, e.g. Spring, Guice etc. In addition, other benefits include data handling support for almost all available data format between Java and JavaScript, call batching and Cross-Site Request Forgery (CSRF) protection [52].

3. TEST BED AND IMPLEMENTATION DETAILS

This chapter gives details on the test bed used for implementation and results evaluation as well as a detailed description of implementation scheme, steps and application deployment.

3.1. Testbed

The test bed considered in this thesis is a multi-robot assembly line located at Factory Automation Systems and Technology (FAST) laboratory, Department of Production Engineering Tampere University of Technology (TUT). The testbed is sometimes denoted in this thesis by the name FASTory.

The line is a pallet-flow based assembly line consisting of 10 robotic workstations, eight assembly stations, one sorting station and one conveyor connecting two rows of workstations. Each of the 10 robotic workstations consists of a robot, a pen feeder, one normal and one bypass conveyor. All the robotic work cells are connected through the conveyors resulting in a continuous pallet flow across the assembly line. Below is the3D model of the assembly line. Figure 29, 30 and 31 present3D isometric, top and front views of FASTory assembly line respectively.

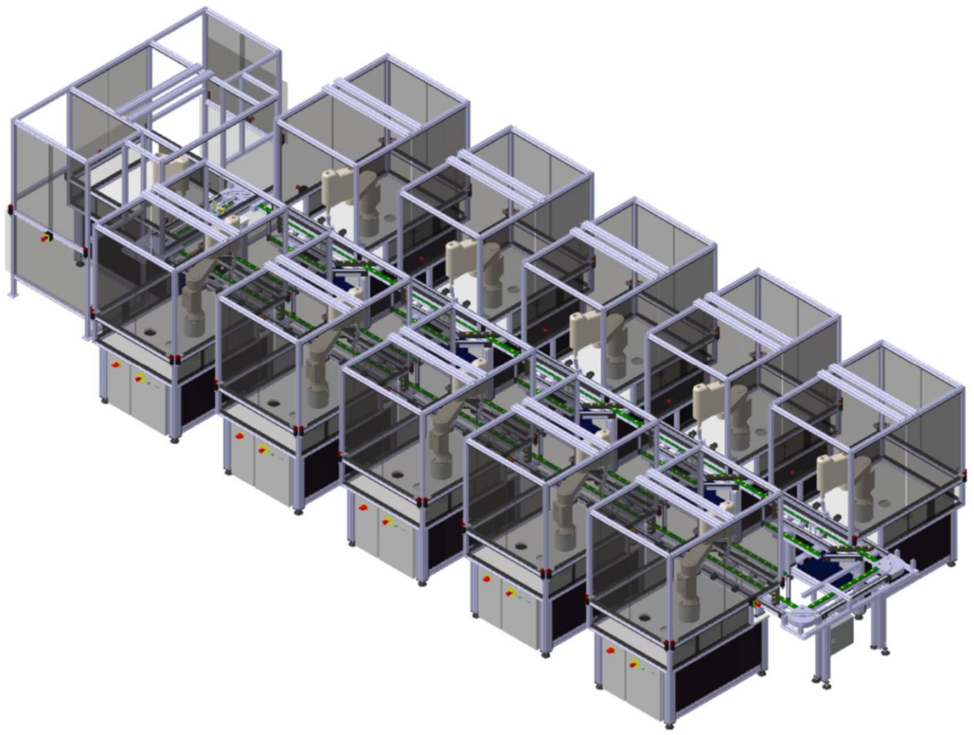


Figure 29: 3D isometric view of FASTory assembly line

As can be seen from the above model, each line side comprised of 5 interconnected work cells connected to the 5 work cells of the other side via a conveyor. Finally, the loading station completes the closed loop the whole assembly line.

Pallets travel all the way through the line and taken up by the specific cell which is programmed to operate that particular pallet. The loading station ultimately picks up the finished pallet and unloads it to the pallet storage, while at the same time places a new pallet on the line.

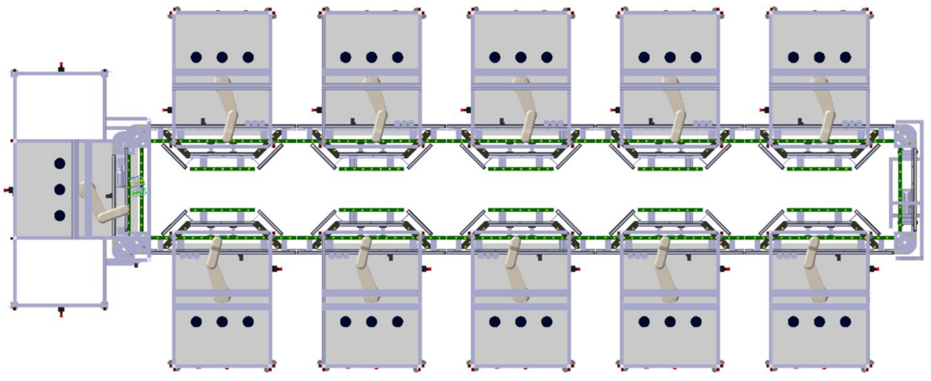


Figure 30: Top view of FASTory assembly line

Each line work cell contains two conveyors: normal and a bypass, a SCARA robot with a pen feeder attached. The robot performs the specified operation by drawing certain patterns on pallet surface. The pallets are diverted from the main line through bypass conveyor if the robot is already in operation or the pallet does not require any operation to be carried out.

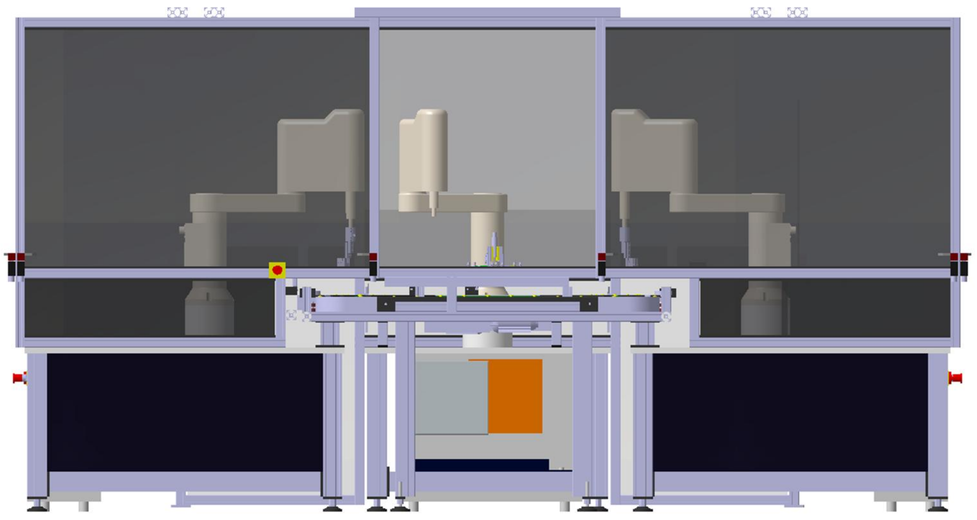


Figure 31: Front view of FASTory assembly line

S1000 is a java based smart RTU, providing various industrial control solutions having web services support. These controllers can be configured to perform certain operations and publish events using web services. The control logic is designed using Structured Text (ST) programming language. The device supports web services discovery and provides references to WSDL files for hosted service containing events information, input/output message formats and the operations carried out.

Within current testbed setup, S1000 controllers are employed in each cell of the FASTory assembly line. The controllers are capable of taking inputs from user as SOAP messages and trigging output functions such as movement and controlling various conveyors or activating robots for drawing desired product sketches etc. In addition, the controllers also enable users to access and configure the controllers online through Ethernet. The output status of an action is published as SOAP message for all the subscribers over the network.

3.2. Research Objectives and Available information

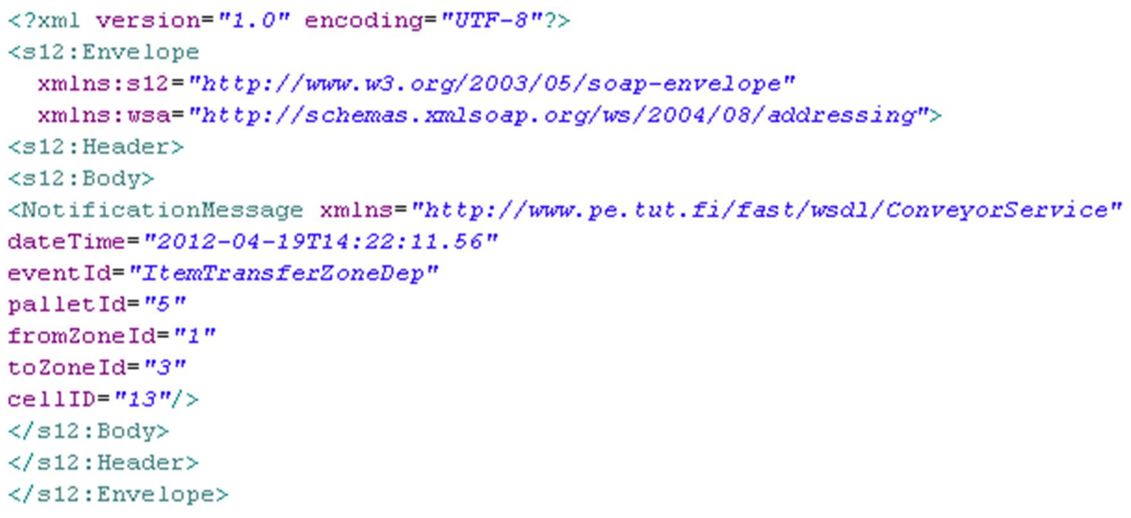
The goals and objectives to be achieved are presented as follows:

- Receive and parse XML informational messages from line using Java Eclipse
- Expose the informational messages as JSON messages on browser
- Access the parsed information from Java application into browser
- Link the data with 3D animation and simulate on browser

3.2.1. Message Types

This work mainly deals two types of XML messages: Pallet Activity (Notification Message, see Figure 33) and Robot Activity (Equipment change state Message, see Figure 34).

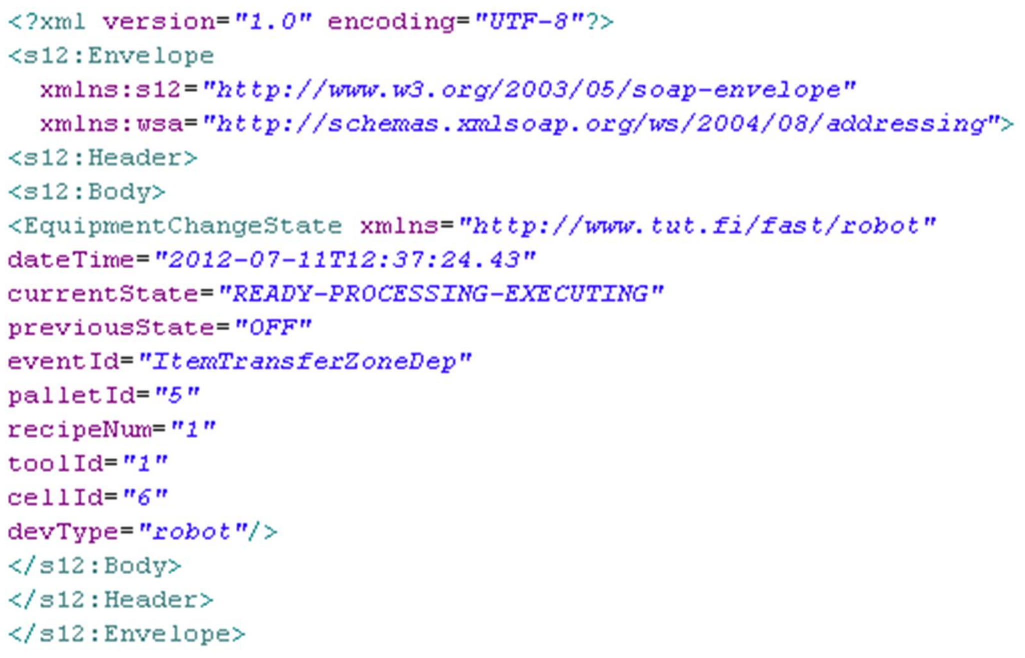
The pallet activity message (Figure 33) contains information about the pallet and its path through zones specification. Each message has a unique timestamp value, pallet and cell id, event id and zone information. The zone information specifies either the pallet would follow main conveyor path or bypass conveyor path. In case of main conveyor path, the pallet follows its path through zone 1 to 2 and 2 to 3. For bypass conveyor path, the pallet follows its path through zone 1 to 4. The notification message sample format is shown under by figure 32.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<s12:Envelope
  xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
<s12:Header>
<s12:Body>
<NotificationMessage xmlns="http://www.pe.tut.fi/fast/wsdl/ConveyorService"
dateTime="2012-04-19T14:22:11.56"
eventId="ItemTransferZoneDep"
palletId="5"
fromZoneId="1"
toZoneId="3"
cellID="13"/>
</s12:Body>
</s12:Header>
</s12:Envelope>
```

Figure 32: Pallet Activity Notification Message Sample

The robot activity message as depicted by Figure 33, mainly describes information about its current and previous state. These states are specified according to CAMX IPC-2541 Equipment change state chart. As like the pallet activity message, robot activity message also keeps record of the similar information like timestamp, pallet and cell id, event id and tool id, device type and recipe number.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<s12:Envelope
  xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
<s12:Header>
<s12:Body>
<EquipmentChangeState xmlns="http://www.tut.fi/fast/robot"
dateTime="2012-07-11T12:37:24.43"
currentState="READY-PROCESSING-EXECUTING"
previousState="OFF"
eventId="ItemTransferZoneDep"
palletId="5"
recipeNum="1"
toolId="1"
cellId="6"
devType="robot"/>
</s12:Body>
</s12:Header>
</s12:Envelope>
```

Figure 33: Robot Activity Message Sample

3.3. Implementation Scheme

3.3.1. Project Implementation Overview

Figure 34gives a brief illustration of overall project implementation.

