TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ANTTI KOSKENALHO

LIFE SCIENCE SOFTWARE FRAMEWORK ALTERNATIVES IN RESOURCE SCARCE CONTEXT

Master of Science Thesis

Examiner: Professor Hannu-Matti Järvinen
Examiner and topic approved by the Council of the Faculty of Computing and Electrical Engineering on 6 April 2016

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
KOSKENALHO, ANTTI: Life Science Software Framework Alternatives in Resource Scarce Context
Master of Science Thesis, 55 pages
May 2016
Major: Distributed Software
Examiner: Professor Hannu-Matti Järvinen
Keywords: Life science software evaluation, virtual microscopy, web-based application development

Feature rich applications need to be delivered rapidly given the lean structure of many businesses today. Recently the number of available customizable existing software solutions has increased, enabling even small development teams to deliver complex solutions. However, small development teams still face serious risk of failure if unexpected limitations in modifiable off-the-shelf software prevent sustainable solution to business problems.

This thesis introduces a new method for evaluating available customizable existing software in the context of a small development team. As a real-life example a complex whole slide imaging feature is developed into web-based life sciences research application. The introduced evaluation method is used for evaluating different implementation approaches and different whole slide imaging solutions. Finally one solution is picked and integrated with the research application and the suitability of the evaluation method is evaluated.

The evaluation method introduced in this thesis helps utilizing small development teams' limited resources to build complex software. The method can be generalized to be used to any development teams use, regardless the team's size and to any software project, regardless the nature of the software.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
KOSKENALHO, ANTTI: Biotieteen ohjelmistoviitekehysvaihtoehdot niukkaresurssisessa ympäristössä
Diplomityö, 55 sivua
Toukokuu 2016
Pääaine: Hajautetut ohjelmistot
Tarkastaja: professori Hannu-Matti Järvinen
Avainsanat: Biotieteen ohjelmistojen evaluointi, virtuaalimikroskopia, verkkopohjaisten sovellusten kehitys

Monien yristysalojen luonne vaatii, että sovelluksia täytyy toimittaa aina vain nopeammin tinkimättä ohjelmiston ominaisuuksien määrästä. Viimeaikainen valmiiden muokattavissa olevien ohjelmistoratkaisujen määrän kasvu on mahdollistanut pienehköjen kehitystiimien toimittaa monimutkaisia ohjelmistojaratkaisuja, käyttäen hyväksi jo olemassa olevia ohjelmistoratkaisuja. Pienet ohjelmistokehitystiimit ottavat kuitenkin riskin, sillä muokattavissa olevat valmiit ohjelmistoratkaisut saattavat sisältää odottamattomia rajoitteita, jotka estävät kestävien ohjelmistoratkaisujen kehittämisen.

Tässä opinnäytetyössä esitellään pienille ohjelmistokehitystiimeille sopivaa uutta arviointimenetelmää, jota käytetään arvioimaan valmiita muokattavissa olevia ohjelmistoratkaisuja. Opinnäytetyön esimerkkitapauksessa toteutetaan virtuaalimikroskopiaominaisuus olemassa olevaan verkkopohjaiseen biotieteiden tutkimussovellukseen. Esitettyä arviontimenetelmää käytetään erilaisten ohjelmistokehitystapojen sekä valmiiden virtuaalimikrosopiaohjelmistojen arvioimiseen. Lopuksi yksi ohjelmistoratkaisuista valitaan ja integroidaan tutkimussovelluksen kanssa sekä arviointimenetelmä sopivuus arvioidaan.

Tässä opinnäytetyössä esitetty arviointimenetelmä auttaa hyödyntämään pienten ohjelmistokehitystiimien rajoitettuja resursseja monimutkaisen ohjelmistojen rakentamisessa. Arviointimenetelmä voidaan myös yleistää minkä tahansa ohjelmistotiimin käyttöön tiimin koosta riippumatta sekä minkä tahansa ohjelmistoprojektin käyttöön välittämättä ohjelmiston luonteesta.

PREFACE

This Master of Science Thesis has been accomplished while working as part of PELICAN – personalized cancer medicine group, at the premises of Prostate Cancer Research Center's molecular research laboratory in Tampere. I would like to thank all of the people who are working hard there to understand the fundamental nature of prostate cancer and developing valuable information and new methods for finding the cure one day.

I would especially want to thank Professor George Steven Bova for including me in ILSR development team, patiently introducing me into the world of prostate cancer research, supervising me in ILSR development process and writing process of this thesis. I would also like to thank ILSR development team's senior developer Marc Rohrer, who has given me technological guidance while developing the ILSR. My gratitude also goes to Professor Hannu-Matti Järvinen, who has been directing and keeping me on the right track with my thesis process.

Finally I would like to thank my fiancé Alma Viheräkoski for supporting me at my writing work and cheering me up with her best jokes while I was stressed out. Special thanks also goes to my friends Akseli and Pekka, for being so understanding about my absence during the writing process.

Tarttila, 7.5.2016

Antti Koskenalho

TABLE OF CONTENTS

TERMS AND DEFINITIONS

API	Application Programming Interface.
CT	Computed Tomography: imaging modality used for building 3d reconstructions of organs.
DICOM	Digital Imaging and Communications in Medicine: data interchange standard for biomedical imaging.
ESA	European Space Agency.
FS	Free Software.
H&E	Hematoxylin and eosin: stain used in dying cut sections.
IIIF	International Image Interoperability Framework: a network protocol for streaming image data.
IIP	Internet Imaging Protocol: a network protocol for streaming image data.
ILSR	Integrated Life Science Research: web-based research application that supports life science research's day-to-day work.
IRCA	Identify Review Compare Analyze: method for evaluating open-source software.
JPIP	JPEG2000 Interactive Protocol: a network protocol for streaming JPEG2000 images.
LCM	Laser capture microdissection: technology for cutting section to smaller sub-sections using infrared laser beam.
MRI	Magnetic Resonance Imaging: imaging modality used for example detecting tumours from soft tissue.
MVC	Model-View-Controller software pattern.
NASA	National Aeronautics and Space Administration.
NIST	National Institute of Standard Technology.
OSS	Open-Source Software.
PCRC	Prostate Cancer Research Center.
PELICAN	Project to ELIminate lethal CANcer: Personalized Cancer Medicine Group, a research group led by G. S. Bova.
PET	Positron Emission Tomography: imaging modality that provides information for example about how organs function.
SEI	The Software Engineering Institute of Carnegie Mellon University, Pittsburgh, PA.
SPECT	Single-photon Emission Computed Tomography: imaging modality which provides information for example how organs function.

1. INTRODUCTION

A software project's chances of succeeding get better if it can rapidly cycle between user needs and function delivery. Building feature-rich software from scratch is time consuming and usually requires expertise from several different fields. However, use of customizable off-the-shelf software, such as open-source software, has changed the nature of software development recently. Now small software development teams theoretically have the same capability to develop complex solutions previously requiring far greater resources. But the new opportunity comes with the risk that the small team does not sufficiently account for inherent limitations in the new adapted software. In the worst case the created software cannot provide a sufficient solution to the business problem or cannot be maintained properly.

What specifically makes it easier for small teams to develop feature-rich applications today, as compared to twenty years ago? First, it has become common for software to be developed using open-source frameworks. Second, integrating software with third-party frameworks has been eased, as software solutions are developed using more modular software patterns and application programming interfaces.

This thesis describes a process of developing complex software functions supporting deep integration of imaging in life science research. The work introduces a new evaluation method for evaluating different implementation approaches and different ready-made software solutions, two processes which together intend to reduce the risk of failed software development particularly in the setting of a small software development team. In a specific test case, different implementation approaches to enable whole slide imaging feature to an existing web-based life science research application are evaluated together with different available whole slide imaging solutions, using the introduced evaluation method. Once a potentially suitable solution is found, it is integrated with the research application and finally the newly formed feature is evaluated to determine success of the development process.

To give a wider view of the environment into which the new feature will be added, the concept of imaging in medicine and in life sciences are explained in the second chapter of this thesis. Imaging is used for varied purposes in medicine and life sciences and respectively different modalities are used for acquiring the images. One of these modalities is called whole slide imaging and its purpose is to enable more convenient analysis of microscopic images of stained tissue sections, allowing high resolution zoomable image information to be delivered without having to access the physical glass slides. The second chapter answers questions like: How are whole slide images generated? What purposes are whole slide images typically used for? What challenges does

whole slide imaging generate? Answering these questions are essential when requirements for the whole slide imaging framework are gathered later in this thesis.

After describing the wider environment of the new feature, a web-based life science research application called Integrated Life Sciences Research, or ILSR, is introduced in the third chapter to explain the new feature's more immediate environment. At first, vision of ILSR is described: Why does ILSR exist and what kind of problems does it attempt to solve? Then current state of ILSR and its key areas are introduced, following description of how whole slide imaging fits in the current system and what new features it helps enabling in the future development of ILSR. Knowing the current system, its vision and future plans are required when gathering requirements for the framework that is going to be integrated with. Many of the requirements can be derived directly from the main applications requirements and some of them will be derived from the main systems vision and future plans.

Different software evaluation methods are presented in the fourth chapter. Evaluation methods are used for gathering information about different available software solutions, which possibly are suitable for solving a business problem, and for narrowing down the selection first to top candidates and finally leading to selection of one most suitable solution. First features of two existing software evaluation methods are described. Then their advantages and disadvantages are discussed in context of small development team size and a new evaluation method is introduced, attempting to take developer teams' limited resources into account in the evaluation process.

The path for choosing the best implementation approach and most suitable whole slide imaging software solution using the introduced evaluation method begins in the fifth chapter. Requirements for the framework are gathered as a first step, followed by evaluation of different implementation approaches, the main question being: What is the right balance between implementation work done by the development team and the usage of ready-made software solutions? Too much work will burden the development team, and in the other hand, relying too much on ready-made solutions weakens influence over future development paths, as they are dependent on solution provider's vision and interests. Choosing a sustainable implementation approach is dependent also on what software solutions are available. Suitability of different whole slide imaging solutions are evaluated based on the gathered requirements. Finally the top candidates are compared against each other and decision for implementation approach is made, leading to selection of the most suitable solution for implementing ILSR's whole slide imaging feature.

The sixth chapter describes how successful the implementation of the whole slide imaging feature was. Were there any issues and challenges in the integration? If there were, how were they solved? Did the framework meet the requirements gathered in fifth chapter? Did the end product correspond to what was expected? Consideration of how well the implemented whole slide imaging feature and its planned future improvements are going to support ILSR's future development and overall vision are discussed. At

last, the used evaluation method's found weaknesses and advantages are discussed to find out how effective the evaluation method was.

Finally, the overall process of building a new feature using a small development team's limited resources represented in this thesis is evaluated in the seventh chapter. What are the challenges of the introduced new evaluation method? Could the method be improved and refined even further? Does the new method only aid low-resource developer teams, or could it be used as a lever regardless the size of the developer team? Could the process be generalized for the needs of other development projects or is it specific to life science software projects? And at last, was the process considered useful in ILSR's development and will it be used again in the future?

2. IMAGING IN MEDICINE AND LIFE SCIENCES

Different imaging modalities are used for different purposes in medicine and life sciences. Some modalities are suitable only for acquiring data from living organisms whereas some other are designed gathering data from extracted tissue samples, and some are capable of doing both. Some of the many imaging modalities used in medicine and life sciences are presented in Table 2.1. In addition to these modalities, there are number of medical imaging fusion techniques that combine different modalities to provide integrated data which is critical for decision-making in some contexts.

