Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Mikael Torvinen

DESIGN AND IMPLEMENTATION OF GEOGRAPHIC INFORMATION SYSTEM FOR MOBILE MEASUREMENT DATA

Examiners : Professor Jari Porras
Junior Researcher, M.Sc. Antti Herala

Supervisors: B. Eng. Mikko Kilkki

ABSTRACT

Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Mikael Torvinen

Design and implementation of geographic information system for mobile measurement data

Master's Thesis

53 pages, 9 figures, 5 tables

Examiners: Professor Jari Porras
Junior Researcher, M.Sc. Antti Herala

Keywords: geographic information system, OGC, component selection

Nowadays maps often come in digital form, but still are mostly used to display static content. By adding a time component, a map can be used to display mobile objects like car, people or measurements. In this thesis, design science approach is used to implement a near real-time geographic information system for displaying mobile measurements. Open Geospatial Consortium standards and third party software components are utilized in the design. In addition, software component selection process appropriate for the scale of the project is developed. Open Geospatial Consortium standards were found to be adequate for use in the system. It was noted that attention is needed when choosing between the partially overlapping standards. The software component selection process was found to be successful. Comparison of non-functional aspects of commercial and open source components was found challenging.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Business and Management
Tietotekniikan koulutusohjelma

Mikael Torvinen

Liikkuvan mittatiedon geograafisen tietojärjestelmän suunnittelu ja toteutus

Diplomityö

2016

53 sivua, 9 kuva, 5 taulukko

Työn tarkastajat: Professori Jari Porras
Nuorempi tutkija, DI Antti Herala

Hakusanat: geograafinen tietojärjestelmä, OGC, komponentin valinta
Keywords: geographic information system, OGC, component selection

Nykyään kartat ovat usein digitaalisia, mutta silti niitä käytetään enimmäkseen staattisen sisällön esittämiseen. Lisäämällä aikakomponentin kartalla voidaan esittää liikkuvia kohteita, kuten autoja, ihmisiä tai mittatietoa. Tässä työssä suunnittelututkimuksen keinoin toteutetaan lähes reaaliaikainen geograafinen tietojärjestelmä liikkuvan mittatiedon esittämiseen. Suunnittelussa hyödynnetään Open Geospatial Consortium standardeja ja kolmannen osapuolen ohjelmistokomponentteja. Lisäksi kehitetään projektin koolle sopiva ohjelmistokomponentin valintaprosessi. Open Geospatial Consortium standardit todettiin järjestelmään sopiviksi. Osin päällekkäisten standardien valinnan havaittiin vaativan huomiota. Komponentin valintaprosessi todettiin onnistuneeksi. Avoimen lähdekoodin ja kaupallisten komponenttien vertailu ei-toiminnalliselta osalta osoittautui haasteelliseksi.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my employer, Environics Oy, for giving me the opportunity to study this interesting topic.

Secondly, I would like to thank Lappeenranta University of Technology. Not only for the guidance I had writing this thesis, but also for the high quality education I have received over the years.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface
CARE	COTS-Aware Requirements Engineering
CBSE	Component Based Software Engineering
CMMI	Capability Maturity Model Integration
COTS	Commercial Off-The Shelf
GIS	Geographic Information System
CQL	Common Query Language
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
OMM	OpenSource Maturity Model
OpenBQR	Open Business Quality Rating
OSS	Open Source Software
OTSO	Off-The Shelf Option
PORE	Procurement-Oriented Requirements Engineering
SOS	Sensor Observation Service
SPA	Single Page Application
STACE	Social-Technical Approach to COTS Evaluation
SQL	Structured Query Language
WFS	Web Feature Service
WFS-T	Web Feature Service with Transactions
WMS	Web Map Service
WMTS	Web Map Tile Service

1 INTRODUCTION

In this chapter, the background, goals and delimitations for this thesis will be laid. Research method and core questions will be defined and finally, overview of the thesis structure is given.

1.1 Background

Displaying and browsing geospatial information, typically a map, has for a decade or so been primarily a digital experience. Maps have a long history spanning over centuries in the form of a print or a painting. The availability of web map services and portable digital devices capable of not only displaying the map, but also locating the user on it, has made a map in its tangible form almost extinct.

This new era of digital maps comes with many benefits and new features, such as route finding and overlaying the map with points of interest like hotels and service stations. However, even the new digital maps are mostly static and cannot display temporal information very well. Maps often feature static objects like roads and rocks, but not mobile objects like cars and people.

There is a trend where the declining cost of hardware has led to the rise of smart connected devices in various industries. This is often referred as the Internet of Things. With many measurement devices aware of their location and capable of communicating their state out there, the need for storing this kind of data, but also displaying and analysing it in real-time, is growing. This would require a real-time Geographic Information System (GIS) capable of storing both temporal and spatial properties and updating them in real-time.

1.2 Goals and delimitations

The goal of this thesis is to **design and implement a near real-time GIS for displaying data from mobile measurement devices**. This system will cover retrieving measurements from devices, storing them and finally visualizing them to the user.

The system to develop will rely heavily on standards of The Open Geospatial Consortium (OGC). This will enable the use of the wide selection of existing components and libraries and effectively makes it possible to implement the relatively large system with the limited resources available. It will also be a great assent if using geospatial data from other sources or integrating the system as data source to other systems would ever be considered.

Attention is paid on selecting suitable front-end map component. This component will be integral part of the front-end of the software, and thus the selection is crucial for current and future development. Component selection in the context of software systems development will be discussed and based on findings in literature, a suitable selection process for this case is defined. The process will then be applied to select between a few candidates to determine the most suitable map component.

The work done in this theses is a study on platform development for real-time GIS applications displaying mobile measurement data. To demonstrate the functionality of such platform, a real data source in form of the Linssi database is used. The resulting software will be providing reach back functionality to radiation measurement data in the database. The software will not be full-fledged product, rather a proof of concept, but it could be later developed into one, if so desired.

As delimitation, based on experiences in past projects, the back-end components are mostly not subject to component selection. This is due to lack of viable alternatives in easily extendable OGC compatible GIS servers and will to take advantage of existing knowledge in the field.

1.3 Research method

The research method used in this thesis is design science. As Hevner et. al. (2004) states, science generally seeks truth, but the goal of design science is utility. Artefacts that address relevant problems in novel ways are designed using existing knowledge and their utility will be evaluated.

According to Hevner et. la. (2004), design science research is executed in iterations of two activities: building and evaluating artefacts. The first one develops a design artefact addressing the business needs using the existing knowledge. The latter evaluates the success of the design. Running multiple iterations of the build-evaluate loop will produce more refined designs.

The main artefact designed in this thesis is the near real-time GIS, but to reach the goal a second artefact in the form of component selection process must be designed. To accomplish the design goals, the knowledge base is searched for answers to the following questions: *How to design a real-time GIS using OGC standards? How to select components for a component based software system?*

Utility of the design will be discussed from systems development point of view. The scope of this thesis will be limited to a single iteration of the build-evaluate loop.

1.4 Structure of the thesis

Chapters two and three will cover the theory part of this thesis. The first one will discuss the fundamentals of GIS and review OGC standards relevant to the topic of this thesis.
The later chapter will review the work found in the literature regarding software component selection. Both commercial and open source viewpoints are covered.

The practical work done in this thesis is described in chapter four. First part of the chapter will describe the component selection process used to pick the mapping component. The later part will cover the system design for the near real-time GIS.

Chapter five will compile the results of the thesis and further discuss them focusing on success of the work and future possibilities. Finally, chapter six will summarize the whole thesis.

2 GEOGRAPHIC INFORMATION SYSTEMS

Geographic Information Systems (GIS) are systems specialised in storing information with geographic reference. In this chapter general concepts of geographic information are explored. Later some of the standardization work of the OGC is reviewed.

2.1 Geographic information

Geographic information can be typical map features, for example roads and rivers, but also for example sensor and population data. With the growing number of smart connected devices, such as smart phones, smart infrastructure and various sensors, the variety and quantity of geographic information is rising.

There are two main types of information in the field of GIS. First type is grid or cellular data, where a grid consisting of cells of uniform size covers a geographic area. Each cell has a set of attributes describing the area it covers. Examples of data that could be expressed as a grid could be soil type or rainfall. Grid data can be wasteful in terms of data storage if the data is not very heterogenic, thou dynamic cell size can alleviate this. (Nagy and Wagle 1979)

The second type is object data, where objects have their unique coordinates and geometries. The types of geometries used in GIS are points, lines and regions. In addition to geometries, the objects have a set of attributes describing the area they cover. Examples of data that could be expressed as objects would be road networks and water systems. Object data is fairly efficient to store and relationships between objects are easy to track. (Nagy and Wagle 1979)

Typical GIS implementation is a database driven system allowing storing, modifying and querying information. The data objects in GIS have an extent, the area they cover, and can have spatial relationships to each other. For example, one object may be intersecting the extent of another object. These relationships can be complex and impractical to map to conventional database indexing scheme. Often a special spatial database that can do spatial

indexing is need. (Ooi 1990, pp.1-8)

2.2 Open Geospatial Consortium

Open Geospatial Consortium (OGC) is a not-for-profit trade association for promoting interoperability through standardization in GIS. It has over 500 members ranging from private companies to government agencies and universities. Founded in 1994, the consortium was first known as Open GIS Consortium, Inc., but changed its name to the current form in 2004. (OGC)

OGC has published dozens of standards. Some of them have later become International Organization for Standardization (ISO) standard. For example, the Web Feature Service (WFS) is ISO 19142:2010 standard. In the following chapters, some of the OGC standards that are most relevant for the work done in this thesis are reviewed.

2.2.1 Geography Markup Language

Geography Markup Language (GML) is a XML grammar for describing generic geographic datasets containing points, lines and polygons. Due to its generic nature, more specific application schemas are needed to enable users to refer to concrete geometries like roads and rivers. Description of a feature in GML can be seen in example 1. (OGC 2012)

Real world objects and phenomena are abstracted into features in GML. These features combine geometry to a set of attributes describing the object. The feature types, essentially the set of attributes a feature should have, are described in the application schema. GML is widely utilized in OGC standard services, for example in the WFS. (OGC 2012)

```
<gml:featureMember>
        <tiger:tiger_roads fid="tiger_roads.1">
                <tiger:the_geom>
                        <gml:MultiLineString
                                srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                        <gml:lineStringMember>
```

```
                    <gml:LineString>
                    <gml:coordinates decimal="." cs="," ts=" ">
                            -73.999559,40.73158 -73.999079,40.732188
                    </gml:coordinates>
                    </gml:LineString>
                    </gml:lineStringMember>
                    </gml:MultiLineString>
                </tiger:the_geom>
                <tiger:CFCC>A41</tiger:CFCC>
                <tiger:NAME>Washington Sq W</tiger:NAME>
        </tiger:tiger_roads>
</gml:featureMember>
```

Example 1 GML representation of a feature named “tiger:tiger_roads”. The feature consists of geometry field called “the_geom” and properties called “CFCC” and “NAME”.