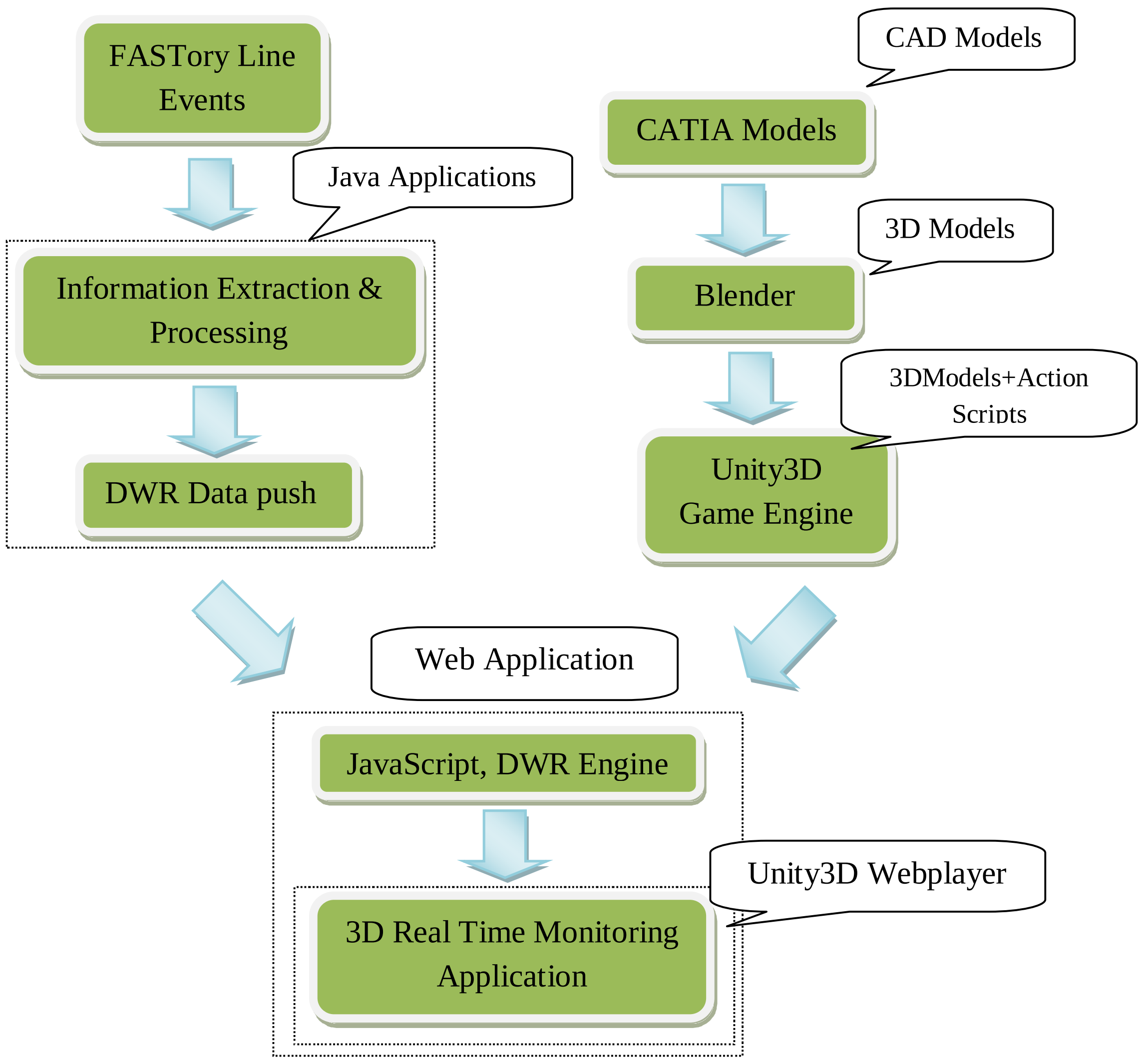


Figure34: Project Implementation layout

The project implementation is based upon three stages: data acquisition and processing, model design and addition of animation adding, plus merger of the aforementioned segments on web.

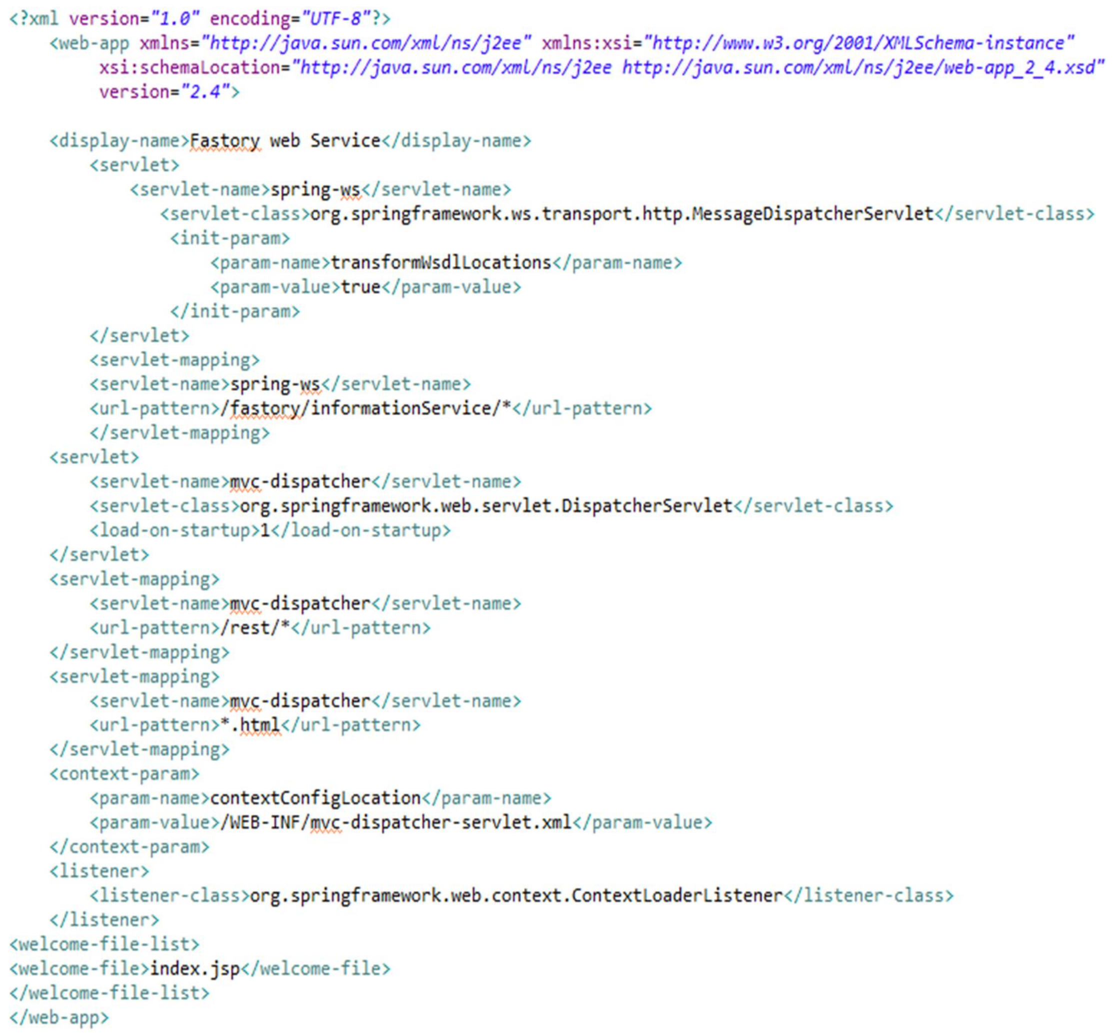
The data acquisition part performed in Java, mainly addresses information gathering from the events received from the test bed FASTory assembly line. Spring framework (WS and MVC) is used to implement the required tasks. The information messages received are parsed and the extracted information is processed for further use. The obtained data is further treated to publish as JSON information message through a RESTful web service or made available to be called from the web application.

The model designing and animation section elaborates the details concerning 3D model designing and adding animation action. The models are first drawn considering the specifications and dimensions of real world test bed apparatus using CATIA software and are later used with Unity3D game engine. However, the constructed models are required to be made compatible to work with Unity3D functional environment. For this purpose, an intermediate application called ‘Blender’ is introduced to accomplish the job. Finally, the refined models are imported in Unity3D and provided with action scripts for desired animation and visualization.

In the end, both the Java and Unity3D applications outputs are merged together in Java to visualize the results on web browser. The output from the Java application is called into the browser using DWR functions while the animation function is imported from Unity3D asset folder to Java application project folder. The gathered information is then linked to the animation function and sent for simulation on web. The Unity3D WebPlayer is included to the project folder to allow 3D animation simulation in the browser.

