INTERACT – Interactive Manual Assembly Operations for the Human-Centered Workplaces of the Future

Grant Agreement Number	**:**	**611007**
	:	**INTERACT**
Project Start Date	**:**	**1st October, 2013**
Consortium	**:**	**DAIMLER AG (DAIMLER)- Project Coordinator** **ELECTROLUX ITALIA S.P.A. (ELECTROLUX)** **INTRASOFT INTERNATIONAL SA (INTRASOFT)** **IMK AUTOMOTIVE GMBH (IMK)** **EMPHASIS TELEMATICS AE (EMPHASIS)** **HADATAP SP ZOO (HADATAP)** **UNIVERSITY OF PATRAS (LMS)** **UNIVERSITAET ULM (IMI)** **DEUTSCHES FORSCHUNGSZENTRUM FUER KUENSTLICHE INTELLIGENZ GMBH (DFKI)**

Title	**:**	**Applications requirements analysis and architecture design**
Reference	**:**	**D5.1.1**
Availability	**:**	**Public (PU)**
Date	**:**	**30/09/2014**
Author/s	**:**	**DAIMLER, ELECTROLUX, INTRASOFT, IMK, EMPHASIS, HADATAP, IMI, DFKI**
Circulation	**:**	**EU, INTERACT consortium**

Summary:

This deliverable describes the INTERACT Enterprise Application Platform (EAP) architecture as well as each application (app) analysis in terms of data and communication. EAP is holistic framework for supporting application lifecycle, management, access rights etc. Furthermore provides a unified way to data access.

The INTERACT project (611007) is co-funded by the European Commission under the 7th Framework Programme.

Table of Contents

Table of Figures

1. EXECUTIVE SUMMARY

The purpose and the scope of this deliverable are to describe in details the architectural design that addresses the requirements defined in deliverable D1.3.1 “Application requirements for the human centered manual assembly workplace” in respect to the pilot cases described in D1.4.1 “Industrial pilots definition”. Furthermore this deliverable identifies the technical relationship between workpackages and documented wherever applicable in section 5 “Service/business layer”.

The document in hand is divided under 6 major chapters, “Generic architectural approach”, “Apps”, “Service/business layer”, “Persistent storage” and “Conclusions”. The first unit provides a holistic view of the system defining the architecture design in chapter 3. In the subsequent chapters 4, 5 and 6 focus is given to different architectural aspects and specifically to apps analysis, business layer for both EAP and apps and persistent storage. Finally, the “Conclusion” in chapter 8 documents the results of this deliverable.

2. INTRODUCTION

Even if the requirements were derived from two different pilot cases the procedure that was followed was to combine them and come up with a single solution that will satisfy both pilot cases. This work was mainly conducted under WP1 and the outcome were documented in D1.3.1 by defining the core functionalities of the EAP (Enterprise Application Platform) as well as the individual functionalities of the apps that will be hosted in EAP.

The work conducted under workpackage 5 was mainly on the technologies that can be used to fulfill the afore mentioned requirements as well as the overall architecture that will allow the individual apps to take advantage of these technologies in a seamless way. Also a second pass of the requirements was conducted in order to identify common application requirements as well as not clearly specified nonfunctional requirements. The new findings where:

1. "3D Scene Viewer" app functionalities are to be utilized by the majority of the apps.
2. Performance is a main nonfunctional requirement that concerns the data manipulation as well as the data retrieval (from persistence storage).

3. GENERIC ARCHITECTURAL APPROACH

From an architectural point of view the main decision was already taken as de facto solution and was concerning the usage of thin or fat clients. From the advance of web technologies (thin clients), HTML 5 advantages (xml3d) and the partner's interests and experience, a web based approach has being decided.

The proposed architecture is a web based approach compromising an application stuck (layered architecture) on the backend side with SOA principles (for the integration of services/modules from other workpackages) along with a main EAP UI application serving as a backend user interface and apps manager.

The apps themselves will be standalone web apps with an HTML5 front end and a lightweight server side controller that will expose/consume services to/from the EAP backend side.