2.2.2 Common Query Language

Common Query Language (CGL, sometimes OGC CommonQL) is a simple query language used in multiple OGC standard services. CQL was introduced in the Catalogue service standard to be its primary query language. Syntactically the CQL is designed to be similar to the Structured Query Language (SQL) WHERE-clause. CQL is also designed to be extensible. New predicates, operations and datatypes can be added to it. CQL in URL form can be seen in example 2. (OGC 2016)

```
CQL_FILTER=NAME LIKE '%Ave%'
```

Example 2 CQL filter in URL query string parameter form. This filter would select all the Avenues from a set of roads. For readability the URL character escapes have been removed.

2.2.3 Web Map Service and Web Map Tile Service

The Web Map Service (WMS) is a Hypertext Transfer Protocol (HTTP) based service interface for fetching map images. These images are renditions of information stored in the GIS. WMS is queried with a set of parameters defining for example the geographic bounding box of the are to render and the dimensions of the output image and projection it

should be rendered in. A simple WMS query can be seen in example 3. Also a CQL filter can be used to limit the contents of the map. Alongside the ability to query maps (GetMap operation), WMS has also function for querying the capabilities of the service (GetCapabilities operation). The capabilities describe for example the available map layers and supported projections. (OGC 2006)

```
http://localhost:8080/geoserver/wms?
        SERVICE=WMS &
        VERSION=1.1.1 &
        REQUEST=GetMap &
        FORMAT=image/png &
        LAYERS=tiger:tiger_roads &
        SRS=EPSG:4326 &
        WIDTH=476 &
        HEIGHT=768 &
        BBOX=-74.03686523,40.6830596923,-73.9551544189,40.8148956298
```

Example 3 Simple WMS query URL. For readability the URL character escapes have been removed.

The Web Map Tile Service (WMTS) can be seen as an extension of the WMS. In WMS the bounding boxes of the queries can be arbitrary as well can the zoom level of the map image, which is conducted from the relation of the bounding box and the output image size. This free form nature of WMS makes it practically impossible to cache the images for reuse on the server providing the service. Since map rendering operations are fairly expensive in terms of computing resources, there are great performance benefits in caching.

WMTS forces the map queries to be made in a grid set defined by the service. Renditions of the map in each tile of the grid can be cached and reused when the same area is queried again (OGC 2010). This provides better performance, but also leads to situation where the map images served are old. In case of static maps that do not change over time, age of the tiles is not a problem. For more lively geographic information, for example weather radar data, WMS is the more suitable service.

2.2.4 Web Feature Service

The Web Feature Service (WFS) provides access to geographic information in feature level. As explained in the GML chapter, features can be map objects like roads and rivers and consist of a geometry and a set of properties. The WFS allows accessing these features and querying them using the CGL. One could for example query a set of features representing operational drawings and limit the set not only by geographic area, but also by type, colour, timestamp, creator or any other attribute a feature may have. A simple WFS query can be seen in example 4. (OGC 2010b)

```
http://localhost:8080/geoserver/wfs?
        SERVICE=WFS &
        VERSION=2.0.0 &
        REQUEST=GetFeature &
        TYPENAME=tiger:tiger_roads &
        SRS=EPSG:4326 &
        CQL_FILTER=NAME LIKE '%Ave%'
```

Example 4 Simple WFS query for feature type representing roads with the CQL filter from example 3. For readability the URL character escapes have been removed.

The WFS also has an operation for discovering the capabilities of the service (GetCapabilities operation) similar to the WMS. Additionally, there is an operation for discovering feature types the service can provide (DescribeFeatureType operation). The feature type describes the set of attributes used for the type. For example, type representing a road could have name and speed limit attributes. (OGC 2010b)

The WFS can also support transaction, in that case being abbreviated to WFS-T. Transactions allow common storage operations, create, read, update and delete, to be used on features. There is also a locking operation to allow exclusive access to a feature being modified. (OGC 2010b)

2.2.5 Sensor Observation Service

The Sensor Observation Service (SOS) is a standard interface for providing measurement information from sensors and also sensor description. The SOS is part of the OGC's Sensor Web Enablement framework providing set of standards for sensors. The goal of the SOS is to provide discoverability and interoperability between a variety of sensor systems from individual static sensors to mobile measurement platforms. (OGC 2012b)

Core implementation of the SOS consists of only three operations. Firstly, like many other OGC services, there is an operation for querying service capabilities (GetCapabilities operation). Secondly, there is an operation for querying sensor descriptions (DescribeSensor operation). Finally, there is an operation for querying measurements (GetObservation operation) that support spatial, temporal and thematic filtering. The SOS can also support transactions enabling writing sensor data to a storage via the standard interface. (OGC 2012b)

Similarities can be seen between the SOS and the WFS. To summarise the differences, the WFS is a freeform way to describe real-world phenomena and always requires application specific schema to define possible feature types. The SOS on the other hand is specialised in describing sensors and measurements and will not require additional definitions.

Since WFS could also be used to serve spatially bound measurement information, the question of which standard to select arises. The SOS would at first appear as obvious selection, but more attention should be payed to this question. Bermudez et. al. (2009) compared the two standards in the context of ocean observations. They found SOS to be more suitable for the case. The main factor to this outcome was that SOS has all the definitions needed and with it there would be no need to maintain schemas. However, Bermudez et. al. (2009) did not consider the costs of implementing the system and did not look into availability of existing software implementing the standards.

2.3 Discussion

GIS is essentially just an information system that deals with data that has a spatial component. This component brings degree of complexity to storage formats and data structures. Spatially indexed database is key to achieving high performance on spatial query operations.

The OGC standards provides comprehensive set of tools for distributing spatial data. Both service interfaces and transfer data formats are covered. There is some overlap between the formats. For example, both the WMS and the WMST serve raster data, difference being that the WMS allows arbitrary bounding boxes and WMST requires the use of a predefined grid set. From technical perspective, WMS is more suitable for dynamic maps and WMST for serving cached static maps efficiently.

Also the WFS and the SOS standards overlap. Both can be used to serve data points with spatial properties. The WFS is more general purpose whereas the SOS is specialised in serving measurement data. Picking suitable standard can be non-trivial task and requires considering different aspects from technical details to availability of existing software and services.

3 COMPONENT SELECTION IN SOFTWARE SYSTEM DEVELOPMENT

Modern software systems have grown to be relatively complex. To manage the complexity, systems are typically divided into separate components that put together form the complex system. Many of the components needed are not specific to the developed system. For example, user authentication components are needed in many systems and work more or less the same way. This opens up opportunity for re-using components. This approach where systems are put together utilising existing components is called Component Based Software Engineering (CBSE).

Component selection is a crucial task not only in software, but everywhere where products are developed taking advantage of existing, possibly third party, components. These components can become integral part of the end product and the deep dependencies can make them practically impossible to change. Thus, some time and effort should be spend comparing and choosing the most fitting component with the entire lifetime of the product in mind.

The system developed in this thesis heavily relies on existing third party components. Many of the components are selected based on experience from previous projects. They are known to fit the task and there already is experience in using them. An exception to this is the web map component needed for the front-end. Since it will be practically inseparable part of the front-end application, attention is paid on the selection process that will be used to find the most fitting component for the task.

In this chapter, some software component selection processes used in the context of CBSE are discussed. Both the case of Commercial Off-The Shelf (COTS) and Open Source Software (OSS) are considered.

3.1 Commercial Off-The Shelf component selection

There are a multitude of various COTS components selection processes proposed in the literature. CBSE often tangles with developing large and complex systems, and in that case

use of complex and time consuming selection processes can be justified. In this chapter, the processes more fitting to the scope of the work done in this thesis are discussed. For example, COTS-Aware Requirements Engineering (CARE) proposed by Chung and Cooper (2004) was left out of consideration due to its complexity.

3.1.1 Off-The Shelf Option

The component selection could be, and to some extent still is, done in ad hoc manner. Since proper selection process takes time and uses the limited development resources, it may in some of the more straightforward cases be the sound solution. However, for more crucial components, well defined process provides benefits. As Kontio (1995) states, a well-defined process enables process development. Earlier experiences can be used to improve the process over iterations resulting to evermore suitable process for the organization.

Kontio (1995) defines Off-The Shelf Option (OTSO) process to be well-defined, systematic process that covers the whole component selection. As can be seen in the figure 1, OTSO breaks down to six phases: Search, Screening, Evaluation, Analysis, Deployment and Assessment. In search phase, possible candidates are quickly listed, and the amount of candidates grows rapidly. In the following screening phase, the number of candidates to enter further more in-depth evaluation is cut down. Here the available resources should be taken into consideration to keep the number feasible.

In evaluation phase, selected components are evaluated using defined criteria and the work is documented. In the following analysis phase, the data collected is used to decide which component will be deployed and used in the software development. To enable future process improvement, the last phase, assessment, considers the success of the selection.

What remains as true challenge and crucial factor in the success of the selection process, is the formation of the criteria used in the evaluation phase. Kontio (1995) discusses the risk of over valuing concrete technical characteristics over the fuzzier ones. Especially when there is little time for decision making, there is a risk of relaying mostly on the easily available concrete characteristics.