3.3.2. Project configuration in Java Eclipse (JEE)

Project configuration is mainly done in web.xml file located inside web content directory. The file resides under WEB-INF folder and holds information related to incoming requests and their respective dispatcher servlets mapping in web.xml document. Figure 35 given below shows the web.xml configuration file followed by explanation of the elements involved:



```xml
<?xml version="1.0" encoding="UTF-8"?>
    <web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
         version="2.4">

    <display-name>Fastory web Service</display-name>
        <servlet>
            <servlet-name>spring-ws</servlet-name>
               <servlet-class>org.springframework.ws.transport.http.MessageDispatcherServlet</servlet-class>
                <init-param>
                    <param-name>transformWsdlLocations</param-name>
                    <param-value>true</param-value>
                </init-param>
        </servlet>
        <servlet-mapping>
        <servlet-name>spring-ws</servlet-name>
        <url-pattern>/fastory/informationService/*</url-pattern>
        </servlet-mapping>
    <servlet>
        <servlet-name>mvc-dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>mvc-dispatcher</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>mvc-dispatcher</servlet-name>
        <url-pattern>*.html</url-pattern>
    </servlet-mapping>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
<welcome-file-list>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

Figure 35: Project configuration file web.xml

- The *MessageDispatcherServlet* is the one which encounters an incoming request and dispatches to certain endpoints. In this case, the dispatcher servlet is named as '*spring-ws*'. As the web.xml shows, *MessageDispatcherServlet* is specified to look for incoming request containing 'fastory/informationservice' as URL pattern.
- The *DispatcherServlet* is responsible for requests handling and referring to Spring MVC controller for dispatching. In this case, the dispatcher servlet is named as '*mvc-dispatcher*'. Two kinds of requests are defined in web.xml's description. Firstly, all the web requests specifying an html request are defined by '*/.html/*' tag. Secondly, the RESTful web services requests are defined by '*/rest/*'. In this way, the URL's containing '*/.html/*' and '*/rest/*' as web address endings are taken up by *DispatcherServlet* and are sent forward to Spring MVC controller for further processing.
- The *load-on-startup* element is used to inform the web container about loading of a particular servlet. Without specifying this parameter, the web container would load the servlet whenever it feels necessary or when a request related to

the servlet arrives. This may lead to an unnecessary delay in response time from the particular servlet for handling the request. The *load-on-startup* parameter tells the web container for servlet loading during deployment time of the application.

- The *ContextLoadListener* is a servletlistener which loads Spring configuration files associated with the servlets. The *ContextLoadListener* loads the particular servlet as defined by any of their dispatcher servlet class. For example: in this project, *MessageDispatcherServlet* and *DispatcherServlet* are named as '*spring-ws*' and '*mvc-dispatcher*' respectively. The *ContextLoadListener* looks and automatically loads relevant servlets during initialization, i.e. spring-ws-servlet and mvc-dispatcher-servlet.
- The *welcome-file-list* element depicts the start-up page of the application as specified by the user. The desired webpage is required to reside within the same WebContent folder to show up during application start-up.

3.3.3. Spring WS Implementation

Spring web service is implemented by employing Spring endpoint feature to handle incoming SOAP requests arriving from the robotic assembly line cells of the used test bed arrangement.

Spring WS implementation in the project is shown as below by figure 36.

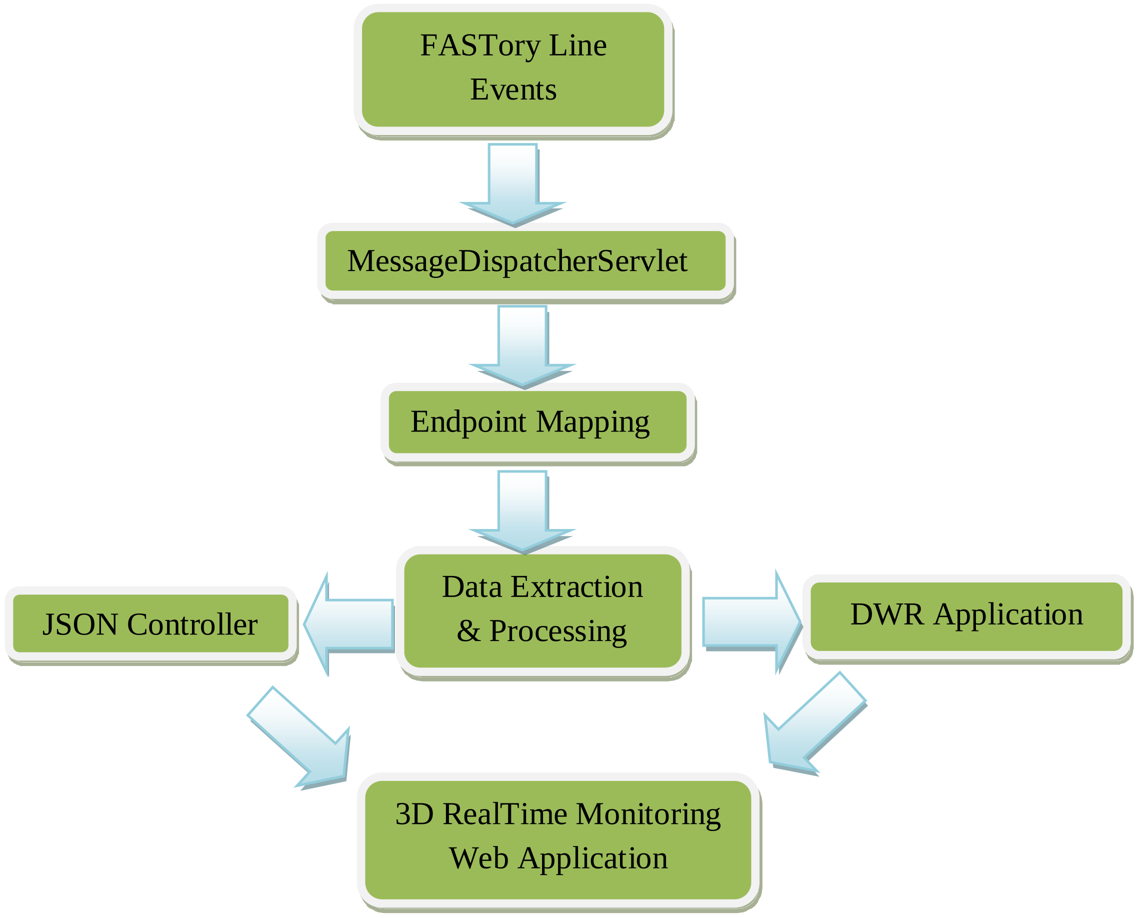
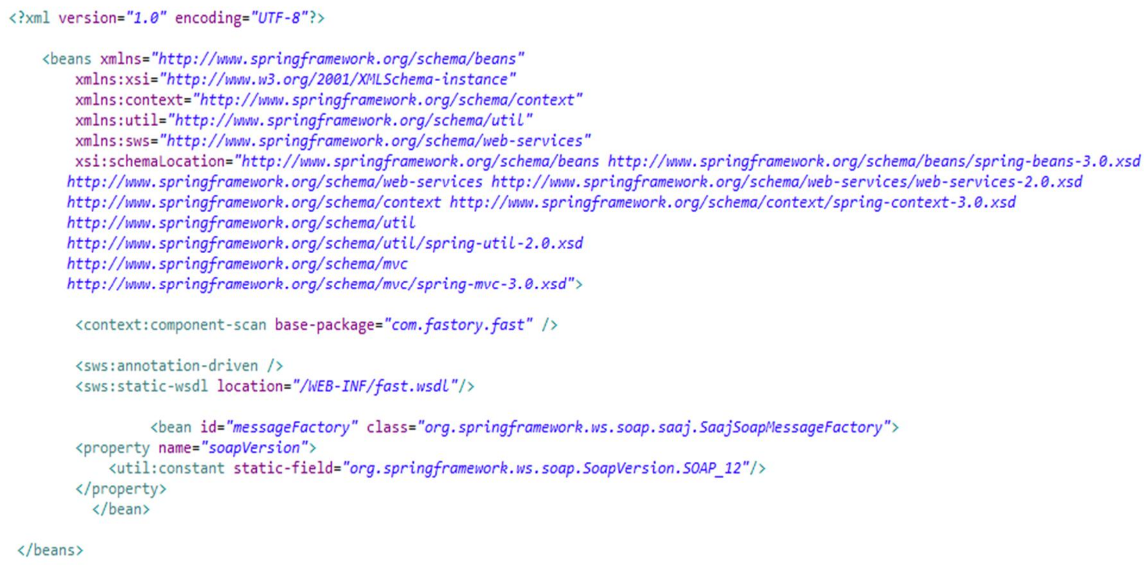


Figure 36: Spring WS implementation

- The events from the test bed are generated and sent through S1000 controller while Spring WS application on server side, receives and processes those SOAP requests.
- The MessageDispatcherServlet handles the messages and maps them to their respective endpoint in Java application.
- The message contents are extracted and are sent to their respective methods for further processing.
- The information is then used by a JSON controller class which utilizes the information and exhibits the result as a RESTful web service containing the information as a JSON Message.
- The received information is also used by a DWR class which transforms the data into an array in order to make it available for web application having DWR data push engine.
- The information is then forwarded to

3.3.4. Spring WS configuration

Following figure 37 shows spring WS configuration done within Spring WS servlet during the project configuration.



```
<?xml version="1.0" encoding="UTF-8"?>

    <beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:util="http://www.springframework.org/schema/util"
        xmlns:sws="http://www.springframework.org/schema/web-services"
        xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
       http://www.springframework.org/schema/web-services http://www.springframework.org/schema/web-services/web-services-2.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd
       http://www.springframework.org/schema/util
       http://www.springframework.org/schema/util/spring-util-2.0.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

        <context:component-scan base-package="com.fastory.fast" />

        <sws:annotation-driven />
        <sws:static-wsdl location="/WEB-INF/fast.wsdl"/>

                <bean id="messageFactory" class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory">
        <property name="soapVersion">
            <util:constant static-field="org.springframework.ws.soap.SoapVersion.SOAP_12"/>
        </property>
          </bean>

 </beans>
```

Figure 37: Spring WS configuration

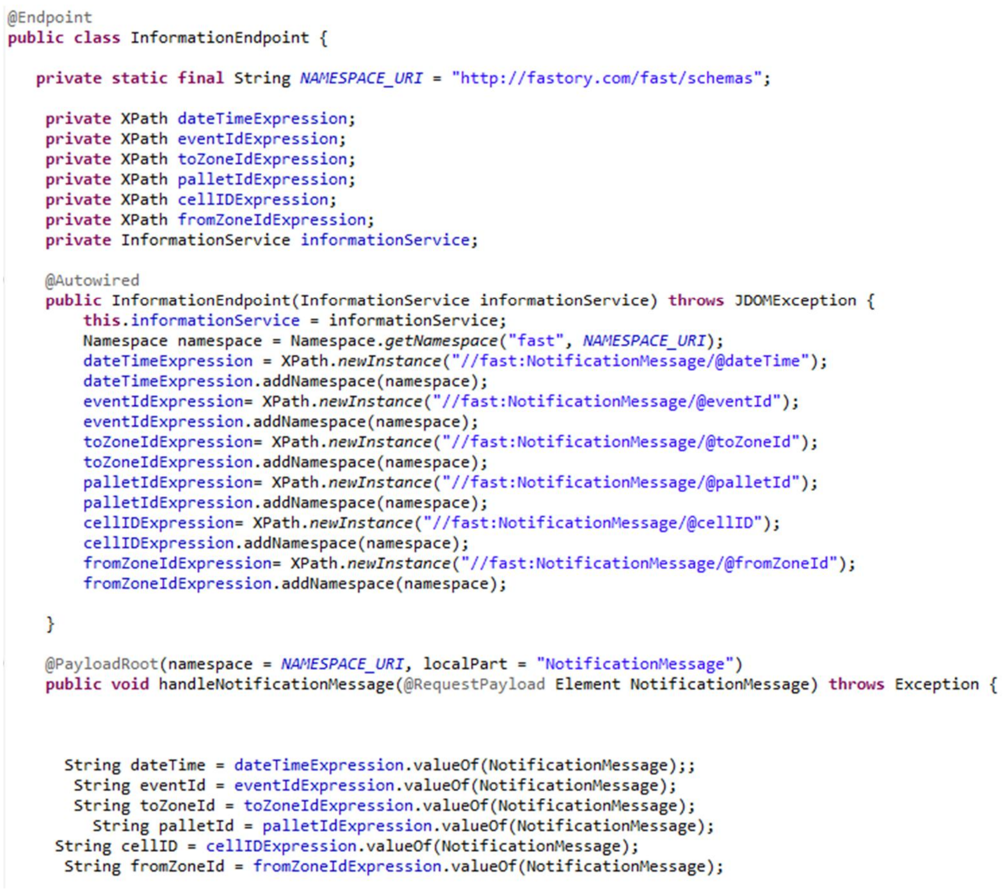
- The Spring WS configuration needs to include number of springframework files to make the web service application working, shown as bean element in the figure above.
- The *‘context:component-scan’* element scans the defined package and automatically registers the beans within the application. In general, the beans are separately declared so that Spring bean container can detect and register those components. The ‘*context:component-scan’* element simplifies the code by exclud-

ing the definition of individual endpoints and allows to manage the dependency directly in the java class. Therefore, the annotations used in the project like *@Endpoint*, @Autowired and @ Payload are automatically scanned and registered in Spring bean container within the application.

- SOAP request messages are defined by their service contracts. Since Spring WS supports contract-first web service, therefore requires the in advance specification of WSDL document describing the messages. The location of the static-WSDL file is described as *'WEB-INF/fast.wsdl'*. The description of the WSDL document together with their XSD schema files having robot and equipment change messages details are shown in Appendices 1 and 2.
- The web service messages are implemented using a WebServiceMessageFactory. In this project, since the project is based on java therefore WebServiceMessageFactory used is Saaj. Based on DOM (Document Object Model), SaajMessageFactory uses SOAP with Attachments API for Java for creating message implementations.

3.3.5. Endpoint Implementation

Following is the description of an information endpoint for receiving and dispatching robot activity notification messages. Figure 38 shows spring WS endpoint implementation.