Table 2.1. *Imaging modalities used in medicine and life sciences*

Imaging Modality	**Example of usage**
Magnetic Resonance Imaging (MRI)	Detecting tumours in soft tissue
Computed Tomography (CT)	Building 3D reconstructions of organs
Positron Emission Tomography (PET)	Providing information about how organs function
Single-photon Emission Computed Tomography (SPECT)	Providing information about how organs function
X-Ray	Detecting broken bones and cavities
Ultrasound	Imaging foetus in pregnancy
Whole Slide Imaging	Studying cut sections

Magnetic resonance imaging (MRI) acquires image data using radio waves with magnetic field [1]. It is often used for diagnosing brain tumours, but has many different uses, for example to detect differences between normal and diseased soft tissues in blood vessels, breasts, bones and joints, spinal injuries and organs in the pelvis, chest and abdomen [1; 2, p. 6]. The advantage with MRI is that it can be used safely as it does not expose patient to radiation and that it can produce high quality images, but one challenge is that it is highly sensitive to movement during imaging [2, p. 7].

X-ray computed tomography (CT), or CAT scan, uses multiple X-ray projections to produce cross-sectional images of organs and other areas inside a subject [1]. Images acquired by CT are used for building very precise 3D reconstructions of organs, and those reconstructions can be used for multiple purposes. The reconstructions can be used for example for studying structure of brain and for assisting surgical planning, training and guidance, as well as for studying other soft tissues, pelvis, blood vessels, lungs and other organs [1; 2, p. 7]. CT provides high quality images, but unlike in MRI, the type of radiation used is known to cause dose-dependent damage to DNA and other

cell structures [3]. CT uses one of the oldest imaging modality, X-ray, as part of the imaging process. When used alone, X-ray technology sends X-ray beams through the subject to form a tissue-x-ray attenuation-based image of internal structure and is usually used for example detecting broken bones, cavities and swallowed objects [1].

One type of CT modality, positron emission tomography (PET), similar to MRI, is also often used for brain diagnosis, but can also be used for detecting cancer and heart conditions as well as evaluating how well treatments affect [1; 2, p. 8]. When using PET, radioactive tracers attached to targeting molecules are first injected into a patient's vein and the patient is then scanned to create an image of how tissues and organs are functioning [1]. When compared to MRI and CT technologies, PET imaging technology suffers from low image quality but can produce high sensitivity depending on the quality of molecular targeting [2, p. 8]. In the context of PET imaging, more sensitive means the ability to detect and record a higher percentage of emitted events which may for example be related to a particular molecular phenotype, such as expression of PSMA in prostate cancer cells, as opposed to being based solely on attenuation characteristics [4, p. 194; 5].

Another form of CT technology, which works very similarly to PET, is single-photon emission computed tomography (SPECT). SPECT is often used for studying blood flow of tissues and organs [2, p. 9]. Same way as PET, SPECT also uses radioactive tracers injected into blood to acquire images [6]. As the two modalities are so similar, what is the main differences and why is one modality usually chosen over another? Some reasons for choosing SPECT could be the better availability, wider usage and lower costs, as PET images are less prone to artefacts, require shorter scan times and have better spatial resolution [7].

Medical ultrasonography, or ultrasound, uses high frequency sound waves to generate images [1]. Ultrasound is probably best known for its application to image the foetus in pregnancy, but can also be used for detecting abnormalities in the heart and blood vessels, imaging organs in the pelvis and abdomen and to evaluate symptoms of pain, swelling and infection [1].

These modalities can be used as they are for diagnosis and to aid treatment, as mentioned previously in this chapter, but often modalities are also combined to get even better results. For example PET and CT can be combined to PET-CT to get more precision, improving oncologic care by improving treatment decisions, disease recurrence monitoring and patient outcomes [1]. Other combinations, just to name few, include MRI-PET, PET-CT, SPECT-CT, ultrasound-MRI, MRI/CT-PET-SPECT and so on [2]. The idea is that any imaging modalities can be used, and are used already, with other modalities to get the best images suitable for the purpose.

2.1 Whole Slide Imaging

Whole slide imaging, or virtual microscopy, is an imaging modality used in pathology, whereas modalities mentioned in Chapter 2 are most often used for imaging live organs inside the patient. Whole slide imaging uses digital images, or digital slides, scanned from conventional glass slides containing sections of tissue obtained from patients typically by biopsy, after surgical removal of organs, or at autopsy. Whole slide imaging does not have the same restrictions as modalities introduced above, as challenges caused by motion and harmful radiation do not need to be considered. One advantage of whole slide imaging is also that it produces very high resolution images when compared to other modalities.

Whole slide imaging has many uses in medicine and in life sciences. For example, digital slides are routinely used for both more convenient local and also for remote diagnosis and consultation by pathologists, researchers, clinicians, and students across the spectrum of health care in all types of organisms. Whole slide imaging is most typically performed using bright field (white light) microscopy, but is also routinely performed using immunofluorescent microscopic imaging. [8]

Whole slide imaging (digital slides) have several advantages when compared to direct manual examination of stained tissue sections mounted on glass slides. Digital slides can be accessed remotely and instantly, so there is marked reduction in time and effort required especially if re-examination of a given slide is required. Tissue sections on glass slides are fragile and prone to scratching, loss of coverslips, oil and glue smudging, and are easily misplaced. Well-collected, properly obtained digital slide images are more difficult to lose, and easier to maintain if properly electronically backed up. In education, a set of digital slides can be relatively easily shared among students, even if the specimen is rare. Digital slides also allow having multiple layers of non-destructive annotation of images for various purposes. One recently achieved big advantage is also that viewing digital slides does not require any special viewing equipment, as recent developed consumer displays (including tablets and mobile phones) are of sufficient resolution in the setting of browser viewing and zooming capacity to enable evaluation of high resolution histologic images. [8]

Even though whole slide imaging has many advantages compared to manual examination of stained tissue sections mounted on glass slides, direct manual microscopic examination is still the norm in most pathology laboratories. Digital slides are always the product of the imaging of a physical glass slide with stained tissue mounted and cover-slipped on the slide. Only limited DNA and RNA analysis can be done using tissues mounted on slides and for example high throughput sequencing of DNA and RNA requires actual microdissection of tissue and isolation of DNA into solution — steps that to a large extent must take place in a test tube, and not on the glass slide and certainly not from a histological image.

The process of gathering digital slides using histopathology's methods is presented in **Figure 2.1**. The process starts when tissue is collected. Tissues can be collected using

different techniques: biopsy is used for removing tissue from a living subject, surgery can be used for removing larger specimens, or tissues can be also collected in autopsy. After collection, removed tissue can be covered with ink to mark the margins, and different colours of ink can be used for marking different areas or orientation of the tissue. After inking phase, tissues are placed on plastic cassette and they are fixed. Tissue fixation is done to preserve tissue components sufficient for routine feature identification. The fixative is selected depending on type of tissue and features to be studied. Five major groups of fixatives are aldehydes, mercurials, alcohols, oxidizing agents and picrates. Formalin, which is type of aldehyde, is often used in immunohistochemistry and glutaraldehyde, another type of aldehyde, is often used in electron microscopy. The best application for mercurials is fixation of hematopoietic and reticuloendothelial tissues. Three remaining fixative types, alcohols, oxidizing agents and picrates are used less often, but they have important specialized applications. [9]

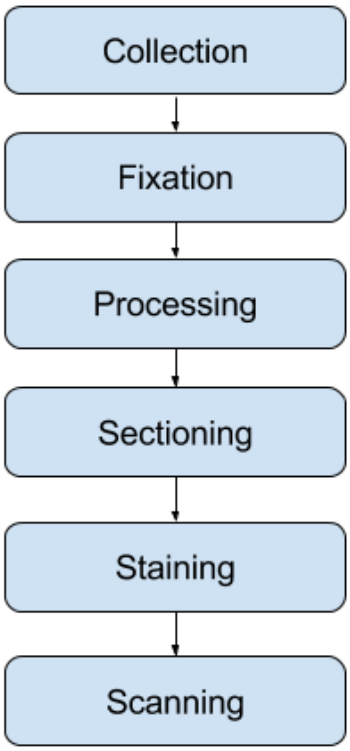

Figure 2.1. *Procedure to gather digital slides*

After tissue has been fixed, it gets processed so that it can be cut into sections later. Often stabilization of the tissue for sectioning is achieved by infiltrating the tissue with paraffin. Freezing of fresh tissues is used as an alternate method to obtain both preservation and fixation of tissue, but it is not the preferred method in clinical practice because it provides markedly reduced histologic quality and is relatively expensive, since frozen blocks must be stored in -80C freezers and paraffin embedded tissue blocks can be stored at room temperature. In paraffin based stabilization of fixed tissue, the tissue is first dehydrated using series of alcohols or mixture of formalin and alcohol. Then the

tissue is cleared of the dehydrant, usually with xylene. Finally, the tissue is embedded (infiltrated) with paraffin. [9]

In sectioning, tissue is cut into thin slices, which are routinely called “tissue sections”, which then can be placed on a glass slide. When paraffin is used, tissue is cut into typically 4 micron thick sections using a microtome. Cut sections are placed on glass slides and placed in a warm oven to melt the paraffin and enable adherence of the tissue to the glass. [9]

After sections are attached to glass slides, the tissue is then stained to allow visualization of cell morphology using light or fluorescence microscopy. The routine stain for human tissue accepted around the world is hematoxylin and eosin, or H&E staining, but hundreds of other staining methods are used for various purposes. After staining, the section is covered mounting medium, a type of glue that is clear and has a refractive index similar to microscope slide glass (to reduce diffraction) and is covered with a coverslip, which is a thin piece of plastic or glass. Sections cannot be well-visualized without coverslips because of diffraction that occurs at the tissue/air interface if no coverslip is in place. [9]

Finally, a special equipment called whole slide scanner is used for scanning the glass slide, producing digital images called digital slides. Whole slide scanner can be used for scanning slides manually, slide by slide, or automatically by doing batch scanning. Briefly, a whole slide scanner is a combination of microscope with lens objectives, light source, robotics to load and move glass slides, digital camera or cameras and a computer loaded with software to manipulate, manage and view the digital slides [8]. Scanner moves the slide under microscope while digital camera acquires images. Some systems uses one camera to acquire the actual image data while second camera is used for automatically monitoring and adjusting the focus continuously, in order to create two-dimensional image out of a three-dimensional slides [10, p. 131]. Finally, after the whole slide has been imaged, produced images are stitched together to create digital representation of the glass slide. The scanner usually saves the same image using different resolutions, so that it can be later viewed using different zoom levels [11, p. 3].

There are several factors that affect the quality of the digital slide: image being out of focus, near-far effect, dirt and dust fragments, pen marks on the cover slip, folds in the tissue and bubbles between the coverslip and sample. Different things can cause image being out of focus: the slide might sit loose in the tray, there might be impurities on the coverslip, or focus points are selected from different focal plane than tissue. Near-far effect causes that only parts of the image are in focus, which can happen when scanning thick tissue sections without enough focus points. Dirt, dust fragments and pen marks on coverslip, can cause out of focus when focus points are picked from the same positions, and can cause trouble later when executing image analysis for the digital slides. Similarly folds in tissue and bubbles between coverslip and sample, both caused by tissue section phase, can also cause out of focus and problems with image analysis. [12]

2.2 Whole Slide Imaging Processes

Whole slide imaging includes several non-standardized processes introduced in Figure 2.2. Once digital slides have been scanned, they need to be archived and managed for later usage. When planning the archiving, whole slide images' big size, compared to regular images, needs to be taken into consideration: uncompressed 40x scanned digital slide can get as large as 14.5 GB, although the same image takes only 576 MB after JPEG2000-compression [12]. Half of a gigabyte is still a large size for one image, given that in big laboratories hundreds of images can be generated within a day. In addition to reserving enough disk space, fast enough network connectivity between the scanner system and storage is also required. If 200 MB image is scanned every minute, the network usage will be around 30 Mbps, so at least 50 Mbps connection should be reserved, and if bigger 40x scans needs to be taken into consideration, 175 Mbps connectivity should be reserved [12]. When storing whole slide images, it is also important to make sure that consistency between digital slide and its metadata, such as scanning parameters, used staining, used fixative and such are preserved. If actual digital slide files needs to be shared amongst multiple collaborators, same size and consistency requirements that affected archiving needs to be kept in mind.