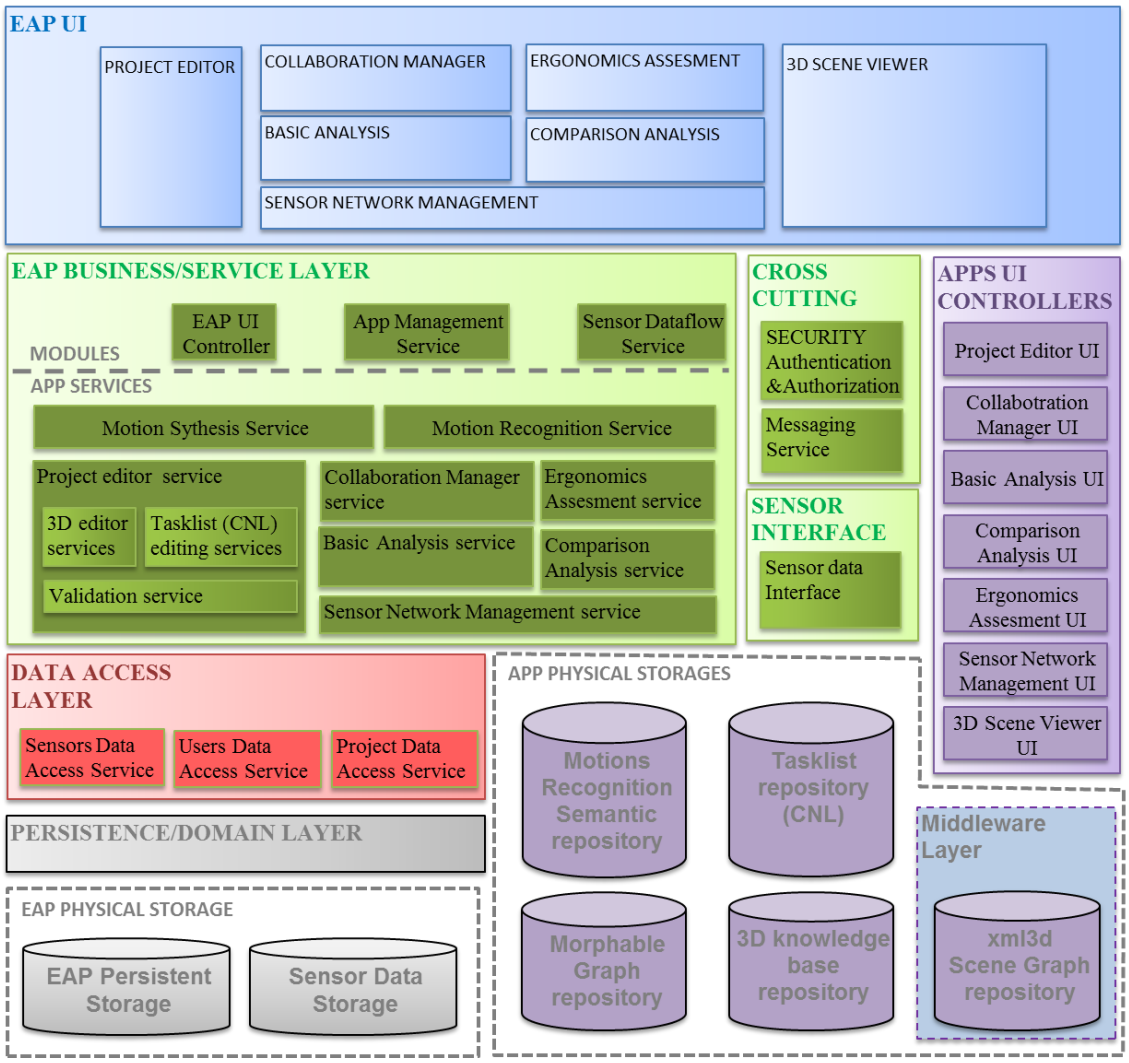

Figure 1: Layered Architecture

Another way of viewing the proposed system is the deployment view. The EAP UI and backend core services will be developed as a single web application residing in an application server. The same is true for the apps with the difference that the app's services might not be part of the standalone app but can be developed as a separate service residing either internal in the EAP backend services or as a separate standalone service. This way the apps will develop a standard communication interface with

the services needed for the application purposes enhancing the modularity of the system and decreasing the overall coupling. The benefits are:

- Changes to one module does not break other modules
- Easy to reuse and test, easy to maintain
- Single-responsibility and separation of concerns

On the other hand Message Creation, Transmission, Translation, Interpretation overheads affect the system performance but whenever performance is required tight coupling can be implemented without affecting the overall architecture.

3.1. EAP Layered Architecture

3.1.1. Presentation Layer (EAP User Interface)

The EAP User Interface serves as the app orchestrator (providing a desktop in which the INTERACT apps will reside) as well as providing functionalities borrowed by the "Enterprise Application Store" paradigm (Apps management, Authentication & Authorisation).

The user interface will implement the MVC (Model-View-Controller) design pattern implementing HTML5 principles (responsive UI controls, websockets etc.).

3.1.2. EAP Business Layer

This layer represents the logic of the EAP platform.

The core functionalities that EAP business layers implements are: the User Interface controller (EAP UI Controller), the App Management services and the Sensor Dataflow service. All these provide support for the apps to be developed in order to ease development effort providing out of the box implementations to communication and collaboration functionalities.

Since the EAP backend will be responsible for the overall security there is no need for a Central Authentication Service (CAS) removing the burden of individual apps to adhere to the CAS paradigm.

On section "5.2 Core Services" details of each core services are provided.

3.1.3. Data Access Layer

The Data Access Layer consists of components providing functions for accessing and storing data. Each component represents a sub-domain of data for ease of manipulation. The major data domains that have been identified so far are:

- The Project domain. This domain will include data related to a specific project that a user can create. The main characteristic of this subdomain is that it does not actually hold the data but references to the data that all together compose the "project". Specifically will include links to
    - 3D data (3D Scene)
    - tasklist (describing in a CNL based language the task that should be performed)
    - sensors and sensor configurations
    - project version data allowing version control and comparison of different project versions.
- The Sensor data domain. The Sensor domain will group sensor related data.
- The User domain. The user domain is responsible for user and role management as well as access-control management.

Each of the above data domains correspond to a specific Data Access service (Project Data Access service, Sensor Data Access service and User Data Access service respectively) which implements a Rest interface for data provision to other local and/or remote components in a seamless way.

3.1.4. Domain Layer / Persistence Storage

This layer includes the physical storage of data (databases, ontology servers, xml servers, file systems etc.).

It has been known early from the specifications analysis that a single storage (i.e. database) will not be capable to handle the overall needed data requirements but the INTERACT storage will include different data sources with interrelated references.

More information of INTERACT storage can be found in section 6.

3.1.5. Cross Cutting

Cross-cutting components are the components that are required in more than a single layer. EAP cross cutting layer include Authentication & Authorization services and Messaging Service. Details about these services can be found in sections 5.1.1 and 5.1.2 respectively.

3.1.6. Interfaces

Up to now the interface with the sensors system has been identified. A detailed description of the system to be integrated can be found in deliverable "D3.1.1: Detailed sensors system architecture and design" and more specifically on deliverable's section "3.5 Sensor Management Platform".

From the EAP point of view, sensor data are to be distributed in a seamless way to whoever application/module needs them with the configuration of the Dataflow engine. The integration will be implemented by the use of "scheduled jobs" (wherever necessary) and a lightweight "Enterprise Service Bus" as a transport layer supporting a big variety of protocols like HTTP based protocols (i.e. websockets, Large Files Streaming), text based protocols (i.e. telnet), binary based protocols, UDP-based Data Transfer Protocols (i.e. UDT). For data intense transfer (big streaming of data, large files) a high-throughput distributed system is going to be implemented.

3.1.7. Module Communication

Most of the EAP modules are self-sufficient and they rely only on Data Access layer to store and retrieve data.