```
@Endpoint
public class InformationEndpoint {

   private static final String NAMESPACE_URI = "http://fastory.com/fast/schemas";

    private XPath dateTimeExpression;
    private XPath eventIdExpression;
    private XPath toZoneIdExpression;
    private XPath palletIdExpression;
    private XPath cellIDExpression;
    private XPath fromZoneIdExpression;
    private InformationService informationService;

    @Autowired
    public InformationEndpoint(InformationService informationService) throws JDOMException {
        this.informationService = informationService;
        Namespace namespace = Namespace.getNamespace("fast", NAMESPACE_URI);
        dateTimeExpression = XPath.newInstance("//fast:NotificationMessage/@dateTime");
        dateTimeExpression.addNamespace(namespace);
        eventIdExpression= XPath.newInstance("//fast:NotificationMessage/@eventId");
        eventIdExpression.addNamespace(namespace);
        toZoneIdExpression= XPath.newInstance("//fast:NotificationMessage/@toZoneId");
        toZoneIdExpression.addNamespace(namespace);
        palletIdExpression= XPath.newInstance("//fast:NotificationMessage/@palletId");
        palletIdExpression.addNamespace(namespace);
        cellIDExpression= XPath.newInstance("//fast:NotificationMessage/@cellID");
        cellIDExpression.addNamespace(namespace);
        fromZoneIdExpression= XPath.newInstance("//fast:NotificationMessage/@fromZoneId");
        fromZoneIdExpression.addNamespace(namespace);

    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "NotificationMessage")
    public void handleNotificationMessage(@RequestPayload Element NotificationMessage) throws Exception {


      String dateTime = dateTimeExpression.valueOf(NotificationMessage);;
       String eventId = eventIdExpression.valueOf(NotificationMessage);
       String toZoneId = toZoneIdExpression.valueOf(NotificationMessage);
         String palletId = palletIdExpression.valueOf(NotificationMessage);
     String cellID = cellIDExpression.valueOf(NotificationMessage);
      String fromZoneId = fromZoneIdExpression.valueOf(NotificationMessage);
```

Figure 38: Endpoint Implementation

- A Spring WS endpoint is a java class annotated by the *@ Endpoint*as shown by the figure above. The MessageDispatcherServlet receives the incoming message requests and maps them according to their respective endpoints. The messages are identified by a specific namespace defined in the message. An error will be generated if the message containing namespace is different from the one defined in namespace of the endpoint configuration.
- XPath is used to extract data from the received XML SOAP messages. XPath is a query, language, used for selecting and extracting node information in an XML document.
- The *@Autowired*annotation is included to autowire bean on the setter method by matching data type. Here *@Autowired* annotation used with setter method attempts to execute byType autowiring on the method.
- The *@PayloadRoot* annotation maps requests to the method by looking at the root element of the payload. The PayloadRoot annotation requires two elements: localPart and namespace. Therefore, the incoming requests messages containing the defined localPart and target namespace are dispatched and processed by the method to extract values of the message parameters.

3.3.6. Web Application implementation

The web application is implemented following a ModelViewController (MVC) pattern running on Apache Tomcat Server. The model is implemented with JSON data and processed information, the controller on top of Spring MVC and the view implementation is done with JavaScript, JSP and Jquery. The following figure 39 briefly underlines the project web implementation.

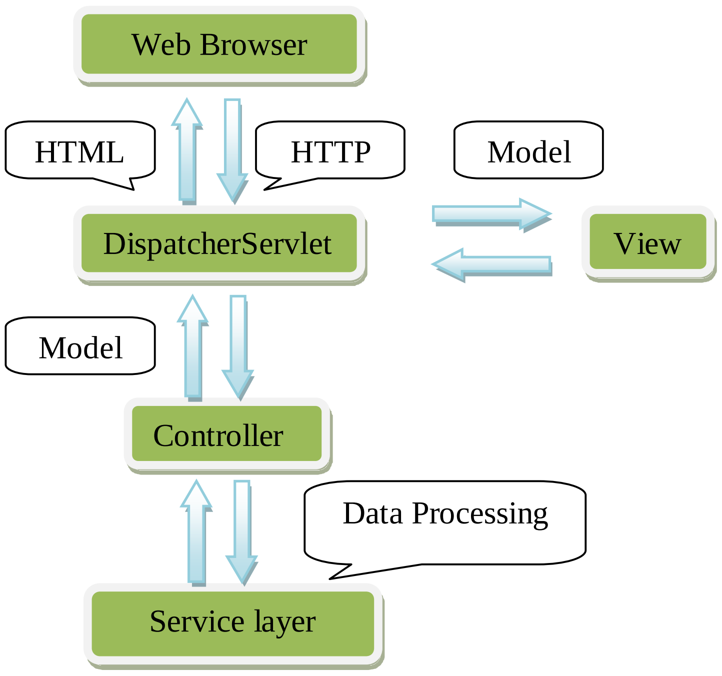
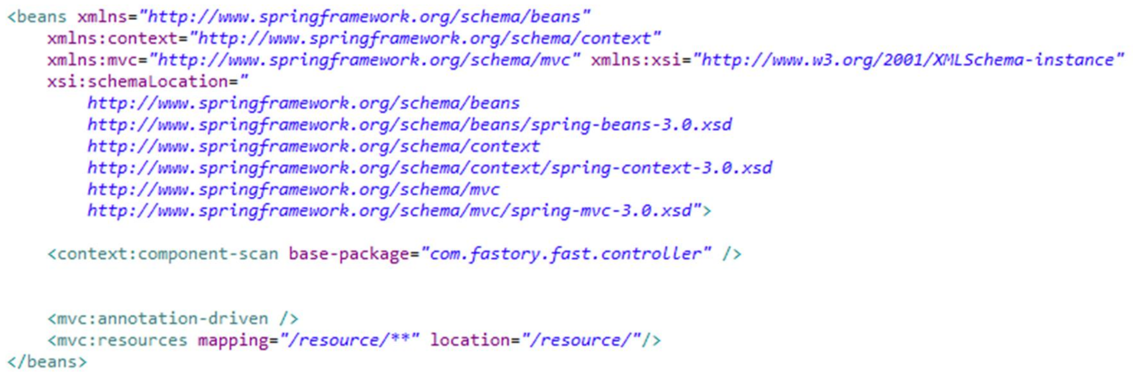


Figure 39: Web application implementation

- The 3D Real time monitoring application and other available information can be accessed on the browser using valid URLs.
- The *DispatcherServlet* receives the HTTP web requests and forwards them to their respective controllers, invoking particular methods to respond to received web requests.
- The methods in the service layer fulfil the request by retrieving the processed data.
- The data is then forwarded to the controller and later on to *DispatcherServlet* with model information to view layer.
- The view’s response through *DispatcherServlet* is then sent back to the browser as HTML for data visualization

3.3.7. Spring MVC Configuration

Spring MVC configuration is done in mvc dispatcher servlet is shown by figure 40.



```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.0.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

    <context:component-scan base-package="com.fastory.fast.controller" />


    <mvc:annotation-driven />
    <mvc:resources mapping="/resource/**" location="/resource/"/>
</beans>
```

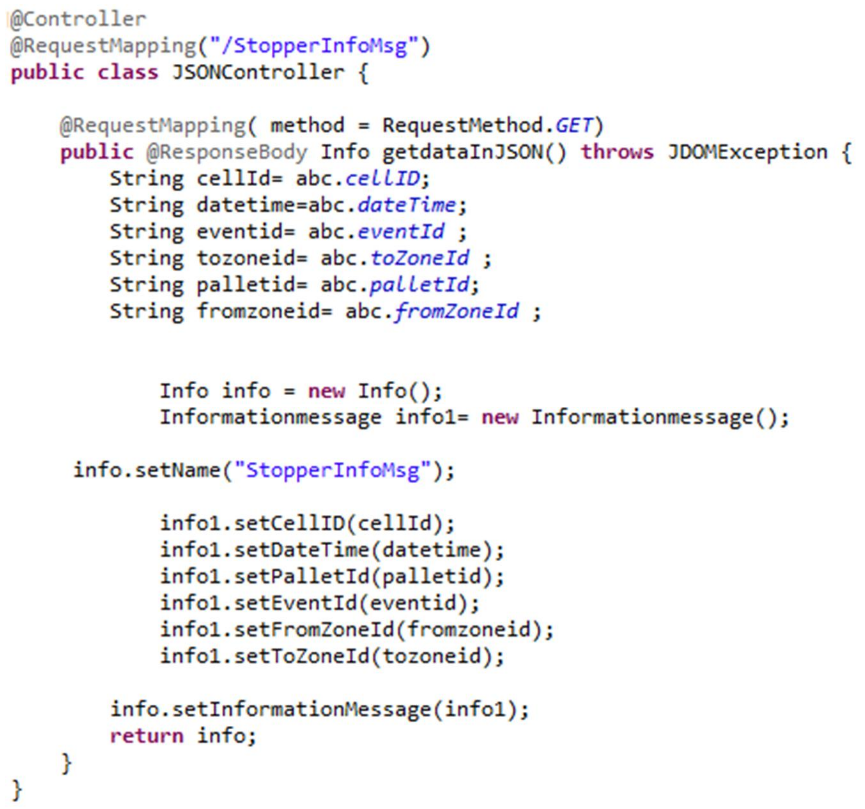
Figure 40: Spring MVC Configuration

- The Spring MVC configuration requires inclusion of number of spring framework configuration files for application implementation, shown as beans element in the figure above.
- As stated previously in Spring WS Configuration, the *‘context:component-scan’* element scans the defined package and automatically registers the beans within the application. The ‘*context:component-scan*’ element simplifies the code by excluding the definition of individual endpoints and allows to manage the dependency directly in the java class.
- The ‘*mvc:annotation-driven*’ tag adds a number of features to the spring mvc application. These include: support for Spring Type ConversionService, allows reading/writing options for XML and JSON, support for date, time data formatting.
- The *‘mvc:resources’* tag allows *ResourceHttpRequestHandler* to dispatch all the web requests following the specified mapping and resource location revealed by URL. Therefore, all the web requests followed by *‘/resource/’* part as URL are

taken up and the corresponding web application in ‘*WebContent/resource/*’ is displayed in the browser.

3.3.8. JSON controller

In terms of controller in a Spring MVC, following is a brief description of the JSON controller implemented to publish the received information as a JSON message. An illustration depicting endpoint mappings of JSON controller class is shown here by figure 41.



```
@Controller
@RequestMapping("/StopperInfoMsg")
public class JSONController {

    @RequestMapping( method = RequestMethod.GET)
    public @ResponseBody Info getdataInJSON() throws JDOMException {
        String cellId= abc.cellID;
        String datetime=abc.dateTime;
        String eventid= abc.eventId ;
        String tozoneid= abc.toZoneId ;
        String palletid= abc.palletId;
        String fromzoneid= abc.fromZoneId ;