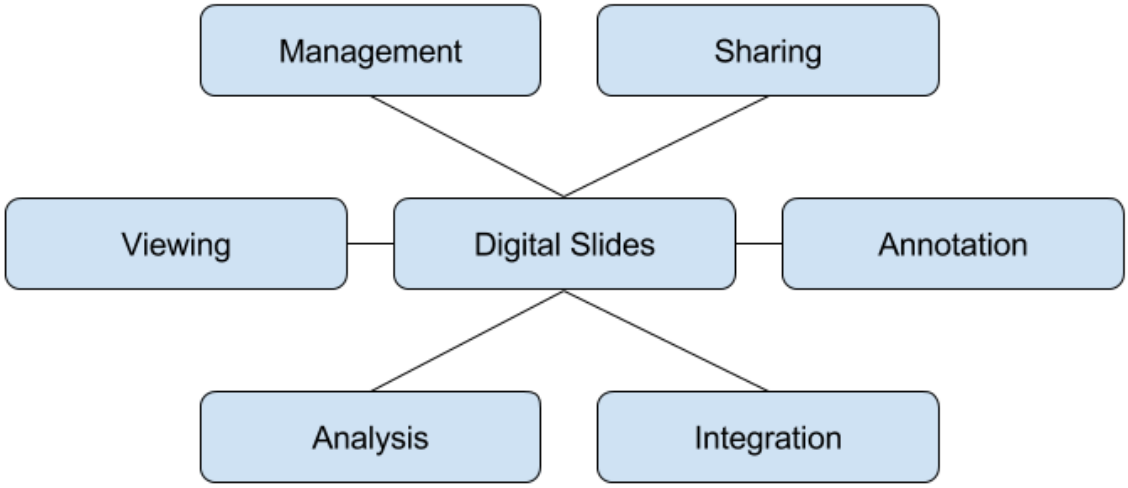

Figure 2.2. *Whole slide imaging processes*

Remote viewing can be used for sharing, if sharing the actual image files is not a requirement. In remote viewing, dedicated server will process the digital slide file and delivers only parts of the image at the time to the viewer software, using network connection. Viewer software can be relatively "light" in terms of memory usage and computational performance, and can be run even on mobile devices such as tablets and mobile phones, although usually bigger displays are used for optimal viewing experience.

The techniques between different digital slide formats are usually similar. Digital slides can be stored using "lossy" compression, for example JPEG2000, or using "lossless" compression, for example TIFF format [8]. Whichever compression method is chosen, digital slides are typically organized into thousands of image tiles which are

organized into a pyramidal hierarchy of addressable sectors to which the user can selectively interact by panning and zooming. Using this method enables efficient transition laterally across various x,y coordinates, and ascending or descending in perceived magnification (as resolution is varied), and reduces data transfer between component that processes the image data and component that views the image [8]. The reduced data transfer is based on the fact that the viewer needs only some tiles of the whole image data at any given time, depending on the zooming and panning of the image. The pyramid format is illustrated in **Figure 2.3**.

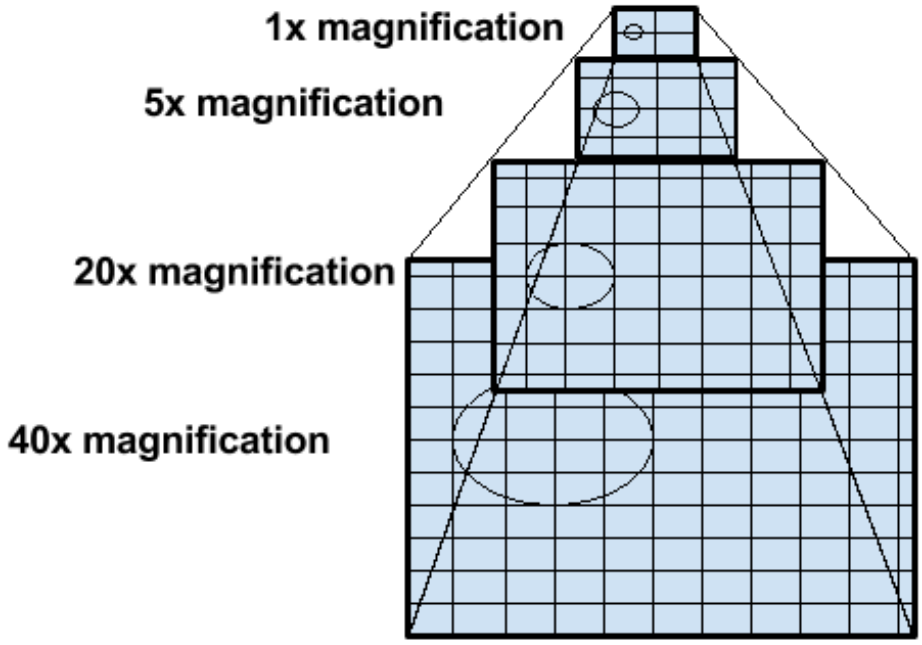

Figure 2.3. *Pyramid image stores duplicates of the image data for different resolutions*

The component that presents the image views is termed the "image viewer component", regardless whether the image processing is performed by the viewer software or done remotely using dedicated image server software. The viewer allows users to view, pan and zoom virtual slides, similar to the effect of manual movement of mounted and stained tissue sections under a microscope and changing magnification with variable microscope objectives [8]. In addition, viewer software can for example also provide features like image rotation, modifying the image via cropping tool, marking regions of interest by annotation tools, measurement tools, synchronized view of two digital slides and connectivity to image analysis functionalities.

Digital slides can be annotated for different reasons, for indicating key diagnostic features in pathology, for indication of regions for microdissection and later extraction of biomolecules such as DNA or RNA from tissue sections, for remote consultation, for clinical review, for using them as a visualization in publications or for educational purposes to name few. Pathologists can use annotations to record diagnostic comments and conclusions by interpreting annotated areas, marked by themselves or by other

pathologists. In manual microscopy, such annotations are often done by placing an ink mark directly on the glass slide, but whole slide imaging enables using digital annotations. Although these ink annotations are marking notable findings, in most situations today, there is no automated link between the annotation and pathologist's interpretation in pathology report [13]. Using digital annotations enables adding this kind of linking.

Another use of annotation comes when sections are cut into smaller sub-sections using laser capture microdissection (LCM). In LCM, a histologist or pathologist annotates the slide and uses the LCM device dissect targeted regions of interest (ROI) using infrared or UV lasers [14, p. 64]. Such dissection can alternatively be performed manually at lower resolution using scalpels, especially when ROI are relatively large. The cutting can also be done in a manual process called macrodissection, in which annotated area is scraped from the slide using scalpel [15, p. 27939]. After extraction, the isolated subsections can then be processed further and DNA or RNA can be extracted from them for example. Regardless of the cutting method, one advantage of using digital slides with the process is that there is a digital copy of the whole slide even after the section is resectioned, and creating a linkage between newly created tissue section sample and original tissue section sample.

Annotations also becomes useful when doing manual or automated analysis of the digital slides. Automated analysis tools are generated for example to automatically detect cancer tumours from digital slides [15]. Another usage of automated analysing is suggesting classifications for annotations based on classifications of visually similar annotations found on the system [16].

Digital slides can be useful by themselves to serve as educational examples illustrating features of interest, but in disease studies they are most useful when they retain linkage to their subject of origin and other data related to the disease under study. Integration could mean for example integrating the whole slide image gathering process with other activities done in the laboratory and integration of the digital slide data processes with existing software solutions used in the laboratory. Integration of the whole slide imaging needs to be carefully planned. Things that needs to be considered are for example the used software systems: Is the whole slide imaging software system going to be separate framework from laboratories regular software systems, or are the two systems going to be integrated as one big system? In practice this could mean whether the data gets stored in the whole slide imaging systems database or in the main systems database, whether the viewer is going to be implemented as part of existing systems frontend or separate application is going to be launched for displaying images and so on. To ease the integration process between different systems, a standard called Digital Imaging and Communications in Medicine (DICOM) can be used when implementing interfaces for the imaging devices and whole slide image viewer frameworks.

2.3 Digital Imaging and Communications in Medicine

The Digital Imaging and Communications in Medicine standard, or DICOM, is a data interchange standard for biomedical imaging, put together to ease communication and integration between different imaging devices and systems. DICOM specifies in detail a non-proprietary message standard, digital image format and file structure for biomedical images and image related information. If the system is implemented using DICOM specifications properly, it can reliably exchange information with another system implementing DICOM. Nowadays DICOM interfaces are available in most diagnostic imaging equipments, which gives imaging system implementers more freedom when selecting the equipment, as proprietary considerations do not need to be taken into account. [17]

DICOM can be used for connecting together any combinations of image acquisition equipment, which could be any devices that uses imaging modalities listed in Table 2.1, image archives, image processing devices and image display workstations such as whole slide image servers and viewers mentioned in previous, hard-copy output devices such as printers. DICOM is not the only standard attempting to specify the medical imaging data interchange. The Health Level Seven, or HL7, also specifies a message model, but abbreviates network communications specification and European Standardization Committees Technical Committees Request and Report Messages for Diagnostic Service Departments document gives only partial guidelines for electronic document interchange. [17]

3. INTEGRATED LIFE SCIENCE

Integrated Life Science Research, or ILSR, is a web-based research application that attempts to ease research work processes done in laboratories and enable better use of produced research data in clinical environments. ILSR's wiki page states: "The purpose of ILSR is to provide a tool for life science and biomedical research laboratories to manage a spectrum of activity from inventory management, to clinical research data management, to integrated lab notebook management, to integration of multiple types of life science data that can be output for analytical processing, publication, collaboration, and direct use in clinical trials. The idea is that ILSR will enable laboratories to do higher quality, more efficient and cost-effective science. Labs will be able to publish new important scientific results better and faster. They will be better able to maintain continuity of their research results and practice over time." [18].

ILSR has many key areas filled with various different features supporting tasks required for running research laboratories and executing research projects. All key areas are listed in Table 3.1. Three of the first main areas, contacts, facility manager and equipment & supplies, are suitable to be used by a laboratory technician or whoever is responsible for managing laboratory's daily tasks. The first area, contacts, can be used for managing laboratory's contact information for individual persons, collaborating laboratories and institutions, equipment and supply manufacturers and suppliers. Contacts area keeps track of addresses, phone numbers and other contact information as well as relations between individual persons and entities. The second area, facility manager, is designed for setting up laboratory facilities layouts by configuring, for example, laboratory's buildings, rooms, refrigerators, freezers, desks, shelves and such to the system. Each of the facility locations are provided with a unique identifier which can be used for tracking the location of individual items in laboratory. The third area, equipment & supplies, can be used for managing laboratory's equipment and supply information, such as their order information, technical details, condition, location in laboratory, ownership and so on.