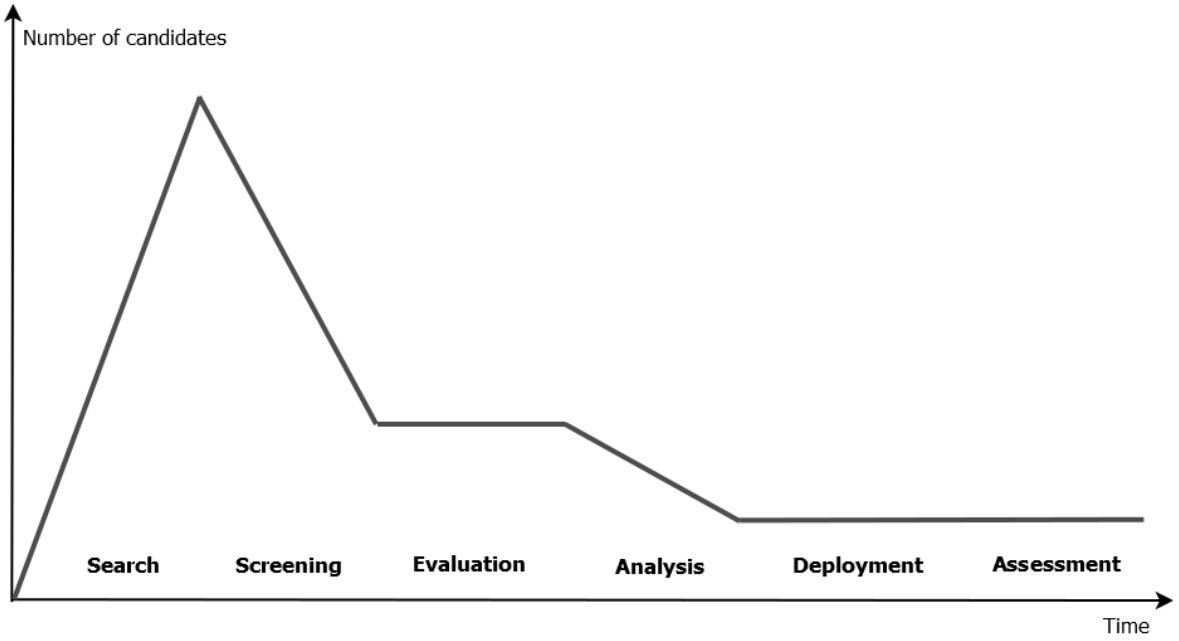

Figure 1 The OTSO process

3.1.2 Procurement-Oriented Requirements Engineering

Maiden and Ncube (1998) introduced Procurement-Oriented Requirements Engineering (PORE) based on their experience selection COTS components to a complex system with large amount of requirements. Key point of PORE is the integration of requirements gathering phase of the software development process and COTS selection process. As can be seen in figure 2, these processes are run simultaneously and the growing number of requirements are used to decrease the number of possible components.

PORE is an iterative process that can be run till there are only components that sufficiently fill the requirement remaining in the pool of candidates. Maiden and Ncube (1998) divide into three phases, leaving option for additional phases in future. In the first one, components are evaluated based on supplier's product information. During this phase, but also in the following two phase, requirements gathering should be focused to gather requirements suitable for the evaluation. In case of the first phase, requirements should be suitable for evaluating product information.

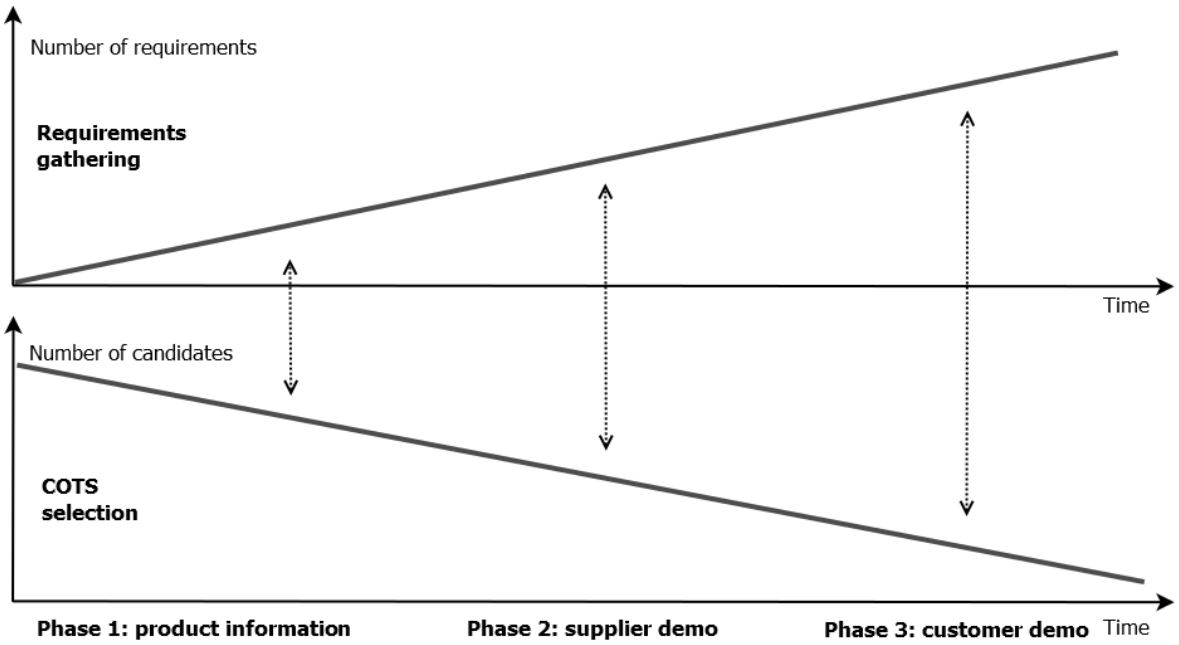

Figure 2 PORE and requirements gathering as parralel processes

In the second phase, components are evaluated based on supplier-led demonstrations. The third phase has the most direct customer involvement. Components are evaluated based on customer-led product demonstrations.

Compared to Kontio's (1995) OTSO, in the PORE process the number of candidates is reduced gradually throughout the whole process. Also the customer can be more involved in the selection of the components.

On the topic of forming the criteria, effectively the customer requirements, Maiden and Ncube (1998) discuss the problems related to weights. They found simple percentage scale to lead to inconsistency in scaling different requirements. They suggest using more sophisticated methods, particularly analytic hierarchy process. Thou applying such complex method over large set of requirements would be too time consuming. Proposed solution to this is to use different decision-making techniques for each requirement. The technique should be selected based on the characteristics of the requirement favouring the most cost-efficient one.

3.1.3 PECA

Comella-Dorda et. al. (2002) propose the PECA process in response to problems identified in the COTS selection. PECA is intended to be flexible enough to be customized to fit the needs of different organizations and development processes. In PECA, the decision of which component to choose is not considered to be part of the process. End product of the process will be recommendations to decision makers.

The name, PECA, is formed from its phases: Planning the evaluation, Establishing the criteria, Collection the data and Analysing the data. As can be seen from the figure 3, the phases of PECA are not necessarily performed in fixed sequential order. There are multiple points where one could return to earlier phase if more work on it is needed.

PECA starts with planning the evaluation. In this phase, the evaluation team is formed. Comella-Dorda et. al. (2002) emphasize the need of wide team with varying backgrounds. A single engineer could not possibly possess the skillset needed for proper evaluation. Also the stakeholders should be identified. Some of them may be also in the stakeholders of the whole software development project, but COTS selection can have its unique set of stakeholders based on the component that is been selected.

The planning also determines the characteristics of the evaluation, particularly the depth of evaluation. The depth depends on how critical the component is to the system. If there are too many candidates to choose from, more cost efficient filters may be used to cut the number down before in-depth evaluation. The filter could be seen as small scale implementation of the PECA process consisting of all of its phases.

Next phase if the forming of the criteria. Comella-Dorda et. al. (2002) point out that the requirements of the whole system rarely address specific needs of the component selection. Thus, evaluation requirements, a subset of the system requirements concerning the component, should be identified. The evaluation requirements can be extended with additional requirements that are needed for the selection, but are not present in the system requirements. The requirements are formed into the evaluation criteria. Comella-Dorda et. al. (2002) define the criteria to consist of two parts: capability statement and quantification

method. The first is clear and measurable statement of capability that should be satisfied. The later defines the way the level of satisfaction should be assessed and transformed into a value.

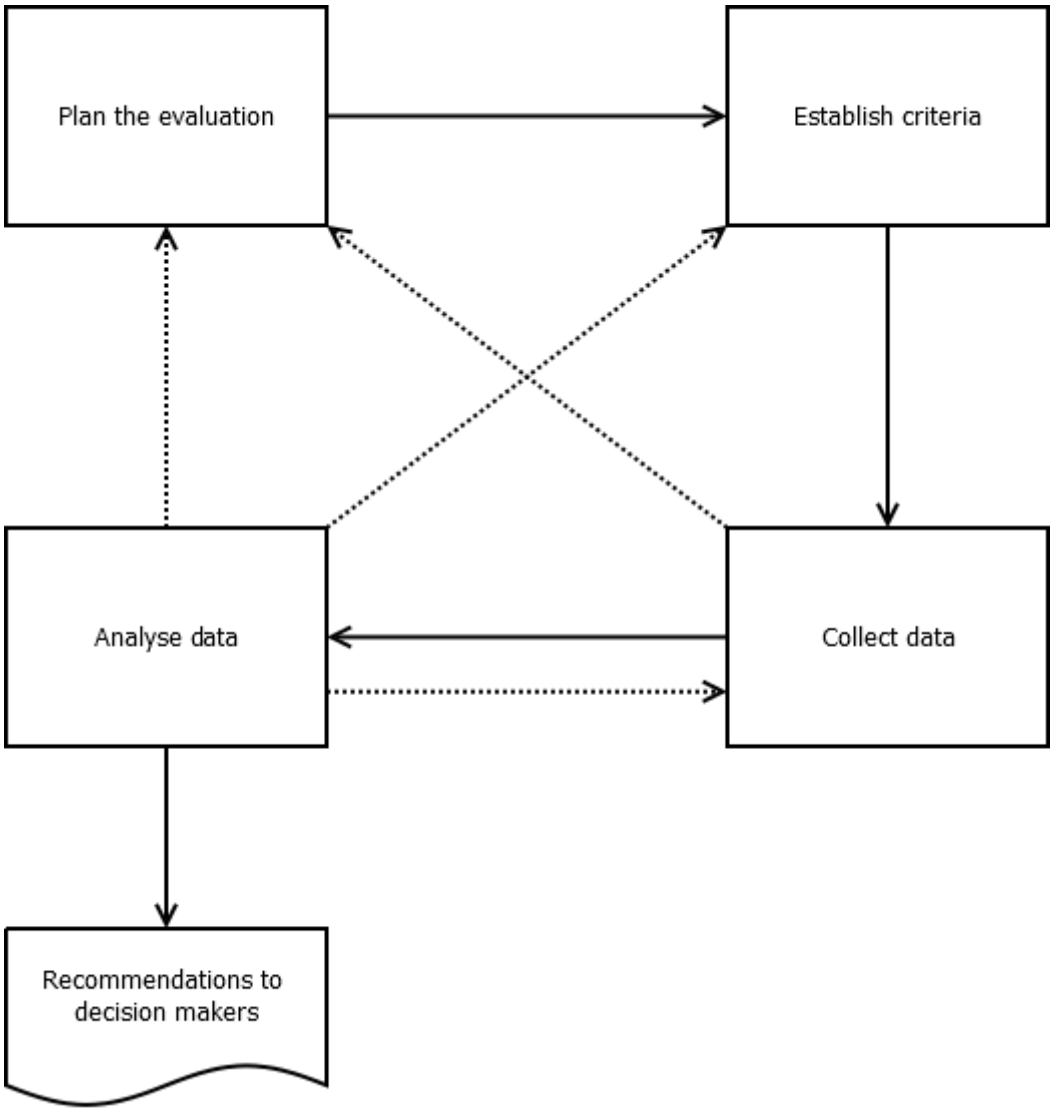

Figure 3 The PECA process

The data collection phase is fairly straightforward execution of the evaluation plans to define how the candidates compare to each other. However, as Comella-Dorda et. al. (2002) states, COTS software is full of surprises, and they come up when knowledge on the candidates increases. This is why PECA process has fall back option to return to evaluation planning phase.