            Info info = new Info();
            Informationmessage info1= new Informationmessage();

     info.setName("StopperInfoMsg");

            info1.setCellID(cellId);
            info1.setDateTime(datetime);
            info1.setPalletId(palletid);
            info1.setEventId(eventid);
            info1.setFromZoneId(fromzoneid);
            info1.setToZoneId(tozoneid);

        info.setInformationMessage(info1);
        return info;
    }
}
```

Figure 41: JSON Controller class

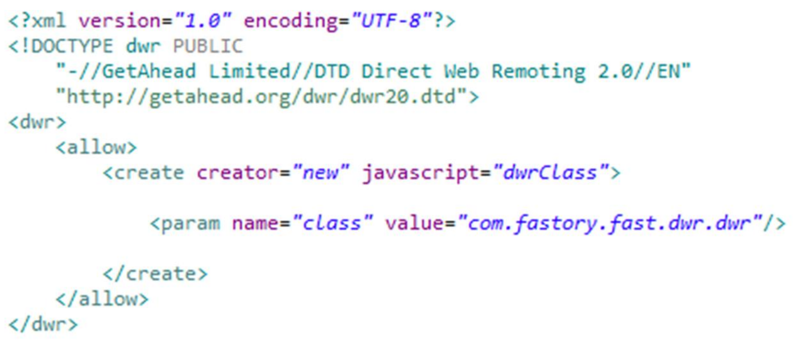
- The *@Controller* annotation acts as reference to a particular class, signifying its role as controller. The dispatcher then looks for subsequent mapped methods used in the annotated class.
- The *@RequestMapping* annotation specifies a URL mapping onto a particular handler method. Here, the URL format is specified is ‘*/StopperInfoMsg’*. Therefore, the web request containing ‘*/rest/StopperInfoMsg’* is taken up and handled by the DispactherServlet which maps the request according to its respective controller method.
- The purpose of adding *@ResponseBody* annotation is to refer writing return type directly to HTTP response body instead of taken up by model or view element.

3.3.9. DWR implementation

Direct Web Remoting (DWR) provides the facility to retrieve and use the informational data on web from java classes by dynamically generating JavaScript's without requiring any supportive software or plug-in to make it functional. The event messages containing status information from the assembly line are processed while extracted data is setup as an array of string variables in another class. In order to implement DWR on web, two kinds of configurations are required: Java and JavaScript. The DWR configuration file dwr.xml and DWR servlet configuration in web.xml is required beforehand in Java. The dwr.xml is a standard practice for DWR configuration and generally placed in *WebContent/WEB-INF/* project folder.

3.3.10. Dwr.xml Configuration

Following figure 42 highlights the XML configuration file of DWR application, located in Java project folder.



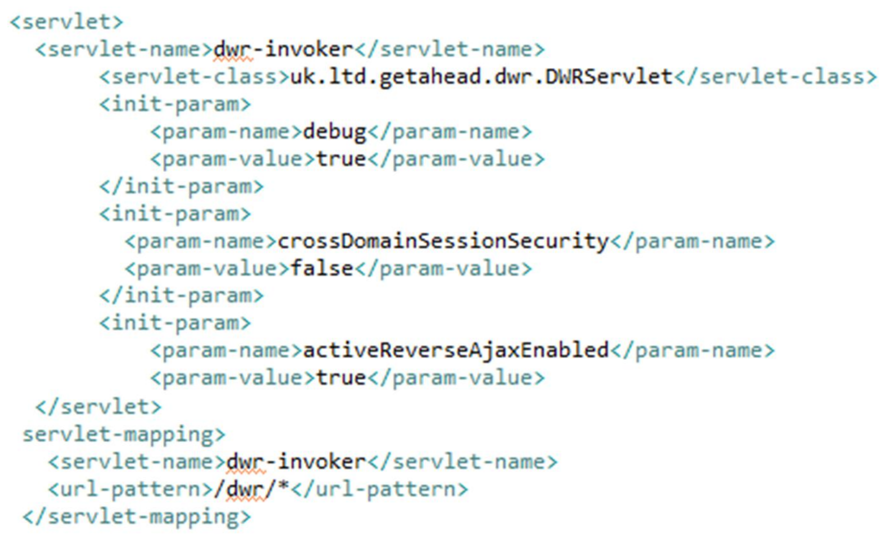
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dwr PUBLIC
    "-//GetAhead Limited//DTD Direct Web Remoting 2.0//EN"
    "http://getahead.org/dwr/dwr20.dtd">
<dwr>
    <allow>
        <create creator="new" javascript="dwrClass">

            <param name="class" value="com.fastory.fast.dwr.dwr"/>

        </create>
    </allow>
</dwr>
```

Figure 42: dwr.xml configuration

The allow tag indicates which JavaScript class is to be created by DWR and through which creator. Here the JavaScript class *'dwrClass'* is created using *new* creator. The *new* creator uses the default constructor to create an instance of the class specified by *param* tag. The java class *com.fastory.fast.dwr.dwr* is converted to JavaScript *dwrClass*, so that a new instance of the class is created each time when a reply *dwrClass.toString* is called in JavaScript function. The response together with the data is then returned to JavaScript reply function.

3.3.11. Web.xml configuration

DWR servlet configuration in web.xml is given by figure 43 as below.



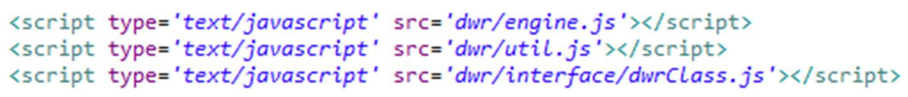
```
<servlet>
  <servlet-name>dwr-invoker</servlet-name>
       <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>
       <init-param>
           <param-name>debug</param-name>
           <param-value>true</param-value>
       </init-param>
       <init-param>
         <param-name>crossDomainSessionSecurity</param-name>
         <param-value>false</param-value>
       </init-param>
       <init-param>
           <param-name>activeReverseAjaxEnabled</param-name>
           <param-value>true</param-value>
  </servlet>
 servlet-mapping>
   <servlet-name>dwr-invoker</servlet-name>
   <url-pattern>/dwr/*</url-pattern>
 </servlet-mapping>
```

Figure 43: DWR Web.xml configuration

- The configuration begins with the servlet name and the associated servlet class as shown by the figure above.
- The URL pattern for the named servlet is then defined in servlet mapping, so that all the web request having the specified URL pattern are handle and dispatch by the DispatcherServlet.
- In addition, several other servlet parameters (shown as *init-param*) are also required for DWR implementation. The debug parameter enables or disables test mode. The crossDomainSessionSecurity parameter is used to allow or forbid requests from other domains. The acitveReverseAjaxEnabled parameter enables or disables polling and Comet mode.

3.3.12. JavaScript configuration

The last step in DWR implementation requires inclusion of few JavaScript library functions as shown by the figure 44. This include DWR engine (engine.js), utility functions (utils.js), the JavaScript class created in dwr.xml (e.g. in this case *dwrclass.js*). The engine.js is the most vital element which offers a number of deployment options and marshal's calls from dynamically generated JavaScript functions [25]. The utils.js contains a variety of utility functions to deal with the received information to facilitate dynamic data update of the web page [25].



```
<script type='text/javascript' src='dwr/engine.js'></script>
<script type='text/javascript' src='dwr/util.js'></script>
<script type='text/javascript' src='dwr/interface/dwrClass.js'></script>
```

Figure 44: DWR JavaScript configurations

4. PROJECT IMPLEMENTATION & RESULTS

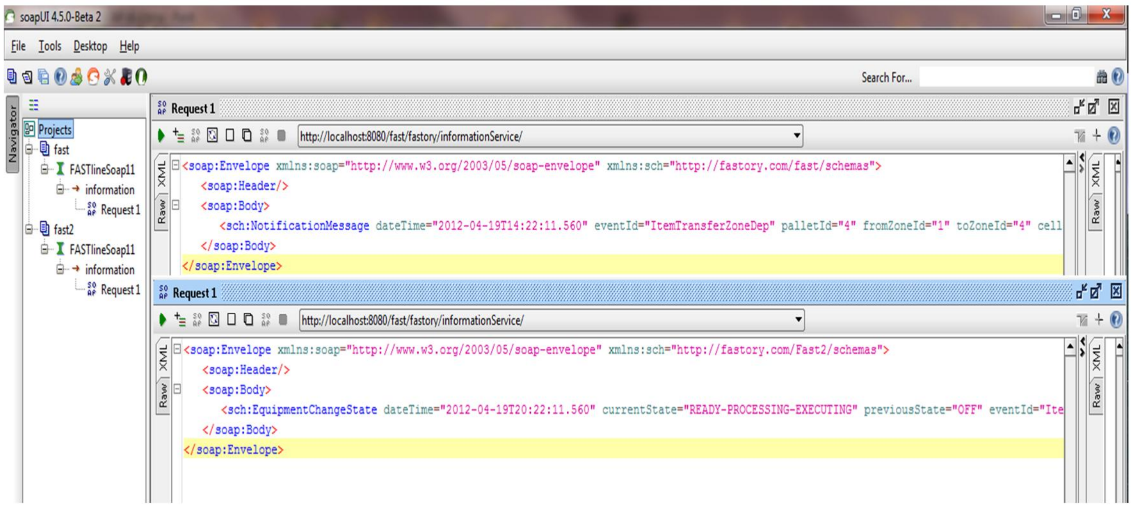
Apart from CAD model design, the 3D animation of the information extracted from test bed events messages is carried out using Unity3D, which essentially is a game development platform. The research work consists of two major tasks: data acquisition and web based visualization.

The data acquisition of the event messages is done in Java Eclipse. Spring web service is used as the java tool for capturing the useful data from the test bed controller SOAP messages. The representation of the results as a web application is done using an mvc design pattern. Spring MVC is used as an overall framework for deploying web application. Apache Tomcat is employed as web server to implement the designed web application.

The web based development is mainly done with the help of JavaScript, HTML and web tools. Direct Web Remoting (DWR) is implemented as web tool for fetching data among java and web application. The animation function is scripted in C++ in Unity3D game development engine. The obtained information from java is then linked to the animation function with the help of JavaScript, jQuery and Ajax. Finally, the web based 3D animated simulation of the events is accomplished through Unity3D Webplayer. The application testing is performed through generating SOAP messages using SOAP UI.
Each of these stages is further detailed as follows.

4.1. Project Testing

The project application testing is done with SOAP UI software. SOAP UI is an open source cross-platform tool mainly introduced for application testing providing number of options and features. SOAP UI has the capability to work with commonly used standard protocols and web services technologies as well. It efficiently employs various testing modes like functional, load, security as well as service simulation.
Figure 45 shows the message generation using SOAP UI utility.



```
http://localhost:8080/fast/fastory/informationService/
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:sch="http://fastory.com/fast/schemas">
   <soap:Header/>
   <soap:Body>
      <sch:NotificationMessage dateTime="2012-04-19T14:22:11.560" eventId="ItemTransferZoneDep" palletId="4" fromZoneId="1" toZoneId="4" cell
   </soap:Body>
</soap:Envelope>
```

```
http://localhost:8080/fast/fastory/informationService/
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:sch="http://fastory.com/Fast2/schemas">
   <soap:Header/>
   <soap:Body>
      <sch:EquipmentChangeState dateTime="2012-04-19T20:22:11.560" currentState="READY-PROCESSING-EXECUTING" previousState="OFF" eventId="Ite
   </soap:Body>
</soap:Envelope>
```

Figure 45: SOAP UI Message Interface

In order to generate soap messages by SOAP UI, the WSDL files containing the message format, data types, and namespace should to be documented beforehand. The WSDL files for both the messages are given in Appendices 1 and 2. The files are located in WebContent directory of the project folder and are made accessible through making a web request with their location as web addresses. The WSDL files are then referenced into SOAP UI which loads the files and creates message formats accordingly. The empty request message parameters are provided with the original data formats used in the real world simulation and the requests are sent to the java application for further processing.

4.2. Web Implementation

Below given is an overview of web implementation of the project as shown in figure 46.

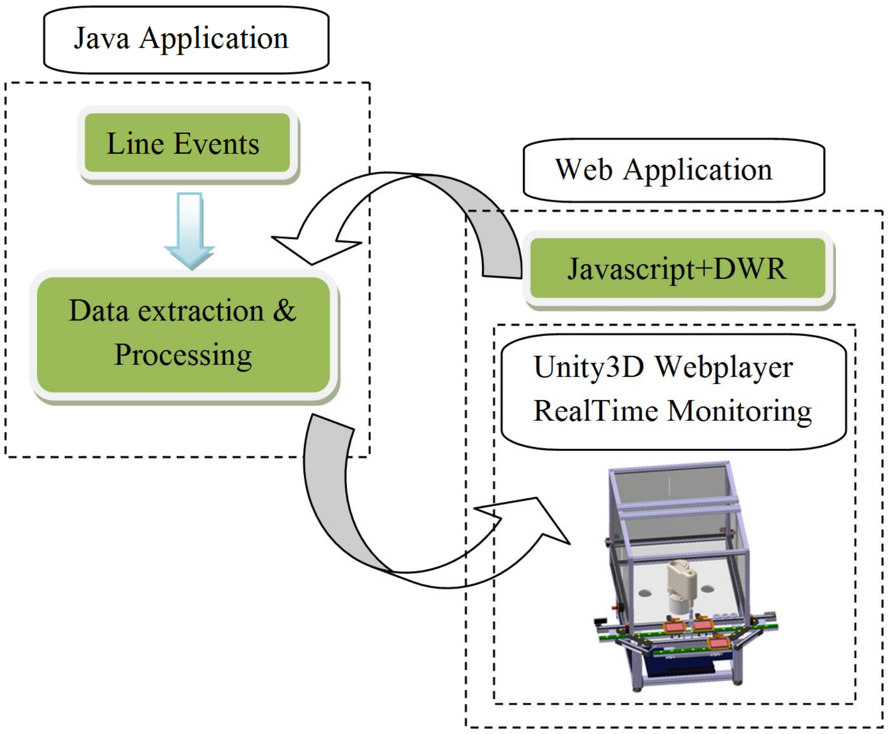

Java Application
Line Events
Data extraction & Processing
Web Application
Javascript+DWR
Unity3D Webplayer RealTime Monitoring

Figure 46: Web ImplementationOverview

The events from the FASTory assembly line or SOAP UI (in testing mode), are received through Spring web service. The information is then processed and made available for both as Restful web service containing JSON data and DWR web application. The DWR caller function scripted in JavaScript calls the Java class for any updated information based on latest arrived events. The Java class responds with the desired requested information and sends it back to the web application function. The received data is parsed and checked if the data is a new set of events based on timestamp value. The parsed data is then sent to the animation function which plays the animation using Unity3D WebPlayer in browser.