Table 3.1. *ILSR's key areas*

ILSR Key Area	Purpose	Typical User
Contacts	Manage contact information of people and entities	Laboratory technician
Facility Manager	Manage laboratory's buildings, rooms, freezers, etc.	Laboratory technician
Equipment & Supplies	Manage laboratory's equipment and supply information	Laboratory technician
Subjects	Manage subject information	Laboratory technician, researcher
Biomaterials	Manage biomaterial samples collected and derived from subjects	Laboratory technician, researcher
Experiments	Manage research experiments and laboratory's protocols	Laboratory technician, researcher

ILSR's study subjects and biomaterials areas are designed for keeping track of all the sample information needed for doing research experiments. Study subjects manages subject information such as patient's general information and medical records, as Biomaterials area is designed for keeping track of sample information such as blocks, slides, body fluids, cell lines, tissue microarray blocks and tissue microarray slides. Blocks and slides have the same definition as was used in Section 2.1: blocks are either paraffin embedded or frozen samples collected from the patient and slides are cut sections from those blocks. Body fluids means fluid samples collected from the patient, such as blood, saliva, plasma etc. and cell lines are living cells that can be grown in the laboratory. Tissue microarray blocks are similar to normal paraffin embedded blocks, but instead of containing only one tissue sample, they can contain up to 1000 or more cylindrical tissue samples arrayed into a single paraffin block [19, p. 123]. Tissue microarray slides are sections cut from these blocks and placed onto a glass slide, similarly to the normal slides.

Experiments area can be used by researchers to manage their research experiments as well as by laboratory technicians to manage protocols needed in laboratory's day-to-day operations. Experiments area's main purpose is to manage experiment and protocol information, but it has functionalities for managing tissue reagent and LCM session information as well. Tissue reagents are samples extracted from tissue such as DNA, RNA, protein and cDNA, and LCM session information contains information such as which slide was cut, how much material was collected, the purity of samples, used instrument, laser configuration and so on. Experiment information holds all the work

phases done that were executed, experiment's author and performer, as well as used equipment instances, supply instances, samples and tissue reagents that were used and created during the experiment. The idea behind the experiments is that anyone with required skills and access to right materials could repeat the experiment at any time by using the using information stored in ILSR experiment. In other words, experiments area can be used as a laboratory notebook when running experiments and as a support for creating publications once experiments have been concluded.

One of the essential functionalities of ILSR is that every bit of information needs to be traceable. Unique identifiers are assigned to each of the items in ILSR, such as equipment instances, supply instances, samples, tissue reagents, and they are all labelled with label containing the identifier, 2d barcode, location in the laboratory and some additional clear text information about the item. In ILSR, the same identifiers are used to link information such as equipment and supply instances or tissue reagents into the experiments, block into the tissue reagents and subject into the blocks and so on.

While writing this thesis, ILSR is used for supporting prostate cancer research's day-to-day work at Prostate Cancer Research Center's, or PCRC's, molecular research laboratory located in Tampere. Since ILSR's development started in 2011, it has been used for partially supporting various life sciences related publications [18; 20; 21; 22; 23]. Although prostate cancer research has been ILSR's main use, it can be used for other areas of life science research work as well, and if resources become available, an attempt will be made to make ILSR a truly scalable solution.

3.1 Whole Slide Imaging in ILSR

Before initiation of this thesis work, ILSR supported storing and displaying normal and whole slide images with related metadata through its Image Management Module. Normal images could be displayed using normal web browser image functionalities whereas whole slide images required dedicated viewer application: a closed-source JVS Web Viewer solutions was used for displaying the whole slide images. The existing whole slide imaging feature was mapped to many of the ILSR's current functionalities, and implementing a new annotable whole slide imaging feature would enable enhancing the current functionalities as well as enabling new functionalities. A short description of current functionalities and planned new functionalities that uses whole slide imaging feature are presented in Table 3.2, including description of how enabling annotations to the whole slide imaging feature will enhance the current functionality.

Table 3.2. *Current and planned ILSR's functionalities using whole slide imaging feature*

Functionality	Status	Purpose	Mapping to whole slides	Effect of annotations
Regular slide management	Current functionality	Is used for managing slide information.	Whole slide image of the slide can be stored as part of the slide information.	Better metadata and slide metadata can be linked with more precision
Tissue microarray slide management	Current functionality	Is used for managing tissue microarray slide information.	Whole slide image of the slide can be stored as part of the slide information.	Better image quality for spot images and spots can be linked with more precision.
Surveys	Current functionality	Is used for building and executing surveys for multiple purposes as part of the research experiments.	One or multiple whole slide images can be displayed as part of surveys.	More precision to questions.
LCM management	Current functionality	Is used for managing LCM information.	Not mapped to whole slide images at the moment.	Linkage between annotated areas and extracted samples. ILSR can be used for driving the LCM process.
3D visualization	Planned functionality	Is used for creating a 3D model of a prostate.	Functionality does not exist yet.	Enable building 3D wireframe of prostate and its tumour and metastases.

In current system, the whole slide images can be stored as part of general information of slide and tissue microarray slide and they were used for studying morphology of the cut sections. Whole slide images can also be used as part of surveys in which one or several whole slide images are shown to the survey participant, whose responsibility were answering series of questions related to the image or images. Survey tool can be

used, for example, evaluating the quality of the whole slide images or for evaluating the suitability of the whole slide imaging as a diagnostic method.

Next step in ILSR's development plan is to enable annotating whole slide images. Annotations could be used for example differentiating the tumour from normal tissue and therefore providing better metadata about the slide, as well giving more precision by mapping the metadata information directly to the annotated area. One possible usage could also be displaying annotated images as part of surveys, allowing more precise questions such as "Would you say that the image's annotated area A in contains cancer?". Annotations also enable more tracking precision when tissue is extracted from the slide using scalpel or LCM and could be used to drive the extraction process. After scanning the slide, areas of interest could be annotated and the new sections would be cut from original slide using LCM device, either manually by viewing the whole slide image using display next to LCM device, or ideally automatically using annotation data as input to the LCM device. After the LCM, the whole slide annotations would work as a linkage between the new extracted sub-sections and the original slide, giving the precise tracking information where exactly the tissues were extracted from. This would help preventing situations mentioned in Section 2.2, where pathologist's interpretations were not linked properly to the pathology report, or when doing experiments, cases where the linkage is not clear between extracted DNA or RNA sample and the exact location on slide where it was extracted from.

Tissue microarray data management functionality would also benefit from whole slide annotations. Tissue microarray slides can have hundreds of small tissues sections, or spots, placed on a single slide. Currently ILSR stores regular small images of each spot and scanned image of the whole slide can be stored as whole slide image, but linkage between those two is missing. Enabling annotations on whole slide image would remove the need of separate spot images, ease the tracking of where the spot is located on the slide, simplify maintaining the tissue microarray image data as single image would be sufficient and enable better image quality for the spot images.

One of the purely new functionalities, which does not exist in ILSR yet, is generating 3D model of prostate including its tumour and metastases. This could be done for example by first cutting sections along the whole prostate and scanning whole slide images from the sections. Then sections' edges would be annotated automatically using image processing algorithms and tumour and its metastases would be annotated manually. The annotations would then be used to create 3D wireframe to visualize how prostate cancer tumour and metastases are located spatially inside the prostate. This kind of 3D model would enable studying how cancer evolves inside the prostate and enable even more precision and quality if 3D model is registered with images gathered using other imaging modalities or fusion of combination of modalities mentioned in Table 2.1.

3.2 ILSR Development

Modern ILSR development started in 2011 and the software's development and evolution still continues at the present day under Dr G. S. Bova's supervision. Dr. Bova developed prior applications with differing goals and code-bases as far back as 1998. The prior software was named the PELICAN database. ILSR has gone through several re-factoring processes, as the used technology has become obsolete and development has been executed with different team configurations. Currently it is being gradually upgraded to use up to date software architecture methods including Model-View-Controller (MVC) and Object Oriented Programming, to enable more rapid development and harnessing new technologies with less effort. [18]

The ILSR development process includes several day-to-day tasks, for example development of new features, re-factoring of old legacy code into MVC, issue fixing, database and server maintenance, ILSR system administration and user support to name a few. At the time of writing this thesis, ILSR development team consisted only three persons, so one of the key aspects in ILSR development is to utilize small developer team sizes and in order to accomplish that, communication between team members is encouraged and development tools such as wiki, issue tracking and bi weekly team conference calls are used. ILSR development does not follow any previously specified software development principles as such, but many similarities can be found with seven principles in lean software development, identified by Mary and Tom Poppendieck in their book Implementing Lean Software Development: From Concept to Cash in 2006:

- eliminate waste
- build quality in
- create knowledge
- defer commitment
- deliver fast
- respect people
- optimize the whole [24].

Using principles like these has been found useful among ILSR development process and suitable for utilizing resources of a small a development team.

Eliminating waste

One of the main principles of the lean software development is to eliminate waste. Seven types of waste in software development can be categorized as:

- defects
- extra features
- handoffs
- delays
- partially completed work
- task switching

- unneeded processes [24].

In the lean development, the focus is to prevent **defects** even before they occur, in contrast of traditional software development process, where defects are fixed once found [24]. In many cases, the issue will cause more burden to the team the later it is found, as in addition to only implementing the fix, the problem solving process usually also includes correcting the possible defected data generated by the issue. The National Institute of Standard Technology, or NIST, noted that fixing an issue found on production takes three times more than an issue found on implementation phase and The System Sciences Institute at IBM stated that issue in production costs four to five times more than issue found in design and up to 100 times more if found in maintenance phase [25]. In ILSR development, defect prevention is maintained by planning new features as thoroughly as possible and testing in early phases of development. To avoid any defects ending up in production, new feature goes through series of tests in different servers, first on developers' local environment, then on the sandbox server, followed by testing on the development server and final testing is done on the production server. If an issue is found, the philosophy is to find the root cause of it rather than just fixing it without understanding the wider context, to prevent issue or similar issues happening in the future. ILSR has also utilized automatic database integration check processes to detect any inconsistencies generated by defects in the application and has a plan of implementing automated testing is scheduled in the near future of ILSR development.

Implementing, documenting and maintaining **extra features** that are never or rarely used can be considered a waste of resources. Jim Johnson, chairman of the Standish Group, states that 64 percent of the features in products are rarely or never used when studying four internally developed projects at four companies [26]. ILSR development team works very closely to PCRC's researchers and laboratory technicians, encouraging the end users to give feedback and report any bugs and inconsistencies, in order to identify and understand the need of the new features and to evaluate the importance of old existing features. Every new feature is carefully planned and discussed amongst the team to avoid implementing seldom used extra features. Tools used for preventing extra features ending up to production are conference calls amongst the team members, which end users and collaborators are encouraged to participate, and using issue tracking system as part of the development process. Issue tracking can be used by other team members of the end users to notify developer team about any defects found, but also for suggesting and planning the new features.

Lean development tries to avoid **handoffs**, or excess documentation between implementation phases, as it can become an extra work for the team without creating much of a value. Traditional waterfall process requires detailed documentation between phases such as requirement gathering, planning, implementation and testing. Especially in big projects, big part of team's time is used for creating and interpreting these documents. Excess documentation can also lead to loss of information, if there is not enough time to go through all the details [24]. In ILSR development, only the aspects that are

considered important are documented using issue tracking system or, if information is considered valuable enough amongst the team members, it is recorded into ILSR wiki.

Delays in development should be avoided, if possible. In ILSR development, new developers are gradually trained to understand the whole system and reasons behind the decision making, in order to increase their ability to choose right decision in the future development. However, no-one has all the answers, so communication and question asking are highly encouraged, other team members are always available for consultation and questions are discussed at latest in team conference calls. If decision making requires gathering information from outside the team's resources, the information is recorded using issue tracking or ILSR wiki to avoid delays when facing similar issues in future development.