In the final phase the data is analysed. For each candidate, three documents are formed. Firstly, the product dossier that collects all the knowledge on the product. Secondly, the evaluation record that in-detail describes the evaluation conducted for this candidate. Finally, the summary and recommendations that concludes the findings for this candidate. The data collection phase has multiple fall back options in case it is revealed that the work on previous phases was not sufficient enough.

3.1.4 The Social-Technical Approach to COTS Evaluation

Kunda and Brooks (1999) identify the lack of COTS selection processes that take soft issues, such as economic and political factors, into account. To address this, they propose the Social-Technical Approach to COTS Evaluation (STACE) framework. According to them, the process is greatly influenced by the OTSO and the PORE processes. Due to this close relationship to work reviewed earlier in this paper, this chapter will only review the most interesting aspects of the framework.

Along with the social-technical approach, STACE is also designed to support evaluating the technology of the COTS products along the products themselves. Kunda and Brooks (1999) also emphasize the need of defined process instead of ad hoc one to enable learning from past projects.

As can be seen in figure 4, the STACE framework consists of four interrelated processes: requirements elicitation, social-technical criteria, alternatives identification and evaluation. These processes can be run simultaneously and provide feedback to each other. Requirements elicitation gathers high-level customer and systems requirements. The requirements are then processed into evaluation criteria in the social-technical criteria process. When forming the criteria, feedback from alternative identification and evaluation process are taken into account. Also experience from past projects is considered an information source for the criteria formation.

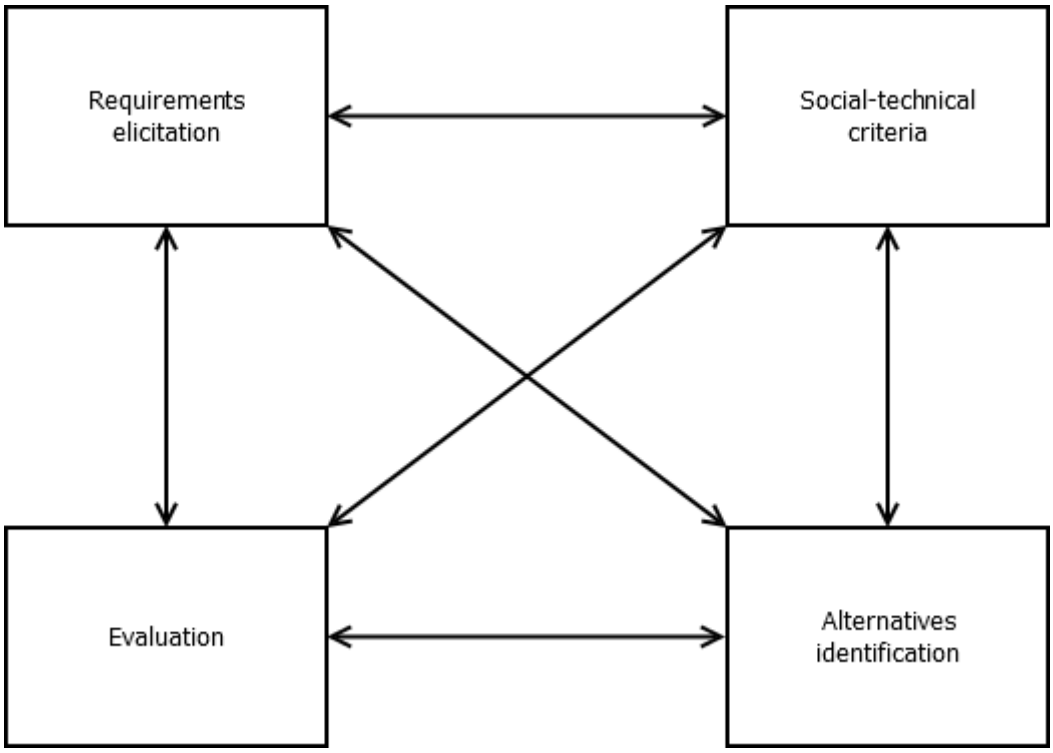

Figure 4 Interrelated processes of the STACE framework

According to Kunda and Brooks (1999) the social-technical criteria should include technology factors, functionality characteristics, product quality characteristics and social-economic factors. Out of those, the technology factors focus on the underlying technology of components. There may be customer requirements on which technology the software should use and the technology of the component can widely effect the system architecture. Thus, the technology should be separated from the product characteristics.

The socio-economic factors are the non-technical factors that according to Kunda and Brooks (1999) often get neglected. These factors include for example cost, business issues, vendor performance, vendor reliability and organisational issues. Out of those, the vendor performance and reliability include aspects such as vendor infrastructure, reputation and references. The organisational issues include management support, internal politics, staff issues and attitudes.

3.1.5 Summary

The literature has plenty of proposed processes for commercial component selection. Many

of them are aimed at developing large information systems and require a lot of work to be fully implemented.

The processes reviewed above share similar phases. All of them have some sort of criteria formation, candidate selection and evaluation phases. The differences lay in the details, and all of them present a different point of view and bring something new to the table.

OTSO, as an early work in the field, is a fairly straight forward process highlighting the importance of having a defined and systematic selection process. PORE places the component selection parallel to the requirements gathering and exploits the ability to have input from one process to the other. PECA presents itself as flexible process that can be scaled to match the implementing organization. STACE points out the need to take the non-technical factors in to account when components are evaluated.

3.2 Open Source Software component selection

Open Source Software (OSS) has lately become a viable alternative to COTS also on the software component market. They provide lower cost more open alternative usually allowing modification of the components. However, as Di Giacomo (2005) states, licensing varies and should be payed attention. In some cases, the components can be freely modified and commercially redistributed, but in some cases the license can require publication of the modified source code.

To some extent, the OSS components can be evaluated using the existing the COTS selection tools. However, this does not work for all aspects. For example, the supplier driven demos used in the PORE process proposed by Maiden and Ncube (1998) would not be available. Also, most of the socio-economic factors in the STACE framework proposed by Kunda and Brooks (1999) could not be directly applied to OSS components. Di Giacomo (2005) states that lack of evaluation tools limits the opportunities of the OSS to compete with the COTS.

According to Di Giacomo (2005), there are two ways of using OSS components: black-box and white-box. The first one enables one to use to component similarly to a COTS

component. The component is used as is and information on it is gained through documentation and other material provided by the development community. However, the ability to view the source code could be used for testing and especially bug fixing.

In the later, the source code of the component is also reviewed and possibly modified while integrating the component to the system. This provides greater flexibility and ownership of the component, making it integral part of the system, but also complicates the issue. First of all, there is the burden of reviewing the source code and familiarising with it to the point where the developer can extend it. This also tends to lead to greater dependency to the component when the integration is not done through minimal Application Programming Interface (API). This yields the need for analysing the maturity of the OSS project to ensure one is investing to a long lasting project.

There are numerous OSS maturity models proposed in the literature. Stol and Ali Babar (2010) listed 20 methods. Many of them evaluate aspects of the software, but do not provide full component selection process the way the previously reviewed COTS selection methods did. Also they are mostly directed on selecting full software products instead of components. However, they provide useful insights on OSS selection. In the following chapters, some of the OSS maturity models are reviewed.

3.2.1 The OpenSource Maturity Model

The OpenSource Maturity Model (OMM) introduced by Petrinja et. al. (2009) is designed to be simple maturity model deliberately similar to the commonly used Capability Maturity Model Integration (CMMI). The alignment with CMMI makes the model easily understood by the industry. By being significantly simpler and less demanding on the infrastructure than the CMMI, the model should be more appealing to the OSS community. The OMM focuses on the process quality of the OSS which is believed to eventually lead to higher software quality.

The OMM divides the maturity into three levels: basic, intermediate and advanced. Petrinja et. al. (2009) define 12 trustworthy elements that represent a characteristics of the development process. These elements are assigned to the levels and to pass a level all the

elements on it must be met. In addition to the elements, the level may also require passing requirements of a CMMI level. The element and their assignment to levels and CMMI levels to pass can be seen on table 1.

Table 1 OMM levels, trustworthy elements and CMMI levels

Level	**Trustworthy element**	**CMMI level**
Basic	Product documentation	
	Use of established and widespread standards	
	Quality of test plan	
	Contribution to the OSS by software companies	
	Licence	
	Technical environment	
	Number of commits and bug reports	
	Maintainability and stability	
Intermediate	Popularity of the software product	Level 2
	Availability and use of product road map	
	Relationship between developers	
	Results of assessment of the product by 3rd party companies	
Advanced		Level 3

3.2.2 The Open Business Quality Rating

The Open Business Quality Rating (OpenBQR) evaluation framework proposed by Taibi et. al. (2007) addresses identified issues on the field OSS selection. These issues include focus on OSS specific aspects, inadequate dealing with both internal and external quality and the lack of assessing support over time.

Taibi et. al. (2007) divide OpenBQR into three phases: quick assessment filter, data collection & processing and data translation. The first phase forms indicators and weights of importance for each indicator. There are five areas that should be considered when forming the indicators: scope and target use, external qualities, internal qualities, product support over time and existence of required functionalities.