APP modules on the other hand need communication either in terms of simple message passing (i.e. Tasklist editing service updates storage and must inform the Motion Synthesis service to "recalculate" motion) which is supported by the Messaging service or require a specific function (i.e. Motion Recognition service in Motion Synthesis service which requires an elementary action list to provide constrains, i.e. time constrains). In such a case a Service Oriented Architecture (SOA) approach is proposed meaning that services must provide an interface (preferably Rest interface) in order to be used by other components.

Furthermore subsystems with jointly effort like work task editing, project validation services, project variant comparison and motion synthesis service will adhere to the same principles for integration.

More on the specific services and their components interrelationship can be found under section "5.1 Application Specific Services".

As stated previously SOA based architecture can introduce performance issues but the principle "Design Loosely & Implement Tightly" wherever performance matters will be followed.

3.1.8. Implementation

There is a big variety of frameworks that implement the proposed layered architecture in multiple programming languages. For a multi-platform implementation Java programming language is the *de facto* standard and by using Java the Layered Architecture's *de facto* standard implementation is the Spring framework.

The Spring Framework provides a dynamic programming and configuration model for Java-based enterprise applications. The key element of Spring framework is the application level support. Spring

provide the skeleton elements of an enterprise applications so that development teams can focus on application-level business logic.

3.2. App Layered Architecture

As previously stated each app will conform to the layered architecture having a user interface (i.e. MVC pattern implementation), a business layer, a data access layer and a service integration layer for consuming either backend services (mainly for data provisioning) or own app's services.

A standalone app's architecture is depicted in Figure 2.

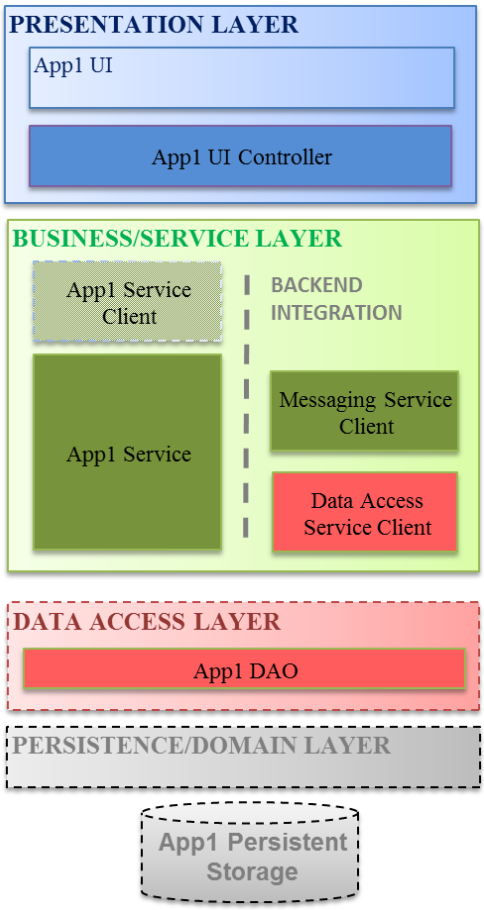

Figure 2 App's architecture

3.2.1. Presentation Layer

The presentation layer consists of the User Interface and the application controller part of the MVC implementation.

3.2.2. Business Layer

The business layer is separated into the core app services and the backend integration services.

The app service implements the core functionality of the app. It can be integrated into the app's layer stack or integrating an external service in which case the app's business layer will include a client for accessing the service (App1 Service Client).

The backend integration services are client services that are communicating with the EAP backend for data manipulation (Data Access Service Client) and for communication (Messaging Service Client).

3.2.3. Data Access, Domain Layer & Persistence Storage

These layers are considered optional since some apps may find all the necessary data for their functionality on the Backend's Data Access service. In such a case these layers can be omitted,

otherwise the app can have access to an internal storage (through the Data Access Layer) to fetch additional data to help the app perform the required function.

4. APPS

For the apps development a framework will be provided based on the same principles EAP UI will be developed. Based on this framework each app will have access to a common set of functionalities that will ease the app development. These functionalities consist of services at the backend side as well as user interface components that integrate with the EAP UI.

The apps can reside on their own server as long as there is access to the EAP backend services for data retrieval and storage. Below is a basic sequence of events for an app that displays some metrics.

1. User clicks on App1 calculate button
2. Request reaches App1's controller
3. App1's controller forwards the request to an "App Service" component.
4. "App Service" calls EAP's backend Data Access service to retrieve a relevant dataset
5. "App Service" process the data
6. "App Service" forwards the result to the App1's controller
7. App1's controller sends response to be displayed in the user's screen (App1)