Features, pieces of code, documentations, bug fixes and items that are **partially completed** should be avoided. In ILSR development team, members aim at finalizing the work they have started and then move to next item in their list, although this is not always possible as all high priority tasks cannot be estimated or expected, for example critical bug fixes. One principle to avoid partially completed work that has been harnessed in ILSR development is to divide the new feature into different smaller implementation stages that takes less time to implement. Every stage of the feature is complete work in the sense that it is usable, but can be extended and improved in the next stage.

Task switching cause delays, as it takes some time for the developer to readjust to the new task, and should be avoided, if possible and they can cause even defects, if developer is assigned to a task without having any previous experience of the task's context. In ILSR development same principle is applied for task switching as was used for avoiding partially completed: tasks are finalized as one continuous flow, if possible. Also, team members that have the best knowledge and possible previous experience of the task's context are assigned for implementing the future modifications of the same area, to avoid delays caused by other team members learning the finest implementation details.

Unneeded processes are avoided in ILSR development by trying to find the right balance between meeting intervals, the amount of documentation, the amount of testing and constantly evaluating the development process. It can be argued if bi weekly meetings are considered as excess processes, but one of the ILSR development team's key principles is that the communication between team members, and therefore meetings, are essential to avoid defects, extra features and delays in development. However, meeting interval is not fixed and it can be changed at any point to adapt the current situation. Documentation is tried to be kept in minimum, only what is considered necessary for understanding the software, for supporting the future development and for maintaining the code base gets recorded. Developers run unit tests after the new feature has been implemented and once the feature is finalized, it goes through integration tests executed on different servers, usually including testing rounds by a different developer and an end-user. Ideally testing is never an unneeded process, as defects can be costly when

ending up to production, but testing requires time that is taken from away from other implementation tasks, so right balance needs to be kept, especially in small-sized team like ILSR development team, which lacks dedicated testers.

Building quality in

The second lean principle, build quality in, talks about fixing the problems as they arise, even if it means stopping the whole assembly line, in order to prevent the problem occurring again [24]. ILSR has same approach, as stated in ILSR development wiki page: "If we find an error in a part of the application that was assumed to be working, we stop and fix it, we don't leave it until tomorrow. If we find code that is full of misspelled words, or does not contain comments, or is poorly organized, we fix it. If tables, attributes, files, or anything else does not have a clear, easy to recognize and understand name, we change it." [18]. One example of power of building quality in becomes from New United Motor Manufacturing, Inc., which told factory workers to stop the assembly line whenever something prevented them from doing their work, which cause delay of one month before first car was produced, but in the end enabled plant becoming a leader in quality and productivity in U.S. [24].

Create Knowledge

Lean development's third principle, create knowledge, encourages usage of already learned information [24]. ILSR development process maintains knowledge base that is constantly evolving using conference calls to share the newly learned knowledge, by recording implementation steps, using issue tracking software, and recording overall documentation of newly build features to wiki. This ensures that new developers and other team members do not need to go through the same learning process again in future when working with the same or similar features.

Defer commitment

The defer commitment principle is very closely tied to previous principle. In development phases, the balance between using the best available information and time to make decision needs to be determined [24]. In ILSR development, the first implementation method is rarely picked, instead the different implementation solutions are evaluated and the knowledge of documentation tools and other members is utilized in the team meetings before the final decision is made.

Deliver fast

Deliver fast principle means delivering the product in small iterations to keep the end users in feedback loop and affect the future development of the feature [24]. ILSR follows staged implementation plans with larger features and plans can be and often are changed depending on the new knowledge and issues found in using the features early stage implementations.

Respect people

ILSR development process tries not only value their team members, but end users as well. Everyone's feedback and work has value in it, and are encouraged to find alternative and better ways to enhance the development process, improve the quality of the application, bring in new technology and communicate with the development team to find the best practices implementing new features and maintaining existing ones. One tool to respect people in ILSR development is by immediately acting on issue an end user has reported and involving the user to follow the bug fixing process via issue tracking tool. When user is involved like this, he or she sees that his or her opinion matters and there is a method for him or her to improve the system.

Optimize the whole

Final principle, optimize the whole, talks about the importance of keeping the big picture in mind, whether you are implementing a feature or optimizing a development process [24]. ILSR development attempts to include this as part of the development process by gradually training new developers and end users to understand the whole system and real-life tasks done in laboratory related to them. That is why developers are physically located near the end users in the PCRC's molecular research laboratory in Tampere, if possible. This helps developers to understand what laboratory's day-to-day work contains, and end users are included into software development process to understand the software developers' side of the software development. Also, the reasons behind why things are done, instead of just telling how things are done, are tried to be emphasized when consulting and training new people to work with ILSR.

4. EVALUATION METHODS

The recently increased availability and popularity of ready-made customizable software components has caused a need for creating methods for evaluating the available solutions. The organization might have many different reasons for picking up a ready-made solution, for example, to save development time, to get more secure or better performing solutions that would not be possible to develop with organizations own resources, to get well reviewed and widely accepted solutions or to outsource the development to focus on other areas of development process. There are several methods for evaluating software components for the needs of solving business problems available, two of them are presented in this thesis. After the existing methods have been introduced, a new evaluation method is introduced. The new method has similarities to existing methods, but attempts to focus not only for evaluating the right software solution for the problem, but also for evaluating the right implementation approach suitable for small development team.

The first method is David Wheeler's Identify Review Compare Analyze (IRCA) method for evaluating open-source software in his paper How to Evaluate Open Source Software / Free Software (OSS/FS) Programs [27]. Wheeler first searches information about candidates in the identify step, followed by reading others evaluations of the software from reviews in the review step. In the compare step, Wheeler briefly compares solutions and narrows selection to the top candidates that are finally evaluated in more depth in the final analyze step to find the solution that best suits the needs of solving the initial business problem. The second method, created by The Software Engineering Institute (SEI) of Carnegie Mellon University, uses scoring system in order to provide a formal evaluation process and it is designed for evaluating commercial closed-source software [28].

4.1 Available Evaluation Methods

Regardless the used evaluation method, the process can be divided to three sequent steps. The first evaluation step is the identify step, in which possible software candidates are listed down. The next step is the reduce step, where all the candidates are evaluated in one or several evaluation rounds in order to narrow down the selection to few top candidates. The final decision is done in the third step, where best suitable solution is picked from the top candidates.

4.1.1 First step - Identify

The IRCA method suggests using combination of techniques for gaining the best coverage, and Wheeler lists down methods such as asking from friends and co-workers, by running searches from sites tracking OSS/FS programs, using general search tools such as Google, using search engines that are focused on context of initial business problem and trying to find if Linux distributions already have included suitable software [27]. Wheeler suggests avoiding search engines with conflicts of interests, for example search engines that are hosted by the same company that provides suitable solutions, as they are unlikely going to help finding information about their competitors [27]. He also suggests trying various search terms and searching with combination of known solutions to find pages listing down or comparing similar products [27].

SEI's method suggests using selection team for identifying candidates, in order to eliminate single-person perspective from the equation. The selection team would consists of technical experts including systems and software engineers and several developers when selecting software components, building blocks for larger system, and inclusion of business domain experts and potential end users to the selection team would be useful when selecting larger software systems. SEI's method also lists down several approaches that could be used for the identifying process: vendor surveys, vendor white papers, technical specifications, representation at the conferences, communication with other customers and conducting a pre-bid conference. The first approach includes vendors to the identification process by allowing them to rate their own products based on the suitability for solving the business problem. The next two approaches, vendor white papers and technical specifications, relies on using product's documentation in identifying process. The fourth and the fifth approaches suggest contacting vendors, other users and companies providing support for the product either in conferences or using other available methods. The last approach suggest organizing an event allowing possible vendors to visit the organization and discuss their products with the evaluation team [28].

4.1.2 Second step – Reduce

Once the software candidates have been listed, the number of candidates is reduced by initial evaluation. The IRCA method relies on narrowing down the selection in two steps: first by reading existing reviews about the candidates to narrow down the selection to the leading candidates, and then briefly comparing the leading software's attributes to the needs of solving the business problem. First, the reviews need to be found and Wheeler suggests searching reviews using general search engines and searching OSS/FS Content Management Systems and OSS/FS Software Management Systems. Wheeler reminds that many reviews might be biased as magazines are funded by advertisements and some systems allow anyone to rate software. Still, searching for reviews might lead to identifying new candidates and a review might reveal aspects of software that the development team was not considering before. Also, reading and searching re-

views gives indication of product's market share, which is important, as products with high market share usually have better sustainability and provides better support. [27]

The review step is followed by the compare step in IRCA method. The idea is to briefly evaluate the leading candidates against business problem's needs in order to quickly narrow down the selection to few top candidates. The IRCA method suggests using software projects web site and documentation as source of information. Wheeler lists down 13 important attributes that should be used as base to the evaluation process: functionality, cost, market share, support, maintenance, reliability, performance, scalability, usability, security, flexibility/customi zability, interoperability and legal/license issues. [27]

The **functionality attribute** is used for evaluating if the software solves the business problem or not, how well it integrates with existing components, does the software use relevant standards, and what hardware and operation system setup are required. Few software provides all functionalities, but often it is possible to modify the software to fulfil the missing functionalities. The **cost attribute** takes into account possible initial license fees, installation costs, staffing costs, support costs, transition costs, software and hardware upgrade costs that comes with adapting the new software. The **market share attribute** tells about how widely software is adapted giving indication of projects sustainability and availability of support from other users. The **support attribute** takes into account how easily users can be trained to use the software, how easy it is to install the software and how easy it is to find answers to users who have specific problems with the software. Usually support aspect of open-source software can be evaluated by studying the documentation it provides, but paid support can be also provided or support can be provided by the development and user community as well. [27]

IRCA method's **maintenance attribute** tells how easily software can be modified and managed. This is important as software tend to evaluate over time, seldom are completely static. The maintenance can be studied by following project's developer's mailing lists and version management information, if available. The **reliability attribute** tells how reliably the program solves the business problem, and is best to be tested on real work load, as well as the **performance attribute**. The **scalability attribute** describes software's ability to adjust to bigger size of data or problem. The **usability attribute** is difficult to measure, but important attribute, as it tells how easily the software can be learned and operated by the users. The **security attribute** is best to be evaluated by first gathering exact security requirements and then comparing software's security features against it. The **flexibility attribute** measures how well software can be used to solve unusual business problems that differ from its original design and needs to be evaluated by the software's suitability to solve the original business problem. The **interoperability attribute** tells how the software connects to bigger software system and how easily it can be replaced with another similar solution if that is required in the future. The last attribute, **legal/license issues attribute** should be examined carefully for all the suitable software, as it states for what and how the software can be used legally. [27]

Metcalfe is listing down a set of 11 tips, in which 9 are overlapping with IRCA's set of attributes, but he also introduces two new aspects: skill set and project development model [29]. The **skill set tip** takes into consideration if the development team have enough skills needed to deploy and maintain the software, or is third party contractors or training plan required as part of the adaption process. The **project development model tip** talks about how well the software project development process is managed: How contributions are made and how they are evaluated for inclusion?

SEI's method uses three different main criteria for the evaluation process: functional requirements, intangible factors, and risk. The functional requirements are considered as an important first step of evaluation but not should be used as only criterion. **Intangible factors** includes programmatic decisions that have an effect on the system utilizing the software, but are not the traditional quantifiable factors. The intangible factors attempts to find answers to questions such as:

- Does software require specialized language training or techniques?
- Is the adapting organization's business processes subject to a change?
- Is the software used only by one area of the system or re-used by many areas?
- Is the solution an overkill to the problem?
- Will adapting the software require training for the end-users?
- How well the software integrates with other software, is modifications needed to change the software's interface?
- What kind of support and documentation are available?
- Are all the costs known up front?
- Does the integrated end product require special skill set to be operated and maintained?