In the data collection and processing phase, indicators that have zero or very low weight are removed. The characteristics described by the remaining indicators are measured and the results for each area are summed with the weights. The end result of the evaluation is a set of five numbers describing the performance of the software in the five areas listed for the first phase. In the last phase, data translation, results of the evaluation and results of evaluation of other similar software are visualized in meaningful way.

The process displays some similarities to earlier reviewed COTS selection processes. Its indicators are effectively the selection criteria of the COTS processes and final phase compares the results to other similar software.

3.2.3 Summary

The open source evaluation processes seem to focus more on evaluation the organization or community behind the software. This kind of indirect quality assessment could be combined with a COTS process focusing on the functional and technical aspects of the components. If the organization looking for components is willing to get involved in the development of the component, evaluating the organization grows more important.

3.3 Discussion

Both commercial and open source component selection have been discussed with reviews of selection and evaluation processes found in the literature. The commercial side has plenty of processes addressing the needs of CBSE industry. Present in all the reviewed processes is the emphasis in using defined processes instead of an ad hoc one. The reviewed processes have different approaches and emphasis on the subject, but they all fall into the same generalized form. They all form criteria from requirements, gather information and analyse the results.

The open source side seems to lack work on component selection. Surprisingly there is very little writing considering OSS as just another form of COTS, different only by the availability of the source code. The diversity seems to make side by side comparison of COTS and OSS components difficult.

If implemented precisely as described, the reviewed processes would be fairly laborious. They seem to be aimed at larger system development projects. For the work done in this thesis the reviewed processes as such would be too time consuming. Thus a custom process borrowing elements from the reviewed ones will be developed.

4 SYSTEM DESIGN AND IMPLEMENTATION

The concreate goal of this work is to design and implement a web based near real-time GIS application for displaying field measurement data. The application is intended for both real-time observation view and browsing historic data. The measurement data for the application will be radiation measurements from the Linssi database. Key features of the application are:

- Browser based user interface
- Raster background maps (WMS/WMTS)
- Ability to create and modify drawings on map, stored on server side
- Displaying measurement points from the Linssi on map
- Near real-time feed of live measurements from the Linssi
- Querying historic measurements from the Linssi
- Showing details of a measurement including spectrum
- User authentication and administration

The chapter 4.1 will introduce the component selection process used to decide the front-end map component for the application. Also the execution of the process and experience from it will be discussed. The chapter 4.2 discuss the system design and explain its components in detail.

4.1 Front-end Component Selection

Selection of the front-end map component was considered crucial for the system. The client application would be map centric with most of the user interaction happening on the map or on its overlays. The capabilities and extensibility of the map component would have heavy impact on the capabilities of the end product and the development costs. It was also acknowledged based on previous experience that developing a map component with fluent user experience and wide set of features from scratch would be a sizable task likely to fall short.

Based on the above considerations, it was decided to devote relatively large part of the development time to selecting a suitable map component. Based on the previous work on component selection reviewed in the chapter 2, a custom component selection process was developed. The custom process is heavily influenced by the works reviewed.

The decision to go for custom process instead of implementing one of the existing ones was due to the relative complexity of the reviewed processes. Particularly the COTS selection processes seemed to be aimed for a fairly large information system development scenario. The work done in this thesis is far from it. The whole system is expected to be implemented by a single developer over a few months.

4.1.1 Selection Process

As stated by Kontio (1995), defining process used for component selection is important. An ad hoc process would work as well for this single case and could yield as good results as the defined one, but for future projects the defined one opens the possibility of process development. It is easier to evaluate the success of the selection process and suggest improvements for future passes when the process is defined in a way that is repeatable.

A defined process also eases the communication within organisation and possible third parties. The process and its results can be reviewed and possible weaknesses addressed by the implementers of the selection, but also by interested outsiders. This helps to build up the credibility of the results, whereas in ad hoc process the credibility would depend mostly on trustworthiness of the implementers.

For the reasons stated above, the component selection process to be used, its phases and produced artefacts, are described below. Overview of the process can be seen in figure 5. Similar to the PORE process, the component selection runs parallel with common software development activities. Simultaneous activities can provide feedback to each other, particularly the requirements gathering and the criteria formation. Similarly, to many of the reviewed selection processes, there is always the option to backtrack to any of the previous phases, if it becomes apparent that more work should have been done there.

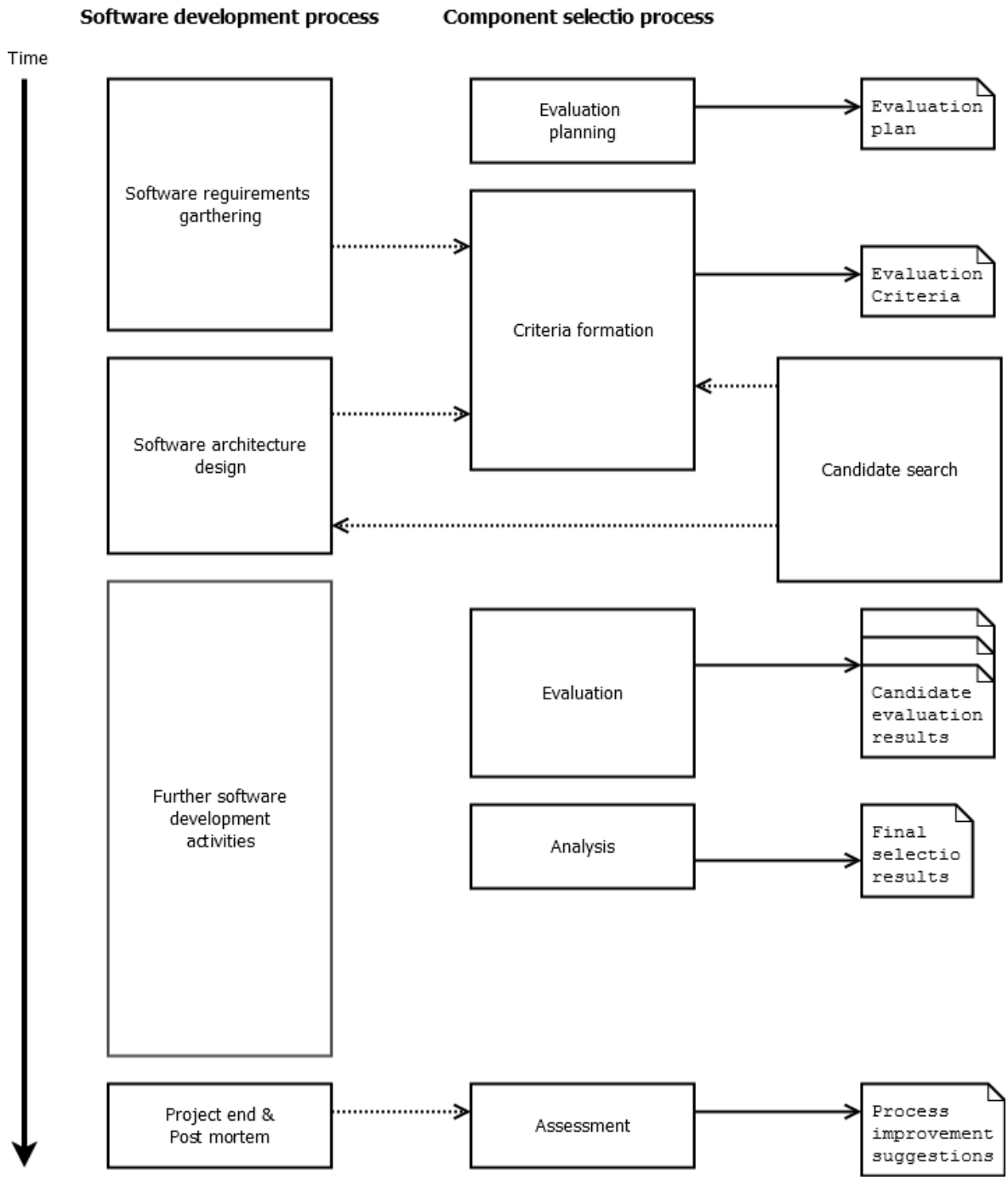

Figure 5 Overview of the selection process parallel to the software development process. The dashed lines represent additional feedback from parallel phase.

The selection process consists of the following five phases:

1. Evaluation planning

The process begins by planning the evaluation. A plan covering participants, their

roles, scoring system and templates to be used in the evaluation.

2. **Criteria formation**

   Evaluation criteria should be formed and documented to templates from evaluation plan. The criteria are a set of capability statements derived from software requirements and constrains of the software architecture. The criteria should also have weight factors for each statement. This phase starts when initial requirements are gathered and will receive feedback from requirements gathering and architecture design phases proceed.

3. **Candidate search**

   The candidate search phase will start when initial criteria has been formed. With knowledge from criteria formation, it is expected that the search phase alone can produce short enough list of suitable candidates for evaluation. There will not be separate refinement phase similar to OTSO's screening.

   The search for components also acts as familiarisation to the subject matter, in case of this thesis to web map components. Thus, this phase will provide feedback to the criteria formation providing answer to the questions of what are important properties for these kind of components and what kind of thing could set the candidates apart?

   The findings on available components may also effect the software architecture design. This is to avoid blindly pushing forward entirely unnecessary or somehow mitigatable limitations that would rule out otherwise prominent candidates.

4. **Evaluation**

   The evaluation will go through the criteria evaluating performance of each candidate in regard to the capability statements. Various methods for acquiring the performance reading can be used. Sometimes documentation or existing demo software is enough to base assessment on, but occasionally writing a short demo software may be needed. Having software development activates running in

parallel can help with the demos. For example, it is much easier to test some front-end components when there already is the first iteration of the back-end providing realistic test data.

When selecting the method, weight factor of the assessed capability statement should be kept in mind. For less important parts method whit lesser workload could be used and time should be devoted to ensure proper assessment of key features. The used method should be documented along with the results.

The candidates are evaluated in relation to each other. This greatly simplifies the quantitation of the results. In case of an organization that repeatedly develops similar software projects, there would be an advantage in using universal and independent scale to make evaluation the results reusable. However, the work done in this thesis is a unique case and the interest is only in how the candidates chosen for this evaluation compare to each other.

5. Analysis

Analysis is fairly straight forward act of gathering scores from different evaluations and summing up the final scores with weight factors applied. The highest scoring candidate wins.