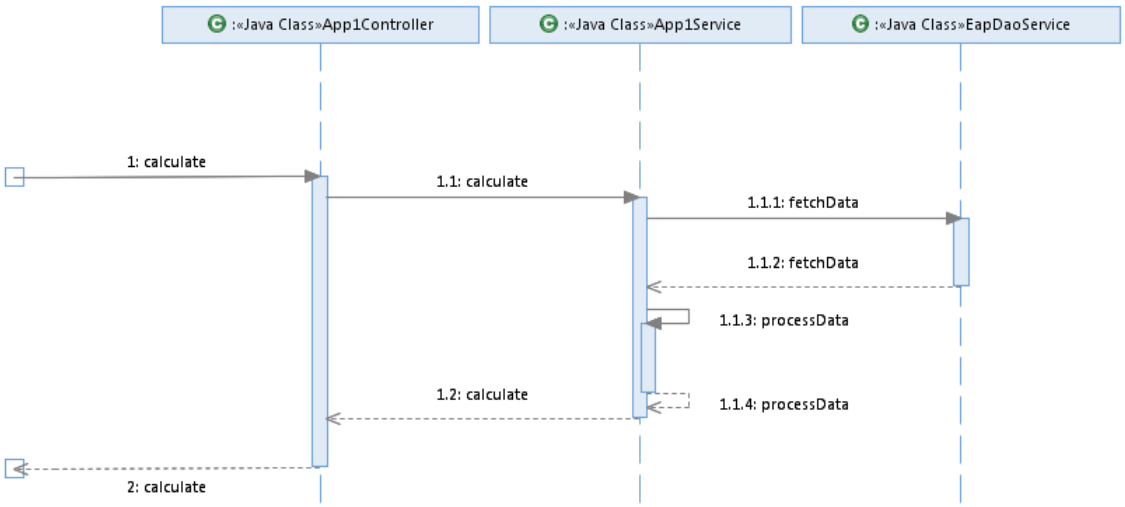

Figure 3 Application communicating with the EAP Data Access service

Apps are categorized as basic or non-basic. Basic apps are those apps that are considered essential for the overall EAP platform experience (3D Scene Viewer, Collaboration Manager) whereas non basic apps are the ones that provide their own aspect over a specific EAP project. For example analysis apps (Basic Analysis, Ergonomics Analysis, and Comparison Analysis) that calculate metrics are apps that are used by a specific role (i.e. ergonomics manager) providing a single aspect of the project (i.e. ergonomic evaluation). On the other hand 3D Scene Viewer app will be used by each role since provides a visual representation of the project.

Next we will present each app's interaction with the available services as well as identified interaction with storages. The notation followed in the subsequent sections:

- The white boxes represent actions/services/modules
- The blue boxes represent data storage
- The arrows represent data-flows

- Common components outside of the app that the app interacts with are stated with a dashed line

4.1. Basic Apps

4.1.1. 3D Scene Viewer

This app will be part of the "basic" apps which will be shipped, pre-installed in the EAP. This application will handle the visualization and the interaction with the virtual environment having two distinct modes, "navigation" and "interaction". In the "navigation" mode the user cannot interact with the 3D environment (view only) whereas in "interaction" mode the user will be able to add and remove geometries as well as move geometries around to create the 3D scene of his preference.

The 3D Scene Viewer app connects directly to the xml3d Scene Graph utilizing the storage's middleware capabilities.

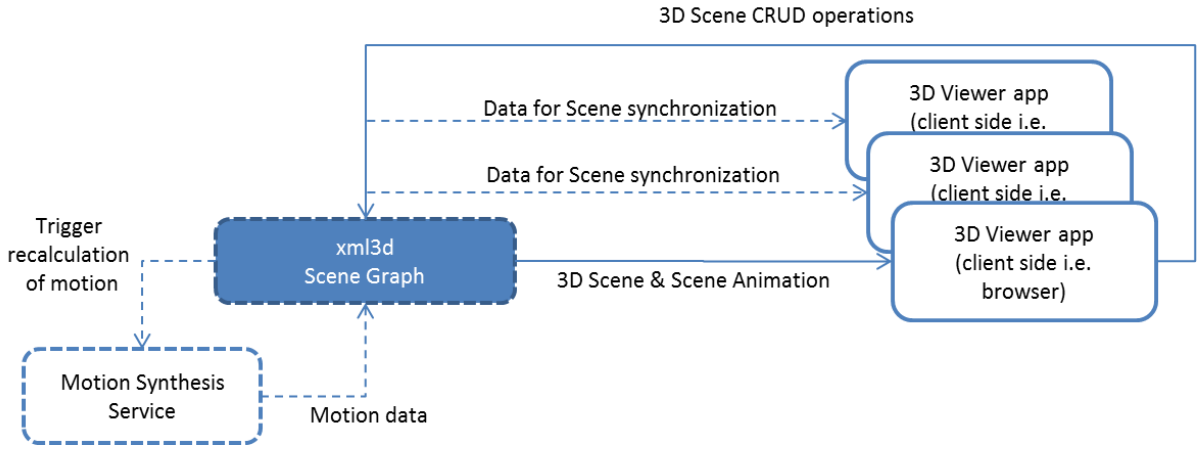

Figure 4 3D Scene Viewer

4.1.2. Collaboration Manager

This app will be responsible for coordinating a collaborative session. It will communicate with the EAP platform for credentials acquisition and will initiate each application in a predefined "mode" according to user's role authority utilizing the App Management Service.

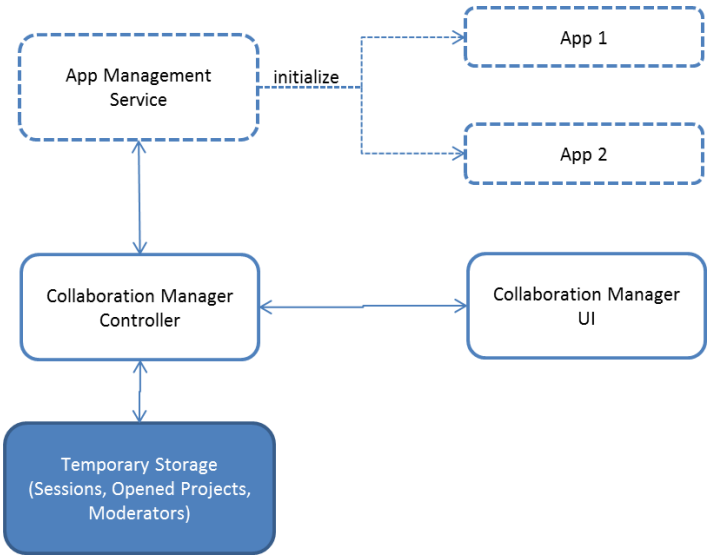

Figure 5 Collaboration Manager

4.2. Other Apps

4.2.1. Sensor Network Management

This app provides a user friendly mechanism for sensor configuration by storing sensor parameters and states in order to be used later on for a quick project setup.