Some of the intangible factors overlaps the IRCA method's list of attributes, but some provide new aspects as well, for example by taking into account adapting organization's future plans, usage of the component in the system and consideration if the software providers only the solution to the current business problem or is it intendent for wider usage. [28]

The SEI's method risk criterion attempts to consider the risk organization is taking when adapting vendor's software product. The method lists down some possible risk factors that should be considered:

- Is the company well established?
- What is the longevity of the company?
- Is support offered?
- Is the vendor flexible to make changes to the software?
- Is the vendor financially stable?
- How mature is the technology used?

The SEI's method also states that the risks should not only be considered as part of evaluation process, but continuous risk management should be applied throughout the whole life cycle of the system that uses it. [28]

Once all the functional requirements, intangible factors and risk factors are listed, next step of SEI's method is evaluating the candidates using scoring system. Each of the selection criteria items are listed in decision analysis spreadsheet, a percentage weight value is given to the item so that the sum of all items add up to 100% and then each item is given scoring value ranging from 1.0 to -1.0 in increments of 0.5. The solution getting the highest score is considered as the preferred solution, although when risks are evaluated, the solution getting the lowest score is considered as best, and therefore risk evaluation should be done in separate process. [28]. SEI's method does not consider how many iteration rounds evaluation requires, but there is no reason why the scoring mechanism cannot be used to evaluate all the candidates by using only some of the most important criteria in the reduce step and in order to narrow the selection down.

4.1.3 Third step – Final decision

After the reduce step, there should be only few top candidates left for more thoroughly evaluation. The IRCA method executes this process in its analysis step. Wheeler suggests of getting the software instances and testing them using the same list of attributes as in previous step, instead of relying on documentation, in order to find out the solution that best fits solving the original business problem [27]. If all the required features are not supported, it should be examined what it takes to implement them, by studying software's design document and source code base [27]. If the selection was narrowed down in reduce step using SEI's method, the remaining candidates can be evaluated in final decision step by scoring them using full set of criteria and risks.

4.2 New Evaluation Method

Both of the evaluation methods, IRCA and SEI's method, include useful evaluation processes, but they lack some aspects that are considered useful for small development team's needs. Neither of the evaluation methods takes into account the workload needed for implementing a new framework. For example, there are several ways of implementing a whole slide imaging feature to a larger system using ready-made customizable software solutions: the developer team can implement a whole slide imaging framework almost from scratch using low level libraries, the team can adapt an open-source framework, or the team can purchase a commercial solution. Each of these implementation approaches requires different amount of resources from the developer team, and it is not always obvious which one of the approaches is most suitable, until the available solutions have been evaluated. This thesis presents a new evaluation method which has many similarities to evaluation methods presented in this thesis, but is especially suitable for small development teams as it takes into account evaluating the implementation

approach together with the software solutions, to better adjust the selection to small developer team's needs.

Another reason for choosing an alternative evaluation method is that the new method can be used for evaluating both: the open-source and the closed-source solutions, whereas IRCA method is designed only for open-source solutions and SEI's method is used for commercial products. The third reason is adding flexibility to the method in order to avoid extra work. IRCA method lists down a set of attributes used for evaluation process, as the new method includes set of requirements that are derived from the business problem and existing systems environment, evaluating only the requirements that are found important. The new method evaluates only the amount of requirements that are needed to discard the solution, as SEI's method uses decision analysis spreadsheets that are used for counting the total score for each of the solutions to find the best suitable solution. SEI's method also talks about using big selection teams and conducting a pre-bid conferences for vendors, methods which are often not possible for small developer teams.

The evaluation method used in this thesis can be described as three different processes: requirements gathering, evaluating different implementation approaches and evaluating available software solutions. Four different implementation approaches are introduced in this thesis, each requiring different amount of development work done by the development team. In this thesis, the term *available software solutions* in this thesis is defined as any third-party library, framework or customizable ready-made software solution that can be used for implementing a new feature.

Gathering requirements

Unlike IRCA method's set of evaluation attributes, the new method does not use fixed list of requirements as a base for evaluation, but instead relies on gathering most important requirements from the environment where the solution will be used, similarly to the SEI method's approach. Gathering requirements is an important development phase, as defects in early phases can become much more expensive to fix in later phases, as mentioned in Section 3.2. There is an estimation that up to 85 percent of defects are originated from the requirements, and once they have been embedded, they become difficult to find, especially via testing [30, p. 9]. Two most common errors in requirement gathering phase are incorrect assumptions and omitted requirements [30, p. 9]. In ILSR development, defect prevention practices mentioned in Section 3.2, like working closely with the end-users, good knowledge of the existing system and its environment and good communication inside the development team, reduces the number of wrong assumptions and helps including the right requirements.

Evaluation of Implementation Approaches

Four different implementation approaches are introduced in this thesis. The different approaches are characterized based on the balance between implementation work done by the development team and implementation work already done by the solution pro-

vider. The evaluation summary of different implementation approaches are listed in Table 4.1.

Table 4.1. *Implementation approaches*

Implementation Approach	Workload	Advantages	Disadvantages
Implementation from scratch	Low level libraries are given, rest needs to be implemented.	Developer team has wide set of options how to implement the framework.	Requires lots of resources and additional special skills.
Implementation using core solution	Core solutions are given but might need modification, feature sets needs to be implemented.	Special skills are not required and developer team has fair set of options how to implement the framework's features.	Requires fair amount of resources to implement feature set and to possibly modify the core.
Implementation using framework	Whole framework is given but might need modifications.	Developer team needs relatively small amount of resources to learn framework and make modifications.	Developer team's options are limited by the framework's implementation.
Integration with off-the-shelf framework	Whole framework is given, development team communicates with the framework provider for possible changes.	Framework's development does not burden development team.	Framework's provider decides how the new features are implemented.

The first implementation approach, **implementation from scratch**, usually requires most resources from the developer team, possibly including special skillsets, although in some projects building a trivial software component from the scratch might be the fastest solution. In this approach, the developer team implements the whole framework starting from core functionalities using only basic low-level libraries. Feature sets will be implemented on top of these core elements to complete the frameworks functionality. This kind of approach is usually preferable or mandatory if there are no suitable solutions available or if software component is so essential for the system that development of it is required to be kept within the developer team.

In lean development process, the first implementation approach would eliminate waste as no defects, partially completed work or extra features would be inherited by

adopting a ready-made project. On the other hand, re-implementing the framework if suitable ready-made solutions are available could be seen as unneeded process. The first approach also enables building quality in, which is the second lean principle, and enables following the "optimize as whole" principle. As developer team is implementing the whole framework from the beginning, it enables implementation that can be done following the team's standards and wider knowledge of the new feature's context.

The second approach, **implementation using core solutions**, eases the implementation burden from the development team when compared to the first approach, as ready-made core elements can be used, although they might need modification. Developer team's responsibility would be implementing the feature sets around the core elements to complete the framework. This approach's requirement for resources is dependent on how easily the core elements can be extended and how complex the designed set of features is. It might be a preferable option if suitable core elements are available but suitable complete framework solutions are missing. The benefit, compared to the first approach, in addition to resources saved on implementing the core elements, would be that special skillsets would not be necessary required, depending on how much the core elements needs to be modified.

Similarly to the first approach, the second approach would also avoid waste as no defects, partially completed work or extra features would be inherited from the framework implementation, but at the same time the same waste types could be inherited from the adapted core elements. However, the core elements might be complex and require specialization, so inheriting tested solution implemented with experience developer team can actually eliminate waste, if compared to the approach where developer team attempts to implement the same solution with less implementation skills and/or testing capabilities. The disadvantage of the second approach is that the quality cannot be built in and the big picture cannot be kept in mind in implementation of core elements.

The third approach, **implementation using framework**, is similar to the second approach, but instead of adopting only the core elements, the whole framework is adopted. This approach usually requires the least resources from the development team out of the three first approaches, as developer team's only responsibility is to modify the framework if needed. Another benefit is that more complex feature sets can be inherited that would be possible to be implemented by small developer team. The disadvantage of this approach, when compared to the two first approaches, is that developer team is limited by the designs picked up by the framework's developer team, although usually they can be modified to some extent. Another disadvantage is that the developer team takes risk when adopting ready-made solution, such as the risk of code base containing defects, the risk of framework using obsolete technologies and the risk of project being poorly maintained. The implementation approaches is tightly linked to the questions of SEI's method: "What is the longevity of the company?", "Is support offered?" and "How mature is the technology used?". Although SEI's method is designed for commercial products, they apply to open-source frameworks as well. If the original software's development team stops the project, it is developer team's responsibility to con-

tinue development and management of the project. Also, if there are no good support available, management becomes difficult. However, if suitable solutions are available and risks does not seem too high, this approach is usually most preferable especially with the small development teams.

The fourth approach, **integration with off-the-shelf framework**, is opposite of the first approach in the sense that it does not require any development from the developer team, as framework's provider is responsible for developing the framework. Developer team's only responsibility is to integrate the ready-made solution with the host system, and possibly request new features from the framework's provider. The benefit is that this approach does not burden the developer team by any implementation work, but the disadvantage is that the framework provider has their own interest in framework's development plan and implementation of the framework's new features, modification of existing features and fixing defects is their responsibility. Adapting ready-made system can cause trouble later, if conflict of interests occur between framework provider and framework user. The risks described in SEI method can be directly applied to this method. However, this is still often preferable solution especially for small developer teams, as it releases lots of resources to other development processes.

Both the third and fourth implementation approaches need to be carefully evaluated when choosing the right implementation approach. As they tend to avoid the lean development processes waste by eliminating unneeded processes, by avoiding the implementation of the framework, the inherited project can also come with lots of waste in form of defects, extra features and partially completed work. At the same time, if framework's original purpose differs lot from intended use, the fact that the feature's context was not kept in mind while building the framework causes risks. However, carefully selected framework might avoid all of these issues, as skilfully implemented and tested framework implemented for correct purpose might bring in quality that could not be achieved if the framework would been implemented by the developer team with insufficient resources and knowhow.

The evaluation of implementation approaches and available solutions are parallel processes, as the decision of right implementation approach is dependent on the available solutions. For example if method of using ready-made framework is picked, but later it is found out that there are no good framework candidates available, implementation approach needs to be re-evaluated. However, if it is clear in early phase that some of the implementation approaches can be eliminated, the lean development process waste can be avoided by limiting the scope of available solutions, as those do not need to be evaluated any further.

Evaluation of Available Solutions

The evaluation process is executed using the same three sequential steps presented in Section 4.1: identify, reduce and final decision steps. In the **identify step**, information about different solutions are gathered and categorized based on the suitability for different implementation approaches into four categories. In the **reduce step**, found solutions

are narrowed down by evaluating them based on the gathered requirements. However, all the solutions do not need to be evaluated based on all of the requirements: to avoid lean development processes waste, the solution can be discarded from the evaluation process once enough information is found to back up the decision. Also some of the requirements might be suitable only for the open-source solutions and cannot used for evaluating closed-source solutions. In the last step, **final decision step**, remaining top candidate solutions are compared against each other and best suitable solution for implementing the required feature is picked. The right implementation approach might be selected during any of the evaluation steps, based on the number and quality of available solutions.