4.3. Implementation details

The project CAD models are designed in CATIA by Dassault Systems and are imported in Unity3D game engine to give animation functionality. However, the CAD models are firstly made compatible to work with Unity3D project building requirement. The compatibility is achieved by introducing an intermediate application called Blender. The Blender is yet another open source free to use 3D graphics tool, designed to create 3D models and to add visual effects and animation to them. In this project implementation, the Blender application refines the CAD models and makes them compatible in Unity3D scene environment. Figure 47 shows the imported project model in Unity3D.

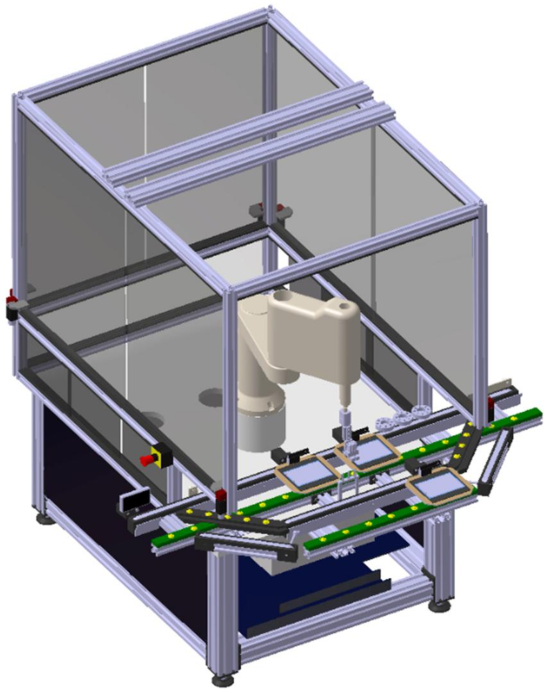


Figure 47: CATIA model in Unity3D

The model animation involves two kinds of animation functions: pallet flow movement and robot activity. The robot activity mainly follows CAMX equipment change state messages, as stated previously in Section 3.6.2. The used case scenario holds two main events for the robot activity, OFF and READY-PROCESSING-EXECUTING. In OFF state, the robot makes no movement while READY-PROCESSING-EXECUTING corresponds to sketch drawing on pallets.

The pallet movement includes pallet flow through various regions/zones across the assembly line as depicted by the figure 48 below.

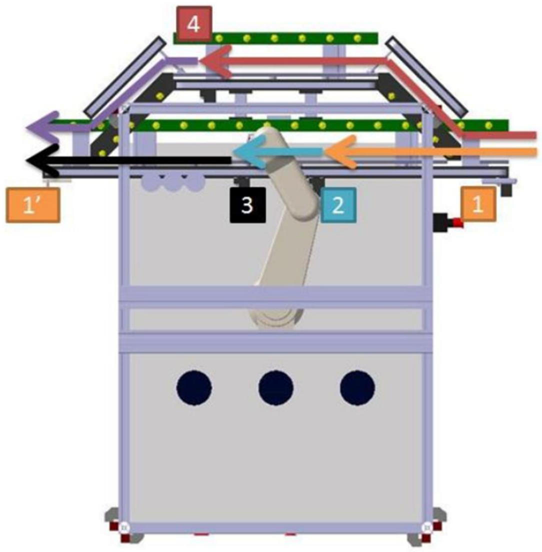


Figure 48: Regions in pallets flow

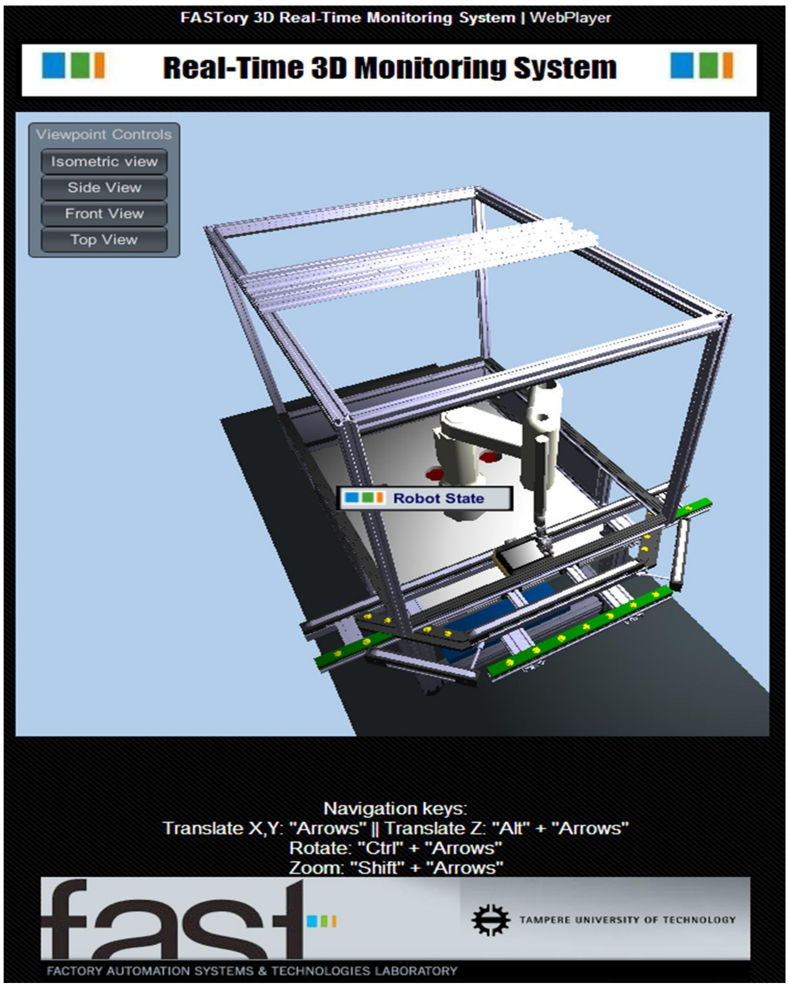
The zones 1, 2 and 3 are located on the main conveyer while zone 4 is on bypass conveyer. Each of these zones is equipped with their respective stoppers. The notification message holds the information of pallet movement from one zone to another by receiving their respective stopper information status. Each notification message contains information of two zones. When the pallet is required to go through a robotic operation, it follows the main conveyer line passing through zones 1, 2, 3. The case where the pallet is required to avoid the main conveyer, the pallet follows the bypass path moving from zone 1 to 4. The zone 1' indicates the entry point of the next and exit point of current cell.

4.4. Results

This section summarizes the outcomes achieved through project implementation.

4.4.1. Real-time Monitoring Application

The web implementation of the real time monitoring application follows execution of the events as depicted by the web application flowchart (Section 4.2).The java class presents the received informational data in the form of an array of strings. The DWR caller function calls the dynamically created java script class and retrieves the information. The data is parsed and formatted as variables which are then sent to the animation functions. The received data is categorized on the basis of events timestamp in order to ensure only newly acquired information, while rejecting any repetitive data. Below given figure 49 shows the 3D real-time monitoring of service oriented FASTory assembly line using Unity3D WebPlayer.


FASTory 3D Real-Time Monitoring System | WebPlayer
Real-Time 3D Monitoring System
Viewpoint Controls
Isometric view
Side View
Front View
Top View
Robot State
fast
TAMPERE UNIVERSITY OF TECHNOLOGY
FACTORY AUTOMATION SYSTEMS & TECHNOLOGIES LABORATORY

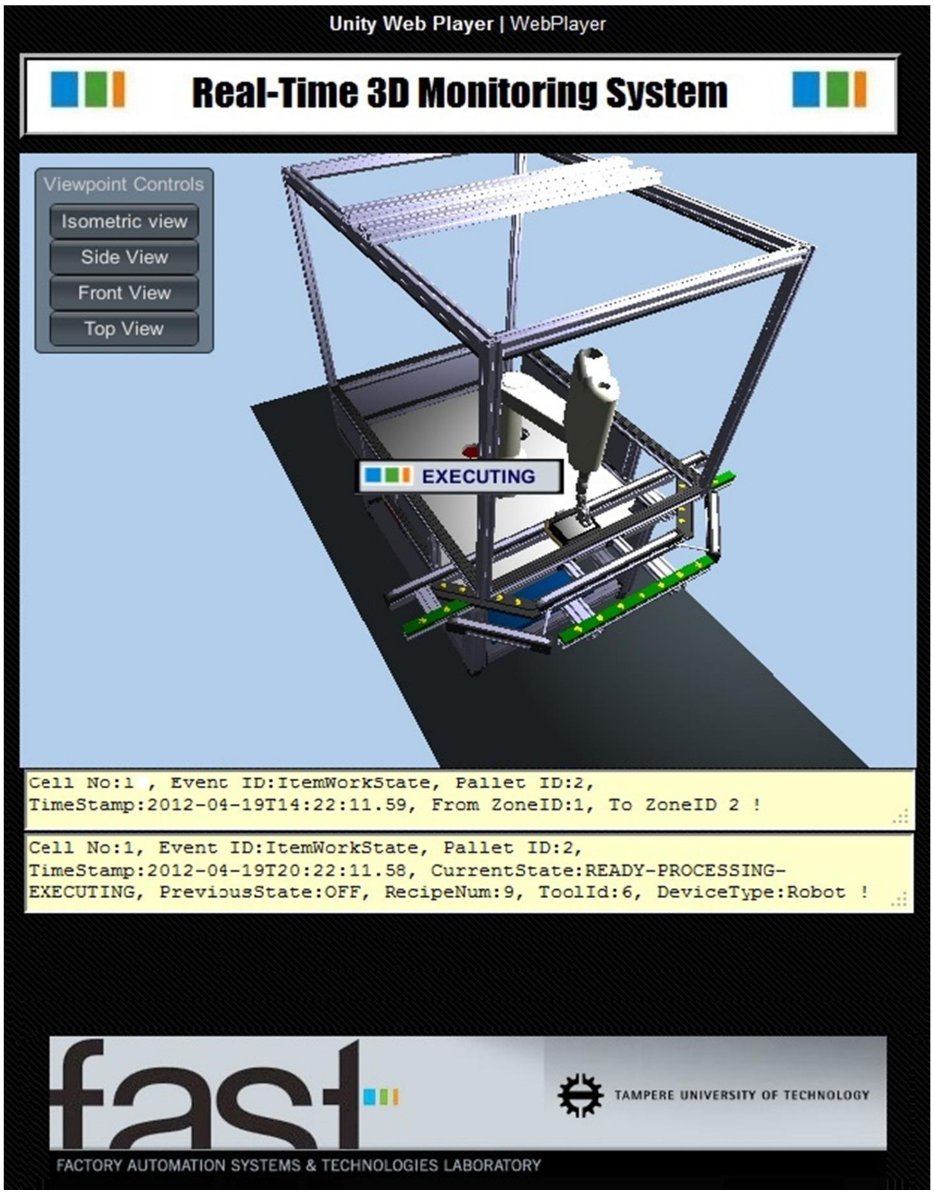
Navigation keys:
Translate X,Y: "Arrows" || Translate Z: "Alt" + "Arrows"
Rotate: "Ctrl" + "Arrows"
Zoom: "Shift" + "Arrows"

Figure 49: 3D Real-time Monitoring Application

The monitoring interface is provided with multiple view options and navigation keys to customize the view from different angles. The robot animation function responds to EquipmentChangeState messages while the pallet movement to stopper NotificationMessages.

4.4.2. Periodic Data Simulation

The designed application can also be used to simulate historic database events. This type of simulation can be useful for the cases where someone wants to track down changes that might occurred at some specific time in past. The following figure 50 shows resulted simulation of the monitoring application for historic database events.