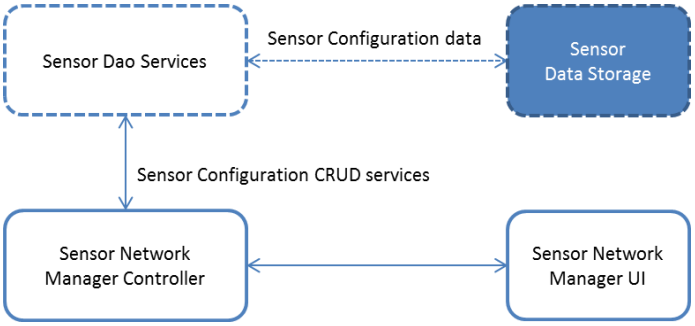

Figure 6 Sensor Network Management

4.2.2. Basic Analysis

The basic analysis app provides metrics on a project such as times on tasks, walking paths and collision risks.

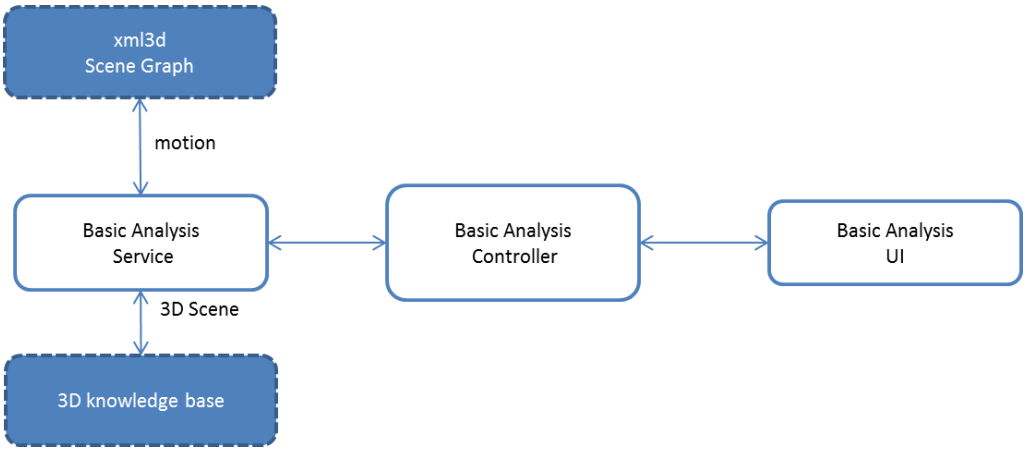

Figure 7 Basic Analysis

4.2.3. Comparison Analysis

This app is responsible of analyzing two different version of a project and provides output according to MTM analyses. For MTM analysis it utilizes the motion and the Work Tasklist (CNL) for determine the MTM-1 elements and corresponding times. Comparison Analysis app also works closely with the 3D Viewer app using the App Management service as the orchestrator.

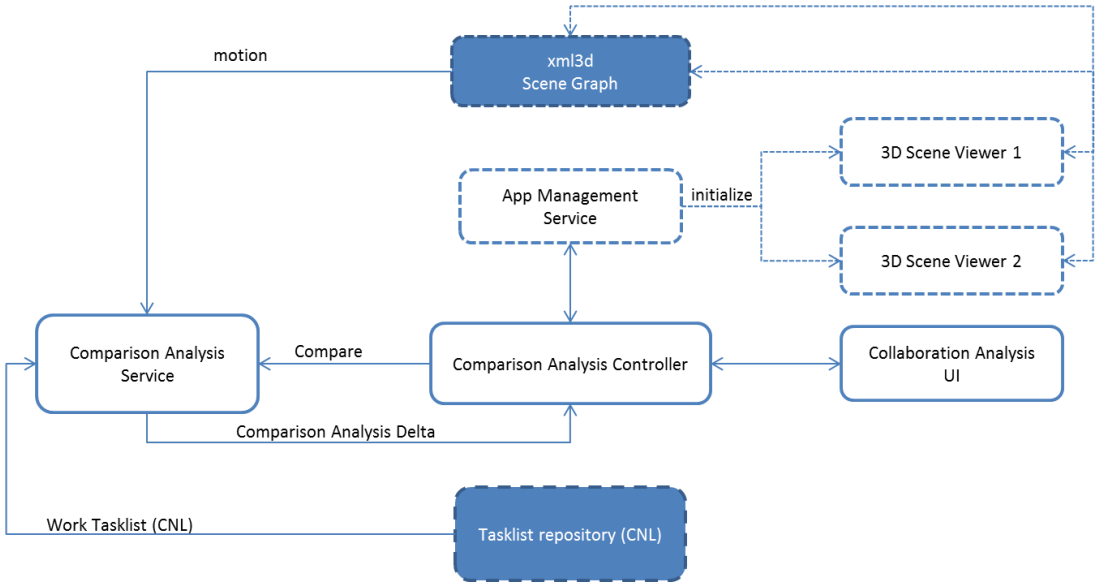

Figure 8 Comparison Analysis

4.2.4. Project Editor

This is the main editing application. It works in conjunction with the 3D Viewer app (utilizing App Management service) for 3D scene manipulation but also contains tasklist editing functionality by the use of CNL. Finally is capable to trigger a motion recalculation to synchronize the xml3D Scene Graph with updated motion.