6. Assessment

For the sake of process improvement, the success of the component selection process should be evaluated and possible needs for improvement identified. Good time for this would be after the software development process has ended and the suitability of the selected component can be assessed.

4.1.2 Execution of the Process

Evaluation for web map component was planned, based on the process described above. It was decided to use simple three step scoring scheme. The numeric steps were mapped to fuzzy terms:

1. Does not fill the criteria
2. Partially fills the criteria or there are issues, for example with quality.
3. Fully fills the criteria

If a capability statement is evaluated with a test that provides concrete quantitative result, for example speed or size, the score should be based on how the candidate compares to other candidates.

4.1.2.1 Criteria

Criteria for the map component was extracted during the formation of software requirements for the whole system. The criteria were formatted as capability statements, and these statements were given weight factors based on their importance. The final statements are listed in table 2.

Table 2 Final list of capability statements and weights

#	Name	Weight
1	Web Map Service support for background maps	4
2	Web Map Tile Service support for background maps	4
3	Web Feature Service support	4
4	Web Feature Service Transaction support	3
5	Customizing feature style	3
6	Selecting features	3
7	Dynamic modification of features	3
8	Basic drawing tools (Points, lines ad polygons)	2
9	Basic measurement tools	2
10	Usability on touchscreen devices	2
11	Quality of API documentation	2
12	Availability of additional tutorials, demos and Q&A material	2
13	Interface for plugin development	1
14	Availability of third party plugins	1

15	Performance by initial load time	1
16	Performance by large amounts of WFS features	3
17	Online map services (Open street maps)	3

The first three statements address the OGC protocol support essential for the system design. Thus they are given the highest weight. From four to seven, the statements address important abilities for displaying and interacting with the measurement data on the map surface. Statements eight, nine and ten deal with additional features that could also be custom implemented, but ideally would come with the component. Thus the lower weight factor.

Statements eleven and twelve consider the documentation and learning material. These can be important for successful development, but since the two statements consider close topics and also evaluation of these topics can be fuzzy, fairly low weight factor is used.

Statements thirteen and fourteen deal with the extensibility of the components. Low weight factor is used, because the intent is to use the component as black box and not extend or modify its behaviour. Still, some modification may be necessary and it is better to have well defined interface that enables modifications without touching the components source code.

Performance is considered on statements fifteen and sixteen. Much higher weight is given to the ability to display large datasets. Long initial load times, typically caused by large code masses being downloaded and evaluated by the JavaScript interpreter, are not considered to be as significant issue.

The last statement considering alternative background map sources arouse during candidate search phase as potential differentiator and was added to the criteria.

4.1.2.2 Candidates

During candidate search, three potential map components were selected for further

evaluation. Two of them were found through internet searches. The third one had been introduced by a sales representative of the component provider some time before this project. A brief overview of the candidates can be seen in table 3.

Table 3 Overview of the candidates

Name	Overview
Open Layers 3	Open source component developed by the Open Source Geospatial Foundation. Provides wide set of geospatial capabilities.
Leaflet.js	Lightweight open source component alternative to the Open Layers. Advanced capabilities are built as plug-ins on top of the simplistic core component.
Luciad RIA	Commercial off the shelf component that is the web front-end to a full stack solution from the component provider. Also capable of using open OGC standards and thus can be used on a custom stack.

4.1.2.3 Evaluation and results

Performance of each candidate was tested in regards of the capability statements. The testing was done one statement at a time so that all candidates were evaluated against it. This helped with the scoring of the results when the candidates can be evaluated relative to each other.

The statements regarding OGC standards support, the ones with highest weight factor, were tested by writing a short demo programs. The evaluation phase was deliberately timed so that the back-end would be finished to a point where it can provide data for the test programs.

For the remaining statements, writing short demo programs was occasionally needed, but often the component suppliers had written tutorial and demo programs suitable for the test. This significantly reduced the work time needed to conduct the evaluation.

Finally, the scores were gathered together and weights were applied. The final results of the component evaluation can be seen in table 4. As can be seen from the bottom row of

the table, highest score after weights were given to the Open Layers 3. This component will be used to display the map in the system developed in this thesis.

Table 4 Final results of the component selection

#	RIA	Leaflet	OL3	Weight
1	2	2	2	4
2	2	1	2	4
3	2	2	2	4
4	0	1	2	3
5	2	2	2	3
6	2	2	2	3
7	2	2	2	3
8	2	2	2	2
9	2	2	2	2
10	2	2	2	2
11	2	2	2	2
12	1	2	2	2
13	0	1	0	1
14	0	2	0	1
15	2	2	2	1
16	2	1	2	3
17	1	2	2	3
Total	26	30	30	
Weighted	**71**	**69**	**76**	

4.2 System design

The overview of the system architecture can be seen in figure 6. The arrows in the diagram represent the flow of data. In this chapter, the key components of the system are explained in detail.

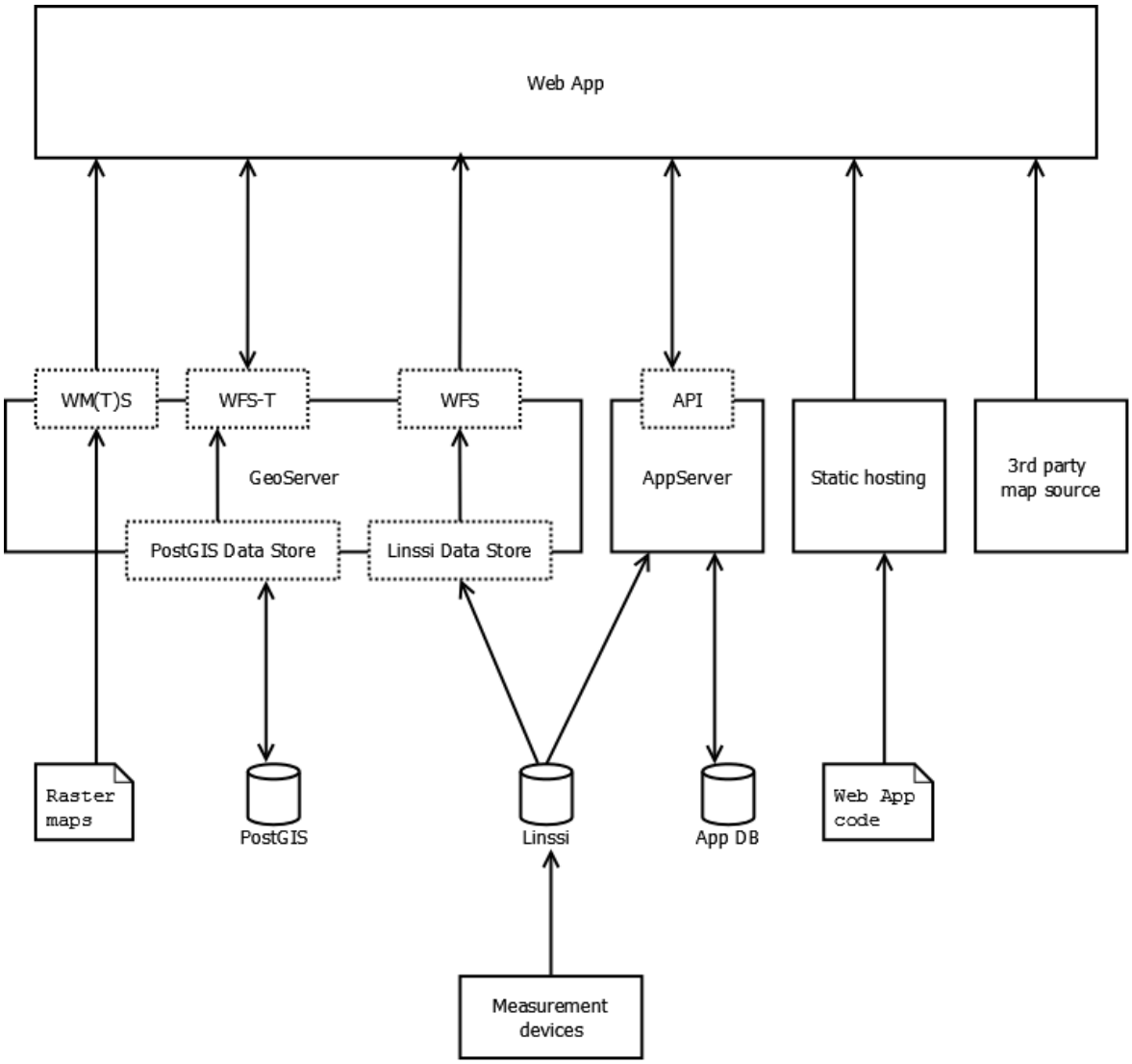

Figure 6 Overview of the system architecture

4.2.1 Linssi Database

Linssi is a MySQL based database and together with accompanied scripts for updating and administrating, it forms a database system for gamma-ray spectrometry. The system was developed by the Health Canada, Aalto University School of Science, and the Finish Radiation and Nuclear Safety Authority.

To fill the needs of laboratories performing gamma-ray spectrum analysis, the Linssi database has grown to be fairly complex. The version 2.3 has 54 tables and 740 fields, thou usually many of them are left unused. The database is design to house heterogeneous data for various applications. For the application developed in the thesis, the Linssi database

will act as access point to measurement data and provides the description of measurements setups. (Aarnio et. al. 2009)

The Linssi database has major limitation that need to be circumvented in design of the other components, particularly the Linssi Data Store. The only way of getting updates, for example when new measurements arrive, is to repeatedly poll the database. Also, due to its complex table design, queries to the Linssi database are fairly slow, particularly in live scenario when there is also the strain from writing operations as new measurements entries are added to the database. The database may be suboptimal, but due to its common use in the field and wide availability of data, it is the only viable option for the application.

4.2.2 PostGIS Database

The PostGIS is an extension of the PostgresSQL database that enables spatial operations, indexing and the use of geometries, for example points and polygons. For the application developed in the thesis, PostGIS database is used to store the drawings users can create and update via the application.

4.2.3 Geoserver

Geoserver is an open source map server software developed by the Open Source Geospatial Foundation. It implements many of the OGC standards, including WMS, WMTS and WFS-T. Geoserver is developed on Java programming language and utilizes the Geotools GIS library. It can be run as stand-alone server or hosted in Java Enterprise Edition web container.

Key concept of Geoserver is that it can handle maps from multitude of sources and formats and serve them as OGC standard services. For example, background map data can be stored as a geotiff, a geotagged image file, and some points of interest could be stored in a PostGIS database. Geoserver can combine both the raster and the vector source and serve them as a single WMS layer.