Unity Web Player | WebPlayer
Real-Time 3D Monitoring System
Viewpoint Controls
Isometric view
Side View
Front View
Top View
EXECUTING
fast
TAMPERE UNIVERSITY OF TECHNOLOGY
FACTORY AUTOMATION SYSTEMS & TECHNOLOGIES LABORATORY

```
Cell No:1 , Event ID:ItemWorkState, Pallet ID:2,
TimeStamp:2012-04-19T14:22:11.59, From ZoneID:1, To ZoneID 2 !
```

```
Cell No:1, Event ID:ItemWorkState, Pallet ID:2,
TimeStamp:2012-04-19T20:22:11.58, CurrentState:READY-PROCESSING-
EXECUTING, PreviousState:OFF, RecipeNum:9, ToolId:6, DeviceType:Robot !
```

Figure 50: Execution with info display

The historic data events are stored in an XML file and are accessed through making an Ajax call. Once the data is uploaded, the information is filtered since the stored information occurs with random sequence. The data is then sent to the animation function which simulates them as one data chunk at a time. Similarly, the information is further forwarded to another function which displays the status of the current simulation for tracking as shown by the figure above.

5. CONCLUSION

5.1. Performance Overview and achievements

Combining real-time monitoring with 3D visualization deploying Service Oriented Architecture provides an advanced level of monitoring in an industrial production system. The application enables all users of the organization to remotely access the on-going production activity from any remote place without requiring any complicated simulation platform or extra plug-in through traditional communication means such the Internet. One of the major plus points of the application is its compatibility with all normally used browsers.

The proposed monitoring system provides an alternate approach to traditionally available monitoring schemes where production activity can be visually assessed and analysed at a central location on a graphical user interface like SCADA systems. The monitoring system developed offers a convenient way of monitoring a system for all types of clients, from shop floor operators to higher level executives and corporate personnel.

Real-time monitoring also provides an efficient way to deal with certain uneven incidents with preciseness. By closely monitoring every single event of the production process, the application user can easily figure out the exact cause and location of the problem whenever the production activity is suspended. The simulation of historical data events helps to track the manufacturing activity for any given time.

The monitoring system also offers easy customization capabilities and addition of new and updated features to the application without having an impact on the already working system. Therefore the developed solution is independent of the control mechanism of the considered system and only concerns about the event messages from the process, therefore any modification to the overall control strategy of the production process does not affect the monitoring application as long as the monitoring requirements are true.

5.2. Overall System review

The designing and simulation of 3D CAD models of machines, machine parts or even the whole industrial process has already been in practice for many years. However, the design of monitoring system using 3D models has not been adopted on a large scale. This is because of their complexity, time consumption and availability of all the required resources for developing such applications. Similarly, visualizing and monitoring a production process from any remote location using standard internet means is also not so common in conventional monitoring practices.

This thesis presents a detailed and highly interactive 3D visualization based assembly line monitoring system as compared to traditionally 2D based monitoring techniques. Another noticeable feature of the researched approach is that it addresses a web based 3D monitoring scheme using SOA enabling online access of the monitoring application for users irrespective of their location.

5.3. Future work and recommendations

The developed application presently is able to take on only two types of event messages, stopper notification and robot activity. However, the application's application domain can be extended to undertake and visualize more information with certain modifications to the existing design.

Different methods and techniques for project implementation can be exercised to achieve the desired goals. The XPATH (XML Path Language) is a java tool used for parsing XML data contained in the SOAP event messages. JAXP (Java API for XML Processing) can be used as an alternate to XPATH. The application is capable of publishing the processed information as JSON message, which may be useful for some other application. The Direct Web Remoting (DWR) is adopted as a solution for data fetching from java application to web application, however other data push technologies like Ajax Push Engine (APE), Pushlet or COMET may also be realized as substitute for the desired results.

The test bed setup keeps record of various statistical informations in a central database. The visualization of those vital statistical variables such as number and type of frames drawn on pallets by robot operation as line graphs, overall efficiency of the individual machines for certain period of time viewed as pie charts (using equipment change state historical data), energy consumption and consequently the cost of operation may also be represented using Google charts or analogous web tools.

In addition, the designed approach was tested and implemented for a single test bed cell, which in future can be broadened to address the monitoring of all available cells of the same arrangement.

REFERENCES

[1] Pantforder, D.; Vogel-Heuser, B. “Benefit and evaluation of interactive 3D process data visualization in operator training of plant manufacturing industry”, IEEE International Conference on Systems, Man and Cybernetics (SMC) 2009. Page(s): 824-829.

[2] Patrick Michel (*vice-president, DELMIA Solutions & Marketing, Dassault Systèmes*) “The power of simulation: Virtual 3D simulation tools bring increased productivity and ROI”, http://www.automationmag.com/features/the-power-of-simulation-virtual-3d-simulation-tools-bring-increased-productivity-and-roi.html

[3] Parametric Technology Corporation (PTC), 2006 “Verification and Validation”, http://www.ptc.com/WCMS/files/43562/en/VV2089_v3.pdf

[4] DELMIA Overview. http://www.cdcza.co.za/software/delmia-overview

[5] C. C. Ko;Ben M. Chen; C. D. Cheng “Web-Based 3D Real Time Experimentation”, National University of Singapore, Singapore.

[6] Gurpreet Singh Modi; “Seminar report on Service Oriented Architecture & Web 2.0” Department of Computer Science and Engineering Guru Tegh Bahadur Institute of Technology, New Delhi India. www.gsmodi.com/files/SOA_Web2 Report.pdf

[7] Service-Orientation Design Principles http://serviceorientation.com/index.php/

[8] Benefits of SOA - Web Services http://www.devshed.com/c/a/Web-Services/Introduction-to-Service-Oriented-Architecture-SOA/2/

[9] Lihui Wang;Mohammad Givehchi; Göran Adamson;Magnus Holm “A sensor-driven 3D model-based approach to remote real-time monitoring”, Virtual Systems Research Centre, University of Skövde, Sweden.

[10] Xiongming Zhou;Wenqun Su;Jun Xu; Xiyang Xu, “3D real-time display system of cable temperature” , International Conference on Electricity Distribution (CICED), 2010 China. Page(s): 1 - 4

[11] Bo Tang; Ait-Boudaoud, D.; Lik-Kwan Shark; Matuszewski, B.J. “Pseudo Real-Time System for Visualisation of 3D Scenes”, Geometric Modeling and Imaging (GMAI), 2007. Page(s): 8 – 13

[12] Feldhorst, S.; Fiedler, M.; Heinemann, M.; ten Hompel, M.; Krumm, H. “Event-based 3D-monitoring of material flow systems in real-time”, 8th IEEE International Conference on Industrial Informatics (INDIN), 2010. Page(s): 195 - 200

[13] Kularatna, N.; McDowall, J.; Melville, B.; Kularatna-Abeywardana, D.; Hu, A.P.; Dwivedi, A. “Low-Cost Autonomous 3-D Monitoring Systems for Hydraulic Engineering Environments and Applications With Limited Accuracy Requirements”IEEE Sensors Journal Volume: 10, Issue: 2, Feb 2010 Page(s): 331 – 339.

[14] Hojaji, F.; Shirazi, M.R.A, “Developing a more comprehensive and expressive SOA governance framework”, The2nd IEEE International Conference on Information Management and Engineering (ICIME), 2010.Page(s): 563 - 567

[15] Susanti, F.; Sembiring, J.” The mapping of interconnected SOA governance and ITIL v3.0”, International Conference on Electrical Engineering and Informatics (ICEEI),2011. Page(s): 1 - 5

[16] Yu Chen Zhou; XinPeng Liu; Xi Ning Wang; Liang Xue; Chen Tian; Xiao Xing Liang, “Context Model Based SOA Policy Framework” IEEE International Conference on Web Services (ICWS), 2010. Page(s): 608 - 615

[17] Chapter 1: Introduction to SOA with Web
http://www.aw-bc.com/ samplechapter/032118 0860.pdf

[18] Web Services Architecture
http://www.w3.org/TR/ws-arch/

[19] IBM Services Architecture Team, IBM, Software Group “Web Services architecture overview”
http://www.ibm.com/developerworks/webservices/library/w-ovr/

[20] Extensible Markup Language (XML)
http://www.w3.org/XML/

[21] Thomas Soddemann, RZG “Web Services and Service Oriented Architectures”, Delaman Workshop, 2004
http://www.mpi.nl/delaman/workshop/ppt/soddemann.pdf)

[22] Weichao Li; Yong Liu "Application of XML in E-commerce", WRI World Congress on Software Engineering (WCSE), 2009. Page(s): 262 - 264

[23] Checiu, Laurentiu; Ionescu, Dan, "A new algorithm for mapping XML Schemato XML Schema" International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI), 2010. Page(s): 625 - 630

[24] BalasubramanianSeshasayee; Schwan, K.; Widener, P. "SOAP-binQ: high-performance SOAP with continuousquality management", 24th International Conference on Distributed Computing Systems, 2004 Proceedings. Page(s): 158 - 165

[25] Web Services Description Language (WSDL) 1.1
http://www.w3.org/TR/wsdl

[26] Kwong Yuen Lai; ThiKhoiAnhPhan; Tari, Z. "Efficient SOAP binding for mobile Web services", The IEEE Conference on Local Computer Networks, 2005 30th Anniversary. Page(s): 218 - 225

[27] Nitzsche, J.; van Lessen, T.; Leymann, F. "WSDL2.0 Message Exchange Patterns: Limitationsand Opportunities", Third International Conference on Internet and Web Applications and Services, 2008 (ICIW '08). Page(s): 168 – 173.

[28] Delamer, I.M.; Lastra, J.L.M. "Service-Oriented Architecture for Distributed Publish /Subscribe Middleware in ElectronicsProduction", IEEE Transactions on Industrial Informatics, 2006. Page(s): 281 – 294.

[29] Parimala, N.; Saini, A. "Web service with criteria: Extending WSDL", Sixth International Conference on Digital Information Management (ICDIM), 2011. Page(s): 205 - 210

[30] D'Ambrogio, A. "A Model driven WSDL Extension for Describing the QoS of Web Services", International Conference on Web Services, 2006 ICWS '06. Page(s): 789 - 796

[31] Delamer, I.M.; Lastra, J.L.M.; Tuokko, R. "Unified service-oriented architecture for federatedand locally distributed CAMX publish/subscribe middleware", 3rd IEEE International Conference on Industrial Informatics, 2005(INDIN '05). Page(s): 117 - 122

[32] IPC-2501 Definition for Web-Based Exchange of XML Data (Message Broker) http://www.webstds.ipc.org/2501/IPC2501Pub.pdf

[33] IPC-2541 Generic Requirements for Electronics Manufacturing Shop-Floor Equipment Communication Messages (CAMX) http://www.fed.de/downloads/IPC-2541.pdf

[34] Thramboulidis, K.; Zoupas, A. "Real-time Java in control and automation: a model driven development approach" 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), 2005. Page(s): 8 pp. - 46

[35] Peschke, J. , "Real-time Java for industrial controls in flexible manufacturing systems", IEEE International Conference on Industrial Informatics (INDIN), 2003. Page(s): 325 - 331

[36] Mukherjee, A.; Tari,Z.; Bertok,P. "A Spring Based Framework for Verification of ServiceComposition", IEEE International Conference on Services Computing (SCC), 2011. Page(s): 258 - 265

[37] Rod Johnson, "Introduction-to-the-Spring-Framework", http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework

[38] Spring Web Services - Reference Documentation http://static.springsource.org/spring-ws/site/reference/pdf/spring-ws-reference.pdf

[39] Introduction to Spring Web MVC framework http://static.springsource.org/spring/docs/3.0.x/reference/mvc.html

[40] Mela, M.; Sakowicz, B.; Chlapinski, J. "Advertising service based on Spring Framework" Proceedings of International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2008. Page(s): 406 - 408

[41] Chien-Hung Liu; Kung, D.C.; Pei Hsia; Chih-Tung Hsu "Structural testing of Web applications", 11th International Symposium on Software Reliability Engineering (ISSRE), 2000. Page(s): 84 - 96

[42] Hypertext Transfer Protocol -HTTP/1.1 http://www.w3.org/Protocols/rfc2616/ rfc2616-sec1.html

[43] Java Servlet Technology
http://www.peterindia.net/ServletOverview.html

[44] Perrenoud, C.; Phan, K. “Emergence of web technology: An implementation of web accessibility design in organizations”, Technology Management for Emerging Technologies (PICMET), 2012: Page(s): 633 - 645