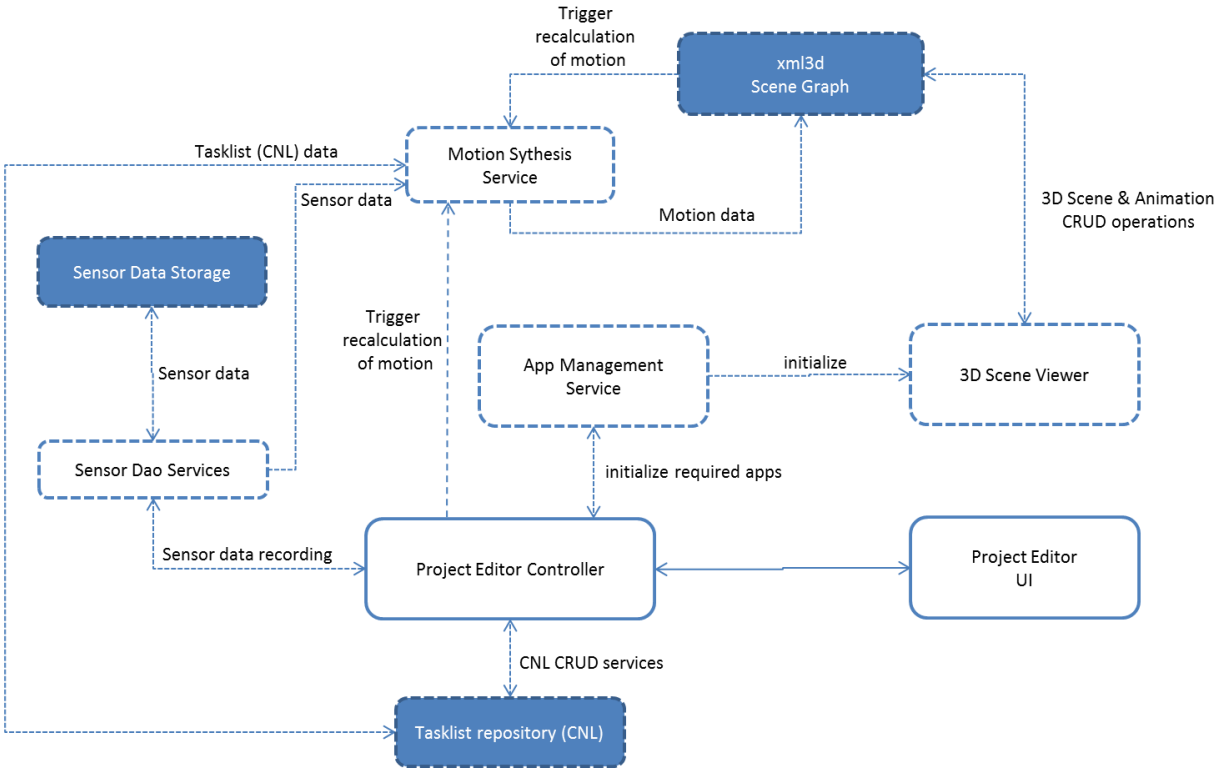

Figure 9 Project Editor

4.2.5. Ergonomics Assessment

This app performs ergonomic assessment on the human motion. It mainly provides metrics for musculoskeletal disorders of workers with manual activities.

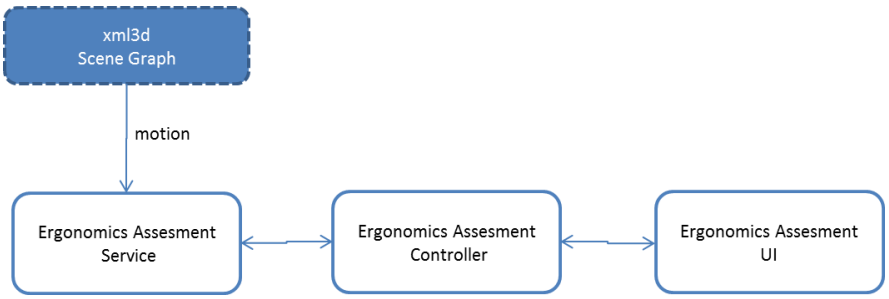

Figure 10 Ergonomics Assessment

5. SERVICE/BUSINESS LAYER

5.1. EAP Services

5.1.1. Authentication & Authorization Service

For the authentication and access-control (authorization) the Spring Security framework will be used. It is a highly customizable framework that provides out of the box protection against attacks (fixation, clickjacking, cross site request forgery etc.) and integrates with Spring MVC for web authorization.

For Spring MVC the Spring Security is implemented by the provision of a "filter" class that process all the requests towards the web application.

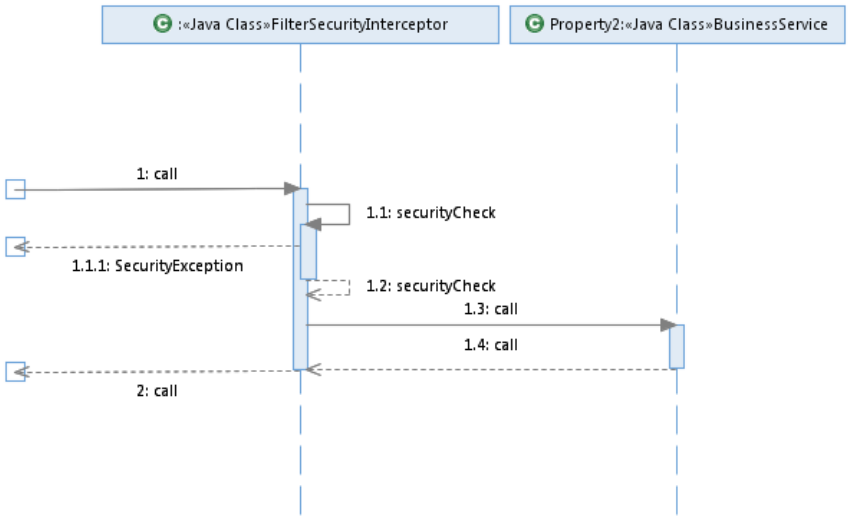

Figure 11 Spring Security sequence diagram

5.1.2. Messaging Service

For inter app communication as well as for coordination/communication between modules a messaging service will be implemented on top of a lightweight ESB (Enterprise Service Bus). EAP will also expose an ESB interface with Rest services to ease the integration effort. For the implementation "Apache Camel" framework will be used in conjunction with the "Apache ActiveMQ" messaging support.

5.1.3. App Management Service

This service is responsible to initialize application upon user request. The user asks EAP to access a specific application (i.e. Clicks: Menu->Apps->Project Editor), and it is the EAP platform's responsibility to initialize the app to the appropriate state (i.e. read-only, editing etc.) according to user's rights.