5. FRAMEWORK EVALUATION

As described in Chapter 3, annotations supporting whole slide imaging feature which can be rapidly modified is required in ILSR in order for enabling future development of new features as well as enhancing existing ones. The evaluation method described in Section 4.2 was used to first gathering requirements for the new feature, which acted as requirements for the whole slide imaging framework at the same time, whether it was implemented by the development team, adapted open-source framework, or purchased third-party solution. After requirements gathering, different implementation approaches suitability for implementing whole slide imaging feature were considered. Then available whole slide imaging software solutions were searched and categorized followed by evaluation of found whole slide imaging software solutions and implementation approaches, finally leading to the selection of the most suitable solution and implementation approach for the needs of the ILSR development.

The fact that ILSR had a whole slide image viewer feature already implemented eased the requirements gathering process, as some of the requirements were already gathered. The adapted requirements only needed re-evaluation in relation to the new requirement of whole slide image annotation and future ILSR development plans enabled by annotations. The existing whole slide image viewer worked also as a benchmark when evaluating the new whole slide imaging solutions in fields of security, usability, performance and adaption.

5.1 Requirements Gathering

Requirements for the whole slide imaging framework implementing the feature are categorized by the importance into two categories: either the requirement is primary or secondary. The new framework is required to meet all the primary requirements, or needs to be modifiable to meet them with relatively small effort. The secondary requirements are not mandatory, but meeting them adds value to the framework. The requirements are defined in Table 5.1.

Table 5.1. *Whole slide imaging framework requirements*

Requirement	Importance	Description
Web-based	Primary requirement	Whole slide images needs to viewed and annotated by using web browser.
Image format support	Primary requirement	JPEG2000 image format needs to be supported.
Security	Primary requirement	Secure encrypted connection is required.
Usability	Primary requirement	Viewer and annotation tools needs to be easy to use.
Maintainability	Primary requirement	Code base needs to be easily modifiable.
Scalability	Primary requirement	Framework needs to support scaling.
Performance	Primary requirement	Whole slide images should be viewed and zoomed without noticeable lag.
Licence	Primary requirement	Licence needs to allow usage of the framework as part of a commercial product.
Adaptation	Secondary requirement	Integration with framework needs to be easy.
Extensibility	Secondary requirement	Framework needs to enable adding new extensions easily.

ILSR is a **web-based** application, therefore the whole slide imaging feature needs to provide a web-based viewer and annotation tools that can be accessed using modern web-browsers such as Google Chrome, Mozilla Firefox and Internet Explorer. The viewer needs to be working without installation of external browser plugins and cannot require any other efforts, such as changing browsers settings, from the users. Using web-based approach eases bringing in new collaborators, such as laboratories, to use ILSR system. When web-browser is used for accessing the application, there are no needs for delivering and installing the application or new versions of it to the user's devices. This is especially useful for small development team, as a new version of the software needs only to be deployed to the server and not to every user's desktop. Web-based application also does not limit the usage to specific operating systems or even to desktops, as mobile devices such as cell phones and tablets can be used as well.

While writing this thesis, ILSR **supports JPEG2000** whole slide images acquired by the whole slide scanner used in PCRC's Tampere laboratory, and it is a requirement for the new framework as well. Direct support of the file format removes the image conversion step that could possibly cause image compression and loss of image quality. Losing image quality can later affect usage of analyzing algorithms for the whole slide image. Another reason to avoid conversions would be that it causes resource waste as effort from either ILSR's maintainers and/or from ILSR's users is required. If effort is required from ILSR's users, it usually also means that extra effort from ILSR's user support would be required as well. Framework's support for additional image formats and possibility to extend to other image formats are considered valuable, even if they are implemented using embedded conversion tools, as scanner technologies in the laboratory environment can change. Another reason for supporting wider range of different formats comes from the plans of scaling ILSR to collaborating with other laboratories, which come with their own imaging technologies and their own file formats that needs to be supported.

The **security requirement** is equivalent with IRCA method's security attribute, and similarly to what the method suggested, the ILSR's security requirements were used as a requirement for the new feature. It is essential that all confidential data in ILSR is encrypted and user access is controlled using latest technologies, and that applies also to the whole slide imaging framework. Connection from ILSR to framework needs to be authenticated and secured, for example using https protocol. If framework implements own user authentication methods, they need to be integrable with ILSR's security system, so that the usage becomes seamless for the user. For example, the user should not be required to remember additional credentials for using the whole slide imaging framework, see any other login screens that ILSR's main login screen, or even be aware that anything like that exists.

The **usability requirement** is also equivalent with IRCA method's usability attribute. ILSR's users have different backgrounds with varying computer skills, so whole slide viewer and annotation tools should be easy to use effectively. Drawing an annotation with mouse needs to be precise but at the same time be fun to do. If drawing an annotation is difficult or loading the image data while zooming and panning takes too long, user can become frustrated which can weaken the quality of the annotations. In worst case, the poor quality of annotations could affect the quality of extracted DNA or RNA, if annotations are used for driving LCM process, as mentioned in Section 2.2. Usability also affects to the adaptation of the new feature and if feature is not used, it can be considered as a waste according to lean development as mentioned in Section 3.2. Furthermore, bad user experience also burdens ILSR's user support team. Although ILSR whole slide imaging feature is currently used only on desktop computers via mouse, usage of viewer on mobile devices and touch screen devices should also be taken into consideration when evaluating viewer's usability.

Framework maintainability and adaptation requirements are also directly linkable to the IRCA method, but also to Metcalfe's skill set tip and SEI method's questions

"Does software require specialized language training or techniques?", "How well the software integrates with other software, is modifications needed to change the software's interface?" and "What kind of support and documentation are available?". These requirements have the most direct effect on the development team's immediate workload. Constant evolution is important for any software, including ILSR and its whole slide imaging feature. The effort needed for implementing a feature is reduced by the maintainability of the code base. Several items affects the maintainability of the code base, such as used software technologies and software architectural patterns, documentation, active community or ecosystem around the solution and used code conventions. Used software technologies include the main programming languages and frameworks that are used for implementing the framework, which often can also dictate the used architectural patterns. If the framework meets other requirements, but the code base contains lots of unstructured source files, extra effort is needed to make the modifications or to refactor the code base before making the modifications. Good documentation always helps to understand what previous developers have meant with their implementation solutions and can consists of for example code comments, API documentations and wiki pages. Active community is usually valuable tool to leverage when documentation fails to explain specific implementation details. Active community can be formed around the solution, but also around the used frameworks and programming languages, therefore usage of widely used known frameworks and programming languages usually increases maintainability. Code conventions probably has less effect on frameworks maintainability than for example used architecture, but is still a good practice to maintain and makes it faster for the developer to interpret the source code.

Framework adaptation is often intertwined with the code maintainability, but not always. For example, if framework is abstracted using very clear and well documented API but the code base itself is implemented poorly, less effort is required in adapting the framework, but maintaining the framework can become difficult. If, however, the solution itself does not provide API, it is often developer team's task to implement some layer between main system and the solution to be integrated, and then same qualities that affect the framework maintainability affect the adaptation. Picking framework that uses already familiar software solutions for the development team, active user community and availability of good API documentation decreases the effort required for adapting the framework.

Scaling requirement is equivalent to IRCA method's scaling attribute and links to SEI method's question "Is the adapting organizations business processes subject to a change?". The ILSR's development plan includes scaling to support collaborating laboratories and moving towards using cloud technologies as part of the back-end solution. Therefore, also the whole slide imaging framework needs to support, or be modifiable, for being cloud ready. Processing whole slide images requires a lot of computational performance when compared to other areas of the ILSR application and storing whole slide images require lots of storage space when compared to regular images. Therefore, the framework causes risk of becoming a bottleneck in ILSR scaling plans in

future, if it does not support or is not modifiable for supporting computational and data storage scaling.

License requirement has also its equivalent in IRCA method's license attribute and is linked to SEI method's "Is the adapting organizations business processes subject to a change?" as it is tightly related to the ILSR scaling process. ILSR's business model needs to be fixed before bringing in new collaborators. The ILSR's final software license was yet to be decided while writing this thesis, so the framework's license should allow usage of the framework as part of commercial product, to keep the ILSR's licensing possibilities as open as possible.

The **extensibility requirement** in this context means possibility to extend the whole slide imaging framework to support new functionalities, either by adding new plugins or modules to the existing framework, or by making a connection to external systems. Extensibility features that are valuable for ILSR are features supporting ILSR's future whole slide imaging development plans mentioned in Section 3.1, such as support for image analysis tools, connectivity to LCM and other external systems and support for 3d modelling. In practice the connectivity to external systems could be implemented using known image streaming protocols such as JPEG2000 Interactive Protocol (JPIP), Internet Imaging Protocol (IIP) and International Image Interoperability Framework (IIIF) to deliver the image data for external applications. The advantage of the JPIP is that it is a streaming protocol designed for JPEG2000 images and supported by the DICOM [31]. Although JPIP has advantage when connecting with other DICOM systems, its limitation is that it is designed only for JPEG2000 images, as other protocols are file format agnostic.

5.2 Evaluation of Implementation Approaches

Once the requirements were gathered, the different implementation approaches suitability for implementing the whole slide imaging feature were considered. In whole slide imaging framework, the libraries mentioned in **implementation from scratch approaches** description, would include image decoding and network protocol libraries for example. Core solutions would mean image viewer for displaying the image and image server for processing the image and serving parts of image to the viewer. Although the viewer could be responsible of the image processing as well. Feature sets would mean any additional frameworks features, such as annotating images, adding security and supporting image analysis. The developer team would need at least some specialization in the areas of image processing, optimization and network protocols if first implementation approach would be chosen, which was unlikely as it would probably burden the ILSR development team too much.

The second approach, **implementation using core solutions approach**, would be useful, as ILSR development team would not need to develop complex whole slide image server and viewer components. The core solutions would need careful evaluation

though, as they might need to be modified in order to support, for example, security requirements and annotation tools.

The disadvantage of the second approach was that the lean development process quality cannot be built in and the big picture cannot be kept in mind in implementation of core elements as they are inherited, but that might not be so crucial in whole slide imaging feature's case. Whole slide images do not have specific characteristics that differ greatly from any other big sized images, so core elements displaying big images in general does not differ much from core elements used for displaying whole slide images. Therefore, it does not matter so much if the whole slide imaging context is not build into the core elements.

The third approach, **implementation using framework**, seemed very suitable approach for ILSR development team's needs, as this approach would require team to customize the open-source framework for ILSR's needs and integrating the framework with ILSR system, but all the implementation work would been completed already. Required modifications could mean activities such as adding enabling security features to the framework, modifying the interface for integration, migrating the frameworks databases with ILSR's databases and modifying the annotation tools.

The fourth approach, **integration with off-the-shelf framework**, is the approach that was previously chosen for implementing the existing whole slide imaging feature in ILSR prior to work for this thesis. The obvious advantage of this approach is that no development of the framework would be required from the ILSR development team, as everything is handled by the framework's provider. At the same time that is the biggest disadvantage: How fast can the framework's provider adjust the framework for ILSR's development needs?

5.3 Evaluation of Available Solutions

First step - Identify

Several different whole slide viewer solutions were evaluated for their suitability for being the solution used for implementing the ILSR's annotable whole slide imaging feature. In the first identify step, the information about different solutions were gathered and categorized based on the suitability for different implementation approaches into four categories. The found solutions are presented in **Figure 5.1** with their relations to each other for those solutions in which relation information was available. In the next reduce step, solutions were evaluated based on the requirements listed in Table **5.1**. In the last decision step, solutions were compared against each other, and the best solution for implementing the ILSR whole slide imaging feature was chosen.