Geoserver functionality can be expanded with extensions. Most interesting ones are the

extensions that enable Geoserver to use geographic data from new sources. There are some available as open source software, but developing one is fairly straight forward process.

Geoserver was chosen to handle all the geographic data for its wide support of OGC standards and extensibility, but also due to existing experience with it as static map server and familiarity with the Java programming language. Geoserver is used for three tasks:

1. Serving static background maps from raster sources as WMTS.
2. Hosting user drawings from PostGIS database as WFS-T enabling inserting, modifying and deleting drawings.
3. Serving measurement data from Linssi database as WFS enabling complex queries including temporal parameters.

Completing the first two tasks is fairly easy as Geoserver has all the required components out of the box. Only thing to do is to provide the data sources and configure Geoserver to use them. Additionally, since WFS-T does not support pushing updates to clients, a separate solution is developed as part of the application server. This simple message broker will be covered in the chapter 4.2.4.

WFS was chosen to be the interface for serving measurement data. This was due to lack of both back- and front-end implementation for the SOS interface. WFS as more generic standard of the two, would provide all the required functionality as long as fitting feature type would be defined.

To enable serving Linssi measurement data from Geoserver, a custom data source extension must be implemented. In practice this means implementing some Geotools classes and wrapping them into a Java archive file with specific metadata file describing the extension. Geoserver automatically picks the extensions dropped on to its library folder on start-up. The developed extension is named LinssiDataStore and from now on referred by this name. The extension defines a feature type for measurements named LinssiMeas.

The queries to layers displaying data from the LinssiDataStore received through WFS

interface, or any other compatible OGC standard interface including WMS, are processed by the Geoserver and passed on to the data store. The provided CQL query parameters are translated into a SQL query and passed on to the Linssi database. The results from the database are transformed to Geotools simple feature objects of the LinssiMeas type and passed to the Geoserver and through it eventually to the querying client. This flow can be seen in figure 7.

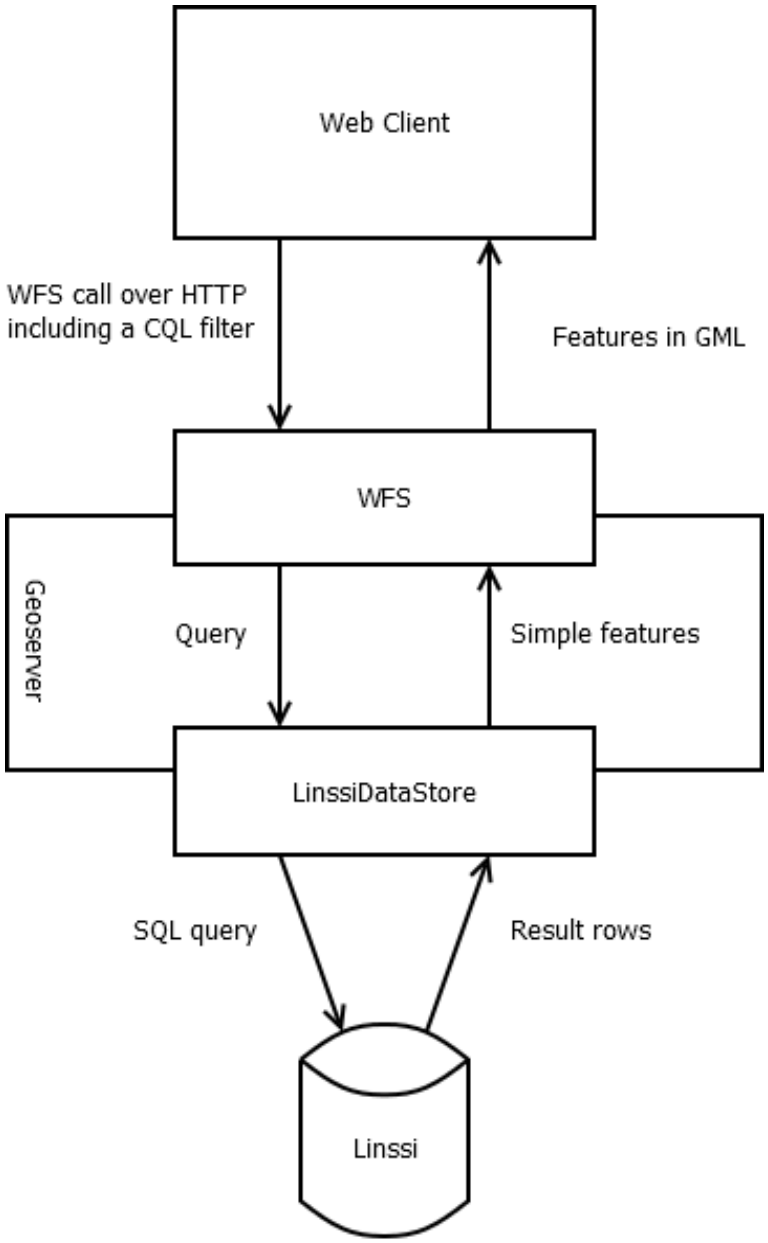

Figure 7 Dataflow from the LinssiDataStore

Since the WFS interface serving measurement data needs to be polled for updates and

database queries are too expensive to waste, a buffer between the WFS and database queries is used. This is particularly the case with multiple clients connected to the server. In some applications, the Linssi database is known to be already under heavy load just from data input.

The LinssiDataStore polls the database for new measurements on frequent interval and deposits the result to the cache. Polling queries from clients are satisfied by checking for updates in the cache. The dataflow in the cache mechanism can be seen in figure 8. Clients making WFS requests are expected to mark update polls with a dedicated query parameter. This separates the special light weight update polling queries from the heavy history queries that go directly to database. Adding such a parameter is an extension to the WFS standard, but it is necessary to enable fluent live updates to the clients.

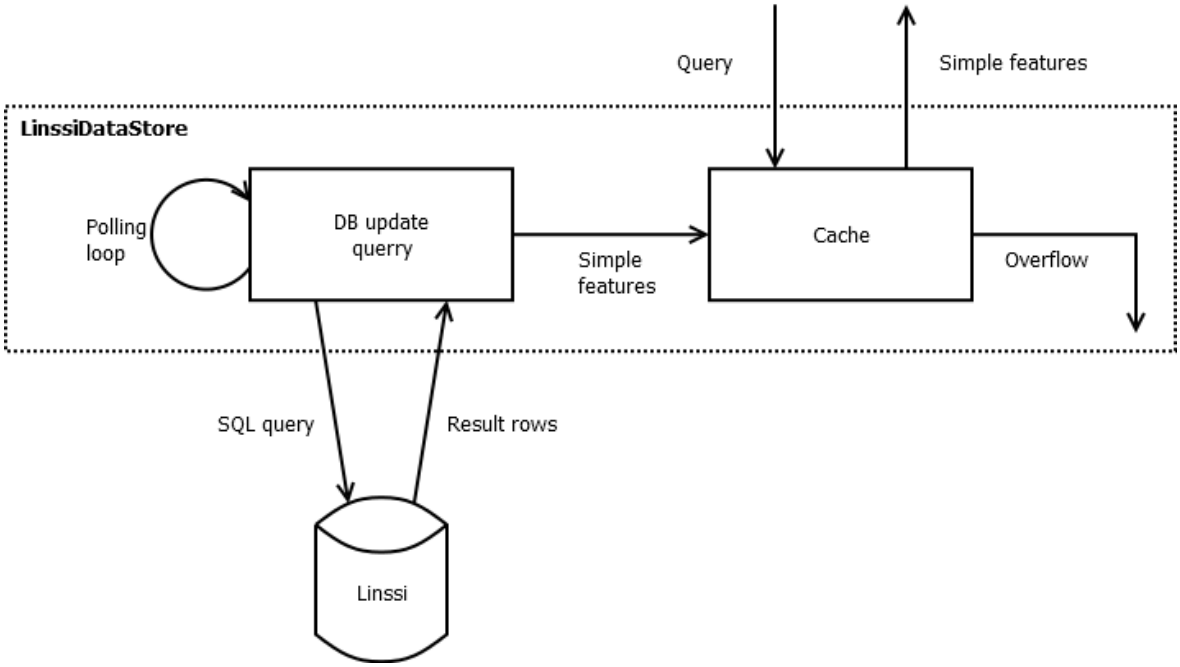

Figure 8 Internal caching mechanism of the LissiDataStore

4.2.4 Application server

The application server, named AppServer, is the back-end component for all non-geographic data and logic. The server connects to the Linssi database, but also has its own database for storing user account information. The AppServer is developed using Java Servlet technology. Conveniently, since also Geoserver is servlet based, this allows hosting

both servers on the same Java Enterprise Edition web container. Alternatively, the servers can be hosted separately, since all dependencies between them are run over the HTTP protocol.

Internally the AppServer consists of multiple independent services implemented as Servlets. Each service provides an API that can be accessed via the HTTP protocol. These services are listed in table 5.

Table 5 Descriptions of services provided by the AppServer

Service name	**Description**
Authentication	Provides login and token authentication for other services and also for the Geoserver.
User management	Allows querying and editing user information and adding new accounts.
Update broker	Simple long poll based message broker. This is used to notify other live clients about changes to user drawings.
Facilities	Provides descriptions of measurement setups.
Measurements	Provides both detailed information on single measurement as well as a light way to query overview of the measurements.
Spectrum	Provides access to spectrum data.
Symbol library	Lists symbols available on the server for user drawings.

4.2.5 Web User Interface

The web user interface is developed as a Single Page Application (SPA). In a typical web page, each view is loaded over the HTTP from web server. Data is parsed in to the view on server side. In a SPA, there is only the initial page load and data for different views is parsed on client side. Essentially, SPA is the client application in client server architecture that gets loaded via HTTP and is executed in a web browser. The SPA client handles everything related to representation of the data, leaving the server to deal with data storage and business logic.

Based on the result of the component evaluation described in chapter 4.1, Open Layers 3

was used as a map component in the application. In addition to being able to render interactive maps, Open Layers 3 also provides tools for parsing GML in to JavaScript objects and making calls to OGC standard services.

Main view of the Web user interface application can be seen in figure 9. The application can be configured to show map layers from various sources. The layers can be displayed simultaneously even if they used different map projection. The Open Layers 3 automatically re-projects the source materials to the same projection.