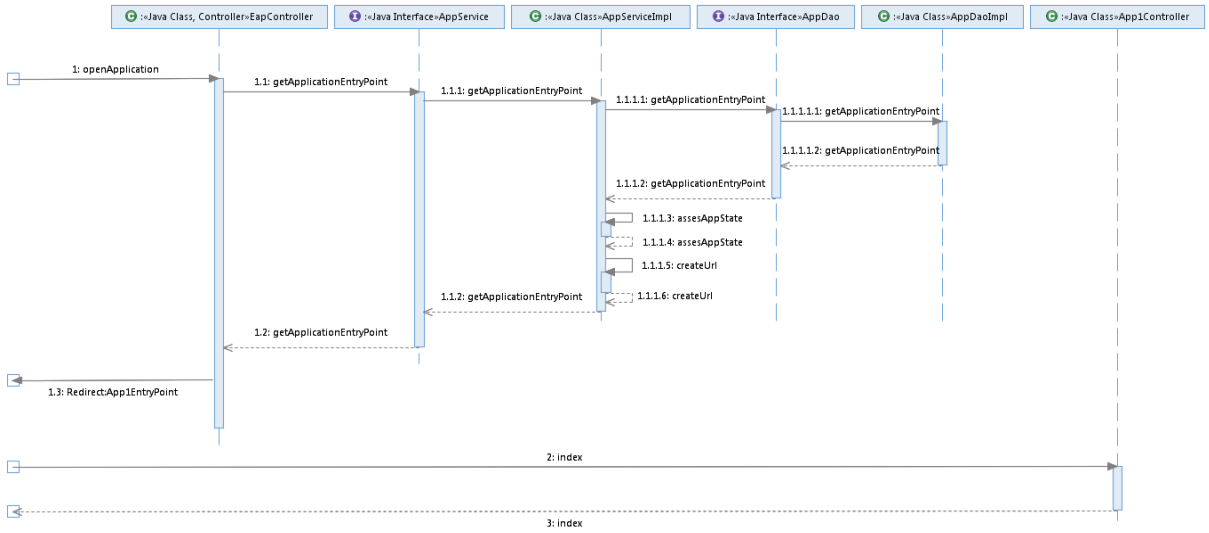

Figure 12 Sequence diagram of an end user accessing an app (app initialization)

5.1.4. Motion Synthesis Service

This service is responsible for the human motion synthesis.

It is composed by several other components deriving from the work of WP2 and WP4.

Below is an overview of the interrelated components.

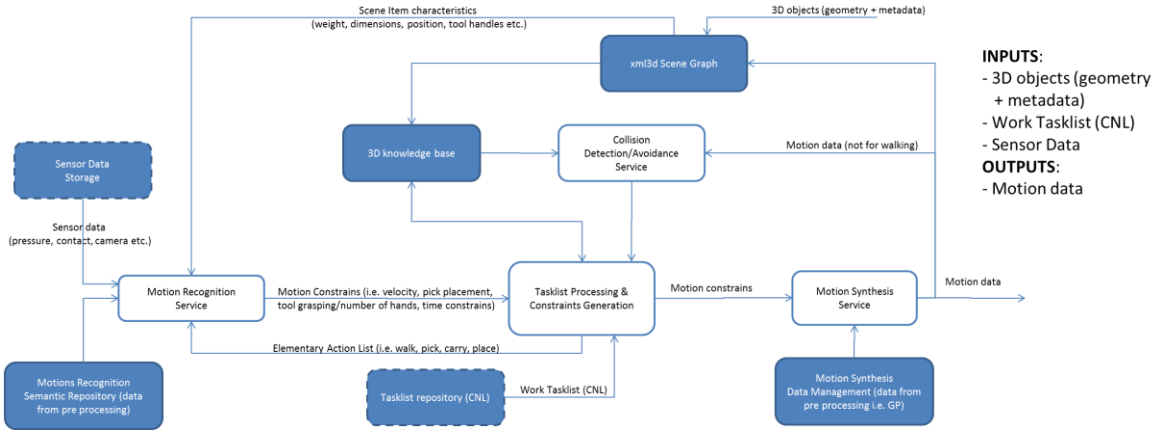

Figure 13 Motion Synthesis Service

5.1.5. Motion Recognition Service

This service is responsible for processing of sensor data in order to extract motion constrains to be used in the motion synthesis. It connects to and maintains the Motions Recognition Semantic repository.

5.2. App Specific Services

The application specific services are considered part of each app development. These services will be documented on the originating work package as their development proceeds. Below you can find a quick description of each service responsibilities and interactions.

5.2.1. Sensor Network Management service

This service implements the business logic of the "Sensor Network Management" app and provides the "sensor" configuration for a specific project. This service will be developed in the context of WP3

5.2.2. Basic Analysis Service

This service implements the business logic of the "Basic Analysis" app. It consumes motion related data (i.e. bounding volume hierarchy files) combined with 3D geometrical data to produce walking paths and collision risk. This service will be developed in the context of WP2

5.2.3. Comparison Analysis service

This service implements the business logic of the "Comparison Analysis" app.. The inputs are two different project versions (specifically each project's calculated motion and work tasklist) and the output is a list of delta in terms of time as well as operations between these two project version scenes. This service will be developed in the context of WP2.

5.2.4. Project Editor Service

This service correlates the functionalities of 3D Scene editing, CNL editing, Sensor configurations and Sensor data into single instance that is called a project. Also performs validation between these associations in order to make sure that a project is in a consistent state (i.e. all geometric elements that are described in the Tasklist exist in the 3D environment). This service will be developed in the context of WP5.

5.2.5. Collaboration Management Service

This service implements the business logic of the "Collaboration Manager" app. It maintains a list of sessions each associated with a project and a moderator. The moderator is considered the one who creates the collaborative session and has access to manipulate project data while others joining the session have only "read only" access to project data. This service will be developed in the context of WP5.

5.2.6. Ergonomics Assessment Service

This service implements the business logic of the "Ergonomics Assessment" app. It requires as input a human motion and produces ergonomic assessments in various standards (EAWS, NIOSH standard etc.). Furthermore interacts with the xml3D Scene Graph in order to provide real-time coloring of affected body parts. This work is performed in the context of WP2.