[45] Kailas Patil; Xinshu Dong; Xiaolei Li; Zhenkai Liang, “Towards Fine-Grained Access Control in JavaScript Contexts”, School of Computing National University of Singapore.
http://www.comp.nus.edu.sg/~liangzk/papers/icdcs11.pdf

[46] Duda, C.; Frey, G.; Kossmann, D.; Matter, R.; Chong Zhou “AJAX Crawl :Making AJAX Applications Searchable”, IEEE 25th International Conference on Data Engineering (ICDE), 2009. Page(s): 78 - 89

[47] Jesse James Garrett “Ajax-New-Approach-Web-Applications”
http://www. adaptivepath.com/ideas/ajax-new-approach-web-applications

[48] Doug Tidwell (XML Evangelist), EMC “Introduction to XML”
www.ibm.com/developerworks/xml/tutorials/xmlintro/

[49] Kevin Nilson, “Pushing Data to the Browser with Comet”
http://www.developer.com/tech/article.php/3756841/Pushing-Data-to-the-Browser-with-Comet.htm

[50] Wen Zhang, Junwei Cao, YishengZhong, Lianchen Liu, Cheng Wu: “Block-Based Concurrent and Storage-Aware Data Streaming for Grid Applications with Lots of Small Files”,
http://stuff.mit.edu/~caoj/pub/doc/jcao_c_block.pdf

[51] Ajax Push Engine
http://www.ape-project.org/ajax-push.html

[52] Direct Web Remoting
http://directwebremoting.org/dwr/introduction/index.html

[53] 3D Canvas
[http://3d-canvas.en.softonic.com/]

[54] University of Bristol’s Department of Mechanical Engineering “From human bite to robot jaws”

http://www.bristol.ac.uk/news/2009/6432.html

[55] Toledo Assembly Plants and Supplier Park
http://www.allpar.com/corporate/factories/toledo.html

[56] Kean Walmsley, “Calling a web-service from a Unity3D scene”
http://through-the-interface.typepad.com/through_the_interface/2012/04/calling-a-web-service-from-a-unity3d-scene.html

[57] Managing SOA environments
http://publib.boulder.ibm.com/infocenter/tivihelp/v24r1/index.jsp?topic=%2Fcom.ibm.itcamsoa.doc_6.2.2%2FOfferingGuide15.htm

[58] Service Statelessness
http://www.exforsys.com/tutorials/soa/service-autonomy/1.html

[59] The structure of a SOAP message
http://publib.boulder.ibm.com/infocenter/wmbhelp/v8r0m0/index.jsp?topic=%2Fcom.ibm.etools.mft.doc%2Fac55780_.htm

[60] Web Services, Part 2: WSDL and WADL
http://www.ajaxonomy.com/2008/xml/web-services-part-2-wsdl-and-wadl

[61] Spring Framework Introduction
http://static.springsource.org/spring/docs/2.0.x/reference/introduction.html

[62] Ajax and Service Oriented Architecture
http://www.openajax.org/member/wiki/Market_Overview_Whitepaper

APPENDIX 1: PALLET NOTIFICATION MESSAGE WSDL

```
<wsdl:definitions
target-
Namespace="http://fastory.com/fast/definitions"xmlns:wsdl="http://schemas.
xmlsoap.org/wsdl/"xmlns:sch="http://fastory.com/fast/schemas"xmlns:soap="h
ttp://schemas.xmlsoap.org/wsdl/soap12/"xmlns:tns="http://fastory.com/fast/
definitions">
<wsdl:types>
<xs:schemaelementFormDefault="qualified"
targetNamespace=http://fastory.com/fast/schemas
xmlns:hr=http://fastory.com/fast/schemas
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:elementname="NotificationMessage">
<xs:complexType>
<xs:attributeform="unqualified"name="dateTime"type="xs:string"/>
<xs:attributeform="unqualified"name="eventId"type="xs:string"/>
<xs:attributeform="unqualified"name="palletId"type="xs:string"/>
<xs:attributeform="unqualified"name="fromZoneId"type="xs:string"/>
<xs:attributeform="unqualified"name="toZoneId"type="xs:string"/>
<xs:attributeform="unqualified"name="cellID"type="xs:string"/>
</xs:complexType>
</xs:element>
</xs:schema>
</wsdl:types>
<wsdl:messagename="NotificationMessage">
<wsdl:partelement="sch:NotificationMessage"name="NotificationMessage"/>
</wsdl:message>
<wsdl:portTypename="FASTline">
<wsdl:operationname="information">
<wsdl:inputmessage="tns:NotificationMessage"
name="NotificationMessage"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:bindingname="FASTlineSoap11"type="tns:FASTline">
<soap:bindingstyle="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operationname="information">
<soap:operationsoapAction=""/>
<wsdl:inputname="NotificationMessage">
<soap:bodyuse="literal"/>
</wsdl:input>
</wsdl:operation>
</wsdl:binding>
<wsdl:servicename="FASTlineService">
<wsdl:portbinding="tns:FASTlineSoap11"name="FASTlineSoap11">
<soap:addresslocation="http://localhost:8080/fast
                      /fastory/informationService/"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

APPENDIX 2: EQUIPMENT CHANGE STATE MESSAGE WSDL

```
<wsdl:definitions
target-
Namespace="http://fastory.com/Fast2/definitions"xmlns:wsdl="http://schemas.xm
lsoap.org/wsdl/"xmlns:sch="http://fastory.com/Fast2/schemas"xmlns:soap="http:
//schemas.xmlsoap.org/wsdl/soap12/"xmlns:tns="http://fastory.com/Fast2/defini
tions">
<wsdl:types>
<xs:schemaelementFormDefault="qualified"
targetNamespace=http://fastory.com/Fast2/schemas
xmlns:hr=http://fastory.com/Fast2/schemas
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:elementname="EquipmentChangeState">
<xs:complexType>
<xs:attributeform="unqualified"name="dateTime"type="xs:string"/>
<xs:attributeform="unqualified"name="currentState"type="xs:string"/>
<xs:attributeform="unqualified"name="previousState"type="xs:string"/>
<xs:attributeform="unqualified"name="eventId"type="xs:string"/>
<xs:attributeform="unqualified"name="palletId"type="xs:string"/>
<xs:attributeform="unqualified"name="recipeNum"type="xs:string"/>
<xs:attributeform="unqualified"name="toolId"type="xs:string"/>
<xs:attributeform="unqualified"name="cellId"type="xs:string"/>
<xs:attributeform="unqualified"name="devType"type="xs:string"/>
</xs:complexType>
</xs:element>
</xs:schema>
</wsdl:types>
<wsdl:messagename="EquipmentChangeState">
<wsdl:partelement="sch:EquipmentChangeState"
name="EquipmentChangeState"/>
</wsdl:message>
<wsdl:portTypename="FASTline">
<wsdl:operationname="information">
<wsdl:inputmessage="tns:EquipmentChangeState"
name="EquipmentChangeState"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:bindingname="FASTlineSoap11"type="tns:FASTline">
<soap:bindingstyle="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
           <wsdl:operationname="information">
<soap:operationsoapAction=""/>
<wsdl:inputname="EquipmentChangeState">
<soap:bodyuse="literal"/>
</wsdl:input>
</wsdl:operation>

</wsdl:binding>
<wsdl:servicename="FASTlineService">
<wsdl:portbinding="tns:FASTlineSoap11"name="FASTlineSoap11">
<soap:address
location="http://localhost:8080/fast/fastory/informationService/"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

APPENDIX 3: JSON CONTROLLER CLASS

```
importorg.jdom.Element;
importorg.jdom.JDOMException;
importorg.springframework.stereotype.Controller;
importorg.springframework.web.bind.annotation.PathVariable;
importorg.springframework.web.bind.annotation.RequestMapping;
importorg.springframework.web.bind.annotation.RequestMethod;
importorg.springframework.web.bind.annotation.ResponseBody;

importcom.fastory.fast.model.Info;
importcom.fastory.fast.model.Informationmessage;
importcom.fastory.fast.model.abc;
importcom.fastory.fast.ws.InformationEndpoint;

importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.Map;
importjava.util.Set;


@Controller
@RequestMapping("/StopperInfoMsg")
publicclassJSONController {


           @RequestMapping( method = RequestMethod.GET)
           public@ResponseBody Info getdataInJSON() throwsJDOMExcep-
tion {
           String cellId= abc.cellID;
           String datetime=abc.dateTime;
           String eventid= abc.eventId ;
           String tozoneid= abc.toZoneId ;
           String palletid= abc.palletId;
           String fromzoneid= abc.fromZoneId ;


           Info info = newInfo();
           Informationmessage info1= newInformationmessage();

           info.setName("StopperInfoMsg");

           info1.setCellID(cellId);
info1.setDateTime(datetime);
           info1.setPalletId(palletid);
           info1.setEventId(eventid);
           info1.setFromZoneId(fromzoneid);
           info1.setToZoneId(tozoneid);

           info.setInformationMessage(info1);

           return info;
           }
}
```

APPENDIX 4: INFORMATION MESSAGE CLASS

```
packagecom.fastory.fast.model;

publicclassInformationmessage {

        String eventId;
        String toZoneId;
        String palletId;
        String cellID;
        String fromZoneId;

        String dateTime;

        public String getDateTime() {
                returndateTime;
        }
        publicvoidsetDateTime(String dateTime) {
                this.dateTime = dateTime;
        }
        public String getEventId() {
                returneventId;
        }
        publicvoidsetEventId(String eventId) {
                this.eventId = eventId;
        }
        public String getToZoneId() {
                returntoZoneId;
        }
        publicvoidsetToZoneId(String toZoneId) {
                this.toZoneId = toZoneId;
        }
        public String getPalletId() {
                returnpalletId;
        }
        publicvoidsetPalletId(String palletId) {
                this.palletId = palletId;
        }
        public String getCellID() {
                returncellID;
        }
        publicvoidsetCellID(String cellID) {
                this.cellID = cellID;
        }
        public String getFromZoneId() {
                returnfromZoneId;
        }
        publicvoidsetFromZoneId(String fromZoneId) {
                this.fromZoneId = fromZoneId;
        }

}
```

APPENDIX 5: INFO CLASS

```
packagecom.fastory.fast.model;

importcom.fastory.fast.model.Informationmessage;

publicclass Info {

            String Message;

            InformationmessageInformationMessage;

            public String getName() {
                        returnMessage;
            }
            publicvoidsetName(String Message) {
                        this.Message = Message;
            }
            publicInformationmessagegetInformationMessage() {
                        returnInformationMessage;
            }
            publicvoidsetInformation-
Message(InformationmessageinformationMessage) {
                        InformationMessage = informationMessage;
            }
            public Info() {
            }

}
```

APPENDIX 6: SPRING SERVLET CONFIGURATOIN

```
<?xmlversion="1.0"encoding="UTF-8"?>

<beansxmlns="http://www.springframework.org/schema/beans"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:context="http://www.springframework.org/schema/contex
t"

           xmlns:util="http://www.springframework.org/schema/util"

           xmlns:sws="http://www.springframework.org/schema/web-
services"

           xsi:schemaLocation="http://www.springframework.org/schema/b
eans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
       http://www.springframework.org/schema/web-services
http://www.springframework.org/schema/web-services/web-services-
2.0.xsd
       http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd
       http://www.springframework.org/schema/util
       http://www.springframework.org/schema/util/spring-util-2.0.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">
<context:component-scanbase-package="com.fastory.fast"/>


<sws:annotation-driven/>
<sws:static-wsdllocation="/WEB-INF/fast.wsdl"/>

<beanid="messageFactory"class="org.springframework.ws.soap.saaj.SaajSo
apMessageFactory">
<propertyname="soapVersion">
<util:constantstatic-
field="org.springframework.ws.soap.SoapVersion.SOAP_12"/>
</property>
</bean>

</beans>
```

APPENDIX 7: MVC DISPATCHER SERVLET CONFIGURATOIN

```
<beansxmlns="http://www.springframework.org/schema/beans"
           xmlns:context="http://www.springframework.org/schema/contex
t"
           xmlns:mvc="http://www.springframework.org/schema/mvc"xmlns:
xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-
3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-
3.0.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

           <context:component-scanbase-
package="com.fastory.fast.controller"/>

           <mvc:annotation-driven/>
           <mvc:resourcesmapping="/resource/**"location="/resource/"/>

</beans>
```