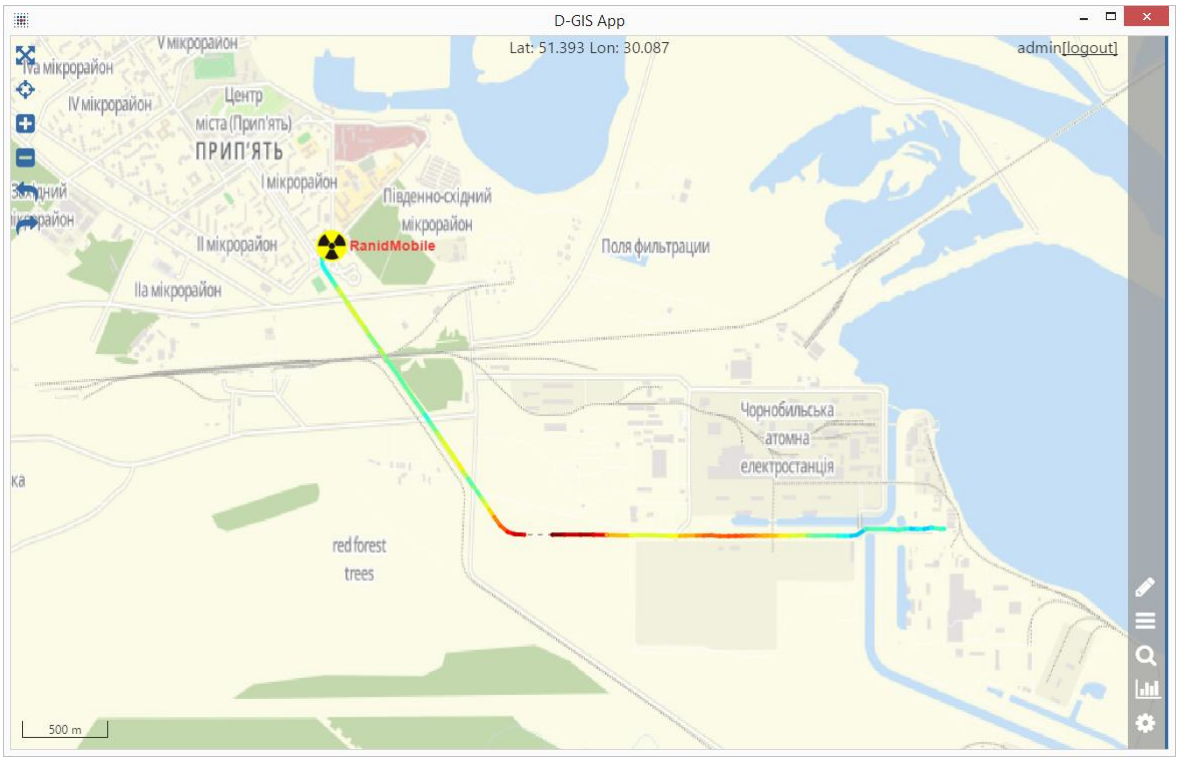

Figure 9 Main view of the Web user interface displaying radiation dose rate measurements. The background map is a rendition of the OpenStreetMap data used under Open Data Commons Open Database License. Copyright OpenStreetMap contributors.

Measurements are displayed forming a continues path on map. Separate path is used for each mobile measuring facility. Data can be either historic record from the database or live feed from ongoing measurement activities.

Typical configuration for a measuring device in a Linssi system is to run a measurement every four seconds. This leads to fairly large dataset in relatively short time. A full 24-hour day of measurements with just a single device would lead to over 20000 data points. To

enable displaying this kind of numbers and still keeping the user experience smooth, simple clustering algorithm is applied. For each zoom level, measurement points are grouped so that only one point is displayed when the points would be only a few pixels away from each other on the map surface. The most significant measurement in terms of dose rate and alarm state is selected to represent the group.

5 DISCUSSION AND CONCLUSIONS

The component selection process designed in this thesis proved to be a success. The selected component fully filled the expectations placed upon it. Running the entire selection process took some time out of the regular software development work. Component selection was not the only outcome from this time investment. As a side product of the selection process, knowledge on the field grew. This knowledge was useful for the parallel and subsequent software development activities.

The process, or rather its implementation and capability statements chosen for it, ended up being fairly functionality oriented. It could have better addressed social and economic aspects. The reason for this shortage was due to the difficulty of forming capability statements concerning for example organisational risks so that both commercial and open source software could be evaluated.

At least GIS seems to be a field where both commercial and open source software should be considered. Comparison of the non-functional aspects of these diverse offerings is a field that could have potential for further research.

The decision to use the WFS standard instead of the more measurements oriented SOS was done more based on the economics than functionality. The SOS did not have suitable existing implementations so using it would have required more work to be done. Use of WFS opened opportunities to use existing third party components in both the front- and back-ends of the system.

The WFS was fully capable of serving the kind of measurement information used in the developed system. For a system with more diverse data from variety of sources there could have been problems. WFS does not standardise the form of the measurement data. A separate schema for that must be maintained on top of the WFS. This could be an issue with a larger and more diverse system. In this light, is may be questioned whether a fully proprietary service would have worked as well for system implemented in this thesis.

The Linssi database system that was used as a data source to the developed application imposed some limitations. The only way to get a feed of live measurements was to poll the database on a frequent interval. Compared to a design were the measurements would be pushed by the data source, or even by the devices themselves, to the system, the polling appears inferior.

Despite that, the live feed functionality of the developed application ended up being adequate. The measurement data is entered to the Linssi database on an even interval. Polling the database on a similar interval lead to fairly steady stream of data. Of course the polling loop delays add up and the measurements will not appear to the map on real time. However, the delay is relatively small and this near real-time solution is adequate for the use it is intended for.

A bigger issue with the Linssi database system was its complexity and lack of spatial indexing. This made doing large searches for historic measurement data very time consuming. To circumvent this, searches that would only pick a sparse result set were developed. The only way to fully evade this problem is to use a more spatially capable database solution.

6 SUMMARY

This thesis was set out to design and implement a geographic information system for mobile measurement data. To achieve its goal, designing a component selection process was required.

To support the component selection process design, literature was reviewed for both commercial and open source software component selection processes. It was found that the commercial side has abundance of proposed processes. The open source side appeared to lean more towards evaluating the community behind the software rather than the product itself. There appeared to be lack of work addressing the side-by-side comparison of both commercial and open source offerings.

Geographic information systems and the Open Geospatial Consortium standards were overviewed. It was found that for a performant system the form of data storage and support for spatial operations are important. It was also found that the standards overlap and one should pay attention also to the non-technical aspects when choosing one.

A component selection process was defined and used to select a component for the system to be developed. The selection was successful, but the comparison of non-functional aspects of commercial and open source software proved to be difficult.

The geographic information system was implemented relying on web feature service standard for measurement data transfer. The standard was adequate for the implemented system, but required maintaining a separate schema to define the form of measurement data.

REFERENCES

Aarnio, p., Ala-Heikkila, J., Hoffman, I, Ilander, T., Klemola, S., Mattila, A., Kuusi, A., Moring, M., Nikkinen, M., Pelikan, A., Ristkari, S., Salonen, T., Siiskonen, T., Smolander, P., Toivonen, H., Ungar, K., Vesterbacka, K., Zhang, W., 2009. Linssi SQL database for gamma-ray spectrometry part 1: Database Version 2.3. pp.1-7

Bermudez, L., Bogden, P., Bridger, E., Cook, T., Galvarino, C., Creager, G., Forrest, D., Graybeal, J., 2009. Web feature service (WFS) and sensor observation service (SOS) comparison to publish time series data. CTS '09 Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems. pp.36-43

Chung, L., Cooper, K., 2004. Defining Goals in a COTS-Aware Requirements Engineering Approach. Systems Engineering 7(1), pp.61–83

Comella-Dorda, S., Dean, J., Morris, E., Oberndorf, P., 2002. A Process for COTS Software Product Evaluation. COTS-Based Software Systems, Volume 2255 of the series Lecture Notes in Computer Science. pp.86-96

Di Giamo, P., 2005. COTS and Open Source Software Components: Are They Really Different on the Battlefield?. COTS-Based Software Systems, Volume 3412 of the series Lecture Notes in Computer Science. pp.301-310

Hevner, A., March, S., Park, J., Ram, S., 2004. Design science in information systems research. MIS Quarterly, 28(1). pp.75-105

Kontio, J., 1995. OTSO: A Systematic Process for Reusable Software Component Selection. University of Maryland Technical Reports. University of Maryland. College Park, MD.

Kunda, D., Brooks L., 1999. Applying social-technical approach for COTS selection. Proceedings of 4th UKAIS Conference.

Mainden, N. A. M., Ncube, C., 1998. Acquiring COTS software selection requirements. Requirements Engineering, 1998. Proceedings. 1998 Third International Conference on. pp.46-56

Nagy, G., Wagle, S., 1979. Geographic Data Processing. Computer Surveys 11(2). pp. 139-181

Ooi, B., C., 1990. Efficient Query Processing in Geographic Information Systems. Lecture Notes in Computer Science 471.

Open Geospatial Consortium, OGC History (detailed). [Online] Available at <http://www.opengeospatial.org/ogc/historylong> [Accessed 22 August 2016]

Open Geospatial Consortium, 2006. OpenGIS Web Map Server Implementation Specification

Open Geospatial Consortium, 2010. OpenGIS Web Map Tile Service Implementation Standard

Open Geospatial Consortium, 2010b. OpenGIS Web Feature Service 2.0 Interface Standard

Open Geospatial Consortium, 2012. OGC Geography Markup Language (GML) — Extended schemas and encoding rules

Open Geospatial Consortium, 2012b. OGC Sensor Observation Service Interface Standard

Open Geospatial Consortium, 2016. OGC Catalogue Services 3.0 - General Model

Petrinja, E., Nambakam, R., Sillitti, A., 2009. Introducing the OpenSource Maturity Model. FLOSS '09 Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. pp.37-41

Stol, K., Ali Babar, M., 2010. A Comparison Framework for Open Source Software Evaluation Methods. Open Source Software: New Horizons, proceedings of the 6th International IFIP WG2.13 Conference on Open Source Systems (OSS). pp.389-394

Taibi, D., Lavazza, L., Morasco, S., 2007. OpenBQR: a framework for the assessment of OSS. IFIP — The International Federation for Information Processing 234. pp.173-186