5.3. Interfaces

5.3.1. Sensors Interface

The Sensor data are to be distinguished into two major categories depending on the data magnitude. Sensor data like pressure, contact etc. are not considered to create big data and can be integrated with the use of Rest services for data storage. On the other hand data from sensors like camera which produce a big amount of data are to be interfaced by either streaming technics like Data Stream Management System (similar to database management systems but for "dynamic" data) where processing is needed real time or by a distributed file system that provides high-throughput access to big data when data are to be processed offline.

6. PERSISTENT STORAGE

The INTERACT storage and their interrelationship can be depicted in the following diagram

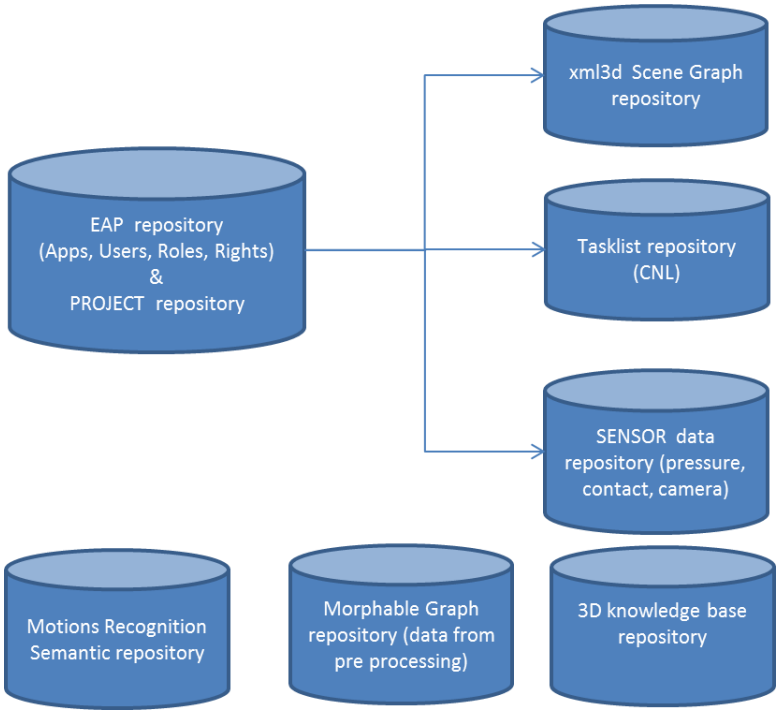

Figure 14 INTERACT persistence storage

Apart from the services internal data storages (Motion Recognition Semantic repository, Morphable Graph repository and 3D Knowledge Base repository) the INTERACT main repositories (xml3d Scene Graph, Tasklist and Sensor Data) are linked together with the use of references by the main EAP repository and form the "Project" as the main entity of the whole platform that binds the apps together.

6.1. Storage Particularities

6.1.1. EAP Persistence Storage & Sensor data repository

The Enterprise Application Platform's storage is separated into a dedicated EAP storage and "Sensor Data" storage. EAP storage manages data for user, role and right as well as application configuration and project reference data. It is considered to be implemented in a traditional RDBMS database. Sensor Data storage on the other hand has specific requirements of big data that RDBMS either cannot support or supports them with the drawback of performance. For Sensor Data implementation it is considered the use of a distributed file system (Hadoop) for offline processing and a data streaming framework for real-time data stream processing (Apache Strom).

6.1.2. Motion Recognition Semantic repository

This repository will store semantic data and will be implemented inside an ontology server. This repository is an internal repository of the "Motion Recognition" service. More details of this repository can be found in "D4.1.1 Design of aggregated datastore for planned and sensor data" and specifically in sections 4 and 5.

6.1.3. Morphable Graph repository

This repository will store pre-processing data like motion primitives and morphable functions. This repository is an internal repository of the "Motion Synthesis" service. More details related to the purpose of this repository can be found in "D2.2.1 Best-fit simulation method and first software prototype" report as well as in "D2.1.1 Manual assembly simulation design and data format".

6.1.4. xml3d Scene Graph repository

This repository contains all the element geometries of the 3D scene. It also handles assets on these elements as well as simulation data (human motion). The particularity of this repository comes from the existence of a middleware layer between the storage and the outside word. This middleware layer provides its own API for data manipulation.

6.1.5. Tasklist (CNL) repository

This repository stores tasklist related information in CNL language. For the implementation traditional RDBMS system should be efficient.

6.1.6. 3D Knowledge base repository

This repository is designed to store a simple representation of the 3D scene. The existence of such storage provides a small amount of data redundancy but it is needed for performance reasons wherever the full 3D scene must be processed (i.e. to calculate the walk path or to assess collision risks of the whole motion inside the 3D environment). Data synchronization between this repository and the "xml3d Scene Graph" repository is unidirectional (from "xml3d Scene Graph" to "3D Knowledge base") and is performed on an ad hoc basis (whenever xml3d data change).

7. CONCLUSIONS

This document apart from the specific Enterprise Application Platform architecture it provides a holistic view of the INTERACT system. Core EAP services as well as specific application services and central components have been identified and the cooperation between them has been documented providing the base of the Enterprise Application Platform (EAP) implementation as well as design principles for the apps development.

The focus of this document from the beginning was a clear architectural design that does not pose restrictions on app development but rather ramp up development by providing the "big picture" and the "role" of each component to achieve the INTERACT goals.

A solid cooperation between workpackages outcomes also is depicted and interfaces between components are identified (in terms of generic data description) which set the base for the different development groups to reach to a mutual data exchange model between the consumer of the data and the data provider.

8. GLOSSARY

RDBMS	Relational Database Management System
EAP	Enterprise Application Platform
UI	User Interface
SOA	Service Oriented Architecture
DAO	Data Access Object
MVC	Model-View-Controller
CAS	Central Authentication Service
ESB	Enterprise Service Bus
CNL	Control Natural Language