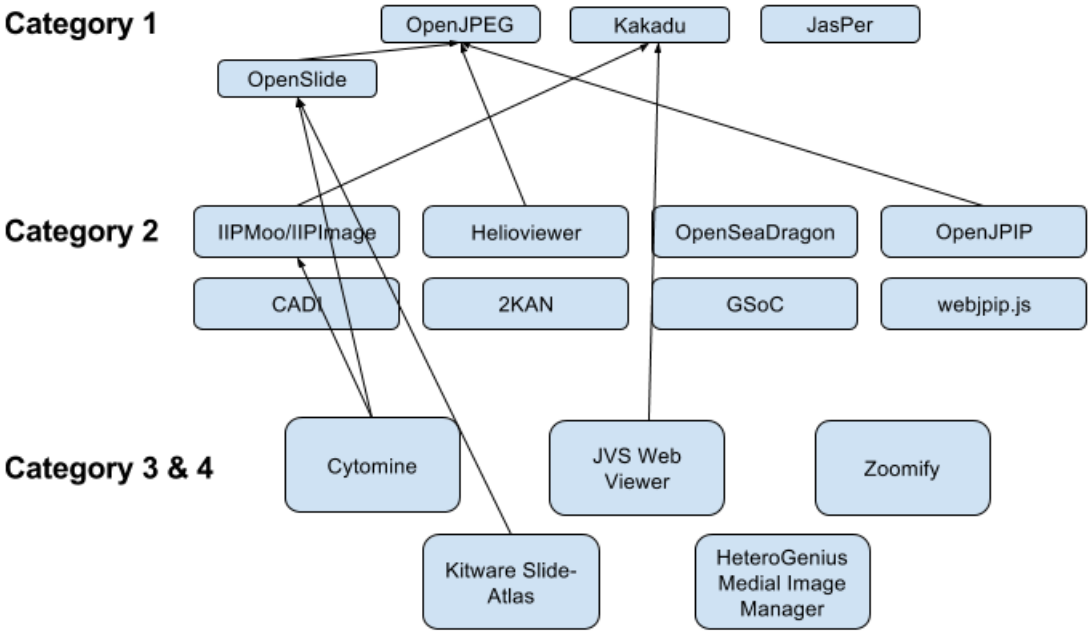

Figure 5.1. *Available solutions categorized based on their suitability for the implementation approach*

In the identify step, information of different solutions were gathered using regular web-searches and reviews from the publications and tools such as Google, Google Scholar and NCBI search were used. Any information about individual tools, core solutions and frameworks intended for displaying whole slide images were searched. The main criteria for the solutions was that they would be able to display JPEG2000 images, the format used in ILSR currently, and/or they would otherwise provide features that would be useful for implementing the ILSR whole slide imaging feature. The information was acquired from online documentation, publications and in some cases by contacting the developer teams for more specific questions. The information was then used for the evaluation process at the next step.

Second step – Reduce

The reduce step started with **first category's solutions**. The first category contains three different codec libraries and one whole slide image library: JasPer, Kakadu and OpenJPEG, which are used for decoding JPEG2000 images, and OpenSlide library, which has support for various different whole slide image file formats and has dependency to OpenJPEG images [32; 33; 34; 35]. Comparison done by William Palmer et al. suggests that out of these three codecs, when comparing the decoding speed, Kakadu is fastest and OpenJPEG slowest and when comparing decoding quality, OpenJPEG and Kakadu give slightly better results than JasPer [36]. Out of these four libraries, Kakadu is licensed under commercial license and others have open-source licenses. Any of the libraries would be suitable for building whole slide imaging core elements around them. However, when evaluating solutions from other categories, it became evident that whole

slide image frameworks are complex systems, and building one from the scratch would burden a small development team too much, as it requires lots of resources and specialization. The first implementation approach including first categories solutions were discarded, narrowing down the selection from 17 down to 13.

Second category's solutions were evaluated next. The second category contains eight whole slide image viewer solutions: Helioviewer, IIPMooViewer, OpenSeaDragon, OpenJPIP, CADI, 2KAN, GSoc and webjpip.js. All solutions provided core elements required for displaying the whole slide images. Often the solutions were divided into two parts where image server processed the image and served parts of the images to the image viewer, which was responsible for displaying the image to the user. Five of this categories solutions, OpenJPIP, CADI, 2KAN, GSoc and webjpip.js, were considered outdated and were discarded, as there were no indication of any recent development on the projects [37; 38; 39; 40; 41]. The number of selections was narrowed down from 13 to 8. Adapting outdated project was considered to be a big risk, as it would likely mean re-factoring of the whole code base to bring it up to date. Also lack of active community's support could become an issue when trying to fix defects. Four of these projects, OpenJPIP, CADI, 2KAN and GSoc, were also desktop projects and converting them to web-based would require too much resources from a small development team. The three remaining solutions, Helioviewer, IIPMooViewer and OpenSeaDragon, were considered as more promising options for building the whole slide imaging framework.

Helioviewer is an open-source project that "aims to enable exploration of the Sun and the inner heliosphere for everyone, everywhere via intuitive interfaces and novel technology" according to Helioviewer Project wiki [42]. Project is funded by National Aeronautics and Space Administration, or NASA, with European Space Agency, ESA. Open-source applications JHelioviewer and Helioviewer.org are implemented as part of the Helioviewer project in order to visualize the solar physics data by displaying high-resolution JPEG2000 images of the sun. JHelioviewer consists of two parts: JHelioviewer client is a desktop application for displaying the images and JHelioviewer is a server for streaming the image data for the JHelioviewer client using JPIP protocol. Helioviewer.org is parallel web-application project to the JHelioviewer. Although Helioviewer.org is extensive framework designed for displaying solar physics data, it is categorized as category two solutions based on lack of whole slide annotation tools.

Helioviewer.org seemed like a good candidate for building the ILSR whole slide imaging framework as it had many advantages. Helioviewer.org is a pure web-based solution, so it is suitable for ILSR's needs as it natively supports JPEG2000 images, which were the two first requirements for the framework presented in Table 5.1. Maintainability requirements were also met as project does have continuity, which would be an advantage as it means that the viewer will be updated and maintained, the project has a good online documentation containing tutorials for users and developers and the developer team responded fast when contacted. The code-base is also well maintained and bug-tracker and coding standards were used. Helioviewer.org also has some extensibil-

ity options, for example connectivity to the YouTube, which might be useful feature in ILSR's future development. The disadvantages included expected difficulties with adaptation. According to Müller et al, the solution could be easily modified to other needs: "While our implementation is focused on accessing solar physics data, our architecture and components can be reused easily in other domains with similar large data volume constraints and browsing requirements." [43]. But when code was studied, the data was organized based on characteristic suitable for solar physics data, and refactoring the application for ILSR's needs could require fair amount of work.

The second remaining category two's solution, IIPMooViewer, is a web-based open-source image streaming and zooming client developed by Ruven Pillay [44]. Unlike Helioviewer.org, it is general usage viewer and not designed to any specific purpose. Similarly to JHelioviewer, the IIPMooViewer is the client application for displaying the images and it is designed to work with IIPImage server, which is responsible for streaming the images, although it can be used with any server software supporting Zoomify, Deepzoom, Djatoka (Open URL) or IIIF protocols [44]. Similarly IIPImage server can be used with any other client software supporting IIP, IIIF, Zoomify or Deepzoom protocols [45].

Same way as Helioviewer.org, IIPMooViewer used with IIPImage server also meets the two first requirements for the ILSR whole slide imaging framework, as the viewer is purely web-based and server natively supports JPEG2000 images in addition to TIFF images. The software was quick to adapt as it was easily set up within few hours. The project is active and code base is well maintained, although the public bug tracking service is seldom used. It is unknown if the developer team has an alternative bug tracking system assigned for internal use. The server and viewer are both extensible through used streaming protocols. The viewer also has a synchronous view mode, where two different images could be zoomed and panned synchronously, which could be useful in future ILSR development. The only disadvantage of IIPMooViewer was that it does not support JPIP protocol and it is unknown how much effort its implementation would require from the developer team.

The last remaining category two's solution, OpenSeaDragon, is a web-based open-source viewer for zoomable images [46]. Unlike Helioviewer.org and IIPMooViewer/IIPImage, OpenSeaDragon is only a client application and needs to be connected to an external server. For the server connection, OpenSeaDragon supports IIIF, Deepzoom, Open Street Maps and Tiled Map Service protocols as well as custom tile sources. IIPImage would be a suitable server to be used with OpenSeaDragon for example as they share common protocol supports. The advantage of OpenSeaDragon is that it is an active project, code base is well maintained, online documentation is provided and issue tracker is used actively. Adaptation is quick as there is no need to set up any back-end for the client as it is implemented purely with JavaScript and can be ran on web browser as it is. Extensibility is supported via plugins. The disadvantage is that if OpenSeaDragon is used, the server needs to be implemented separately and adapting and maintaining two code bases usually means increased risks.

Third and fourth category's solutions were evaluated lastly. Third and fourth category contains five whole slide imaging frameworks, from which two of the first frameworks, Cytomine and Slide-Atlas, are open-source solutions allowing the developer team to modify them for ILSR's need and three remaining frameworks, JVS Web Viewer, HeteroGenius Medical Image Manager and Zoomify, are closed-source solutions in which modifications requires collaboration with the framework providers. All of the five solutions are ready-made frameworks for displaying whole slide images, containing set of annotation tools, and were all considered as feasible option for implementing the ILSR's whole slide imaging feature.

The first of the open-source frameworks, Cytomine, is a web-based research application for managing, displaying, annotating and analyzing digital slides [16]. Although applications initial focus is on biomedical research in cytology and histology whole slide images, Cytomine's developer team is constantly developing application further to create more generic purpose software which could be used in various different fields [47]. Cytomine is developed at the University of Liege's System and Modeling group. Cytomine application architecture consists of several isolated modules communicating with each other through API calls, including an image server and web-based user interface that is used for managing user access, projects and images, and for displaying and annotating whole slide images. Image server module has dependencies to IIPImage and OpenSlide solutions evaluated earlier in this chapter. As mentioned earlier, IIPImage supported IIP, IIIF, Zoomify and Deepzoom protocols, but not JPIP protocol, which is protocol supported by DICOM. However, Cytomine development team has a plan to extend the software to support "latest standard definitions in digital pathology" [16].

Cytomine seemed very potential solution for implementing ILSR's whole slide imaging feature. It met with the web-based requirement, and JPEG2000 images are supported with wide range of other image formats as well [48]. Viewer comes with extensive list of annotation tools including Magic Wand tool that can be used for automatically detecting edges of the object based on the color. Project is active, and bug tracking tools are used. Code base is isolated into modules and latest software technologies, frameworks and patterns are used, which makes modifying and maintaining the code easier and faster to adapt. Cytomine uses Docker virtualization platform for deploying the different modules of the application, which allows distributing the different modules running on different servers if needed, and scaling up by running multiple copies of the same modules is also supported by the system. Cytomine is licensed under Apache License version 2.0 which allows freely use, modify, distribute and sell the software as part of commercial software [49]. The framework can be extended via own algorithms and plug-ins and it comes with machine learning algorithms for detecting for example tissue substructures, cell types and landmarks, which might become useful in future ILSR development [47]. Only disadvantages comes with security requirement and adaptation. Although Cytomine has an inbuilt authentication and user access functions, it lacks support of https protocol, although including it might be trivial. Second concern is adaptation of the system. Although code base is well maintained, Cytomine is build

using various latest technologies in which development team does not have strong previous experience, including technologies such as Docker, Grails, Backbone.js, Openlayers and MongoDB to name few. Learning these new technologies will slow down the adaptation process, although the time might be saved later as the same technologies eases the application maintainability. Another thing slowing down the adaptation process is that Cytomine requires some modifications for seamless integration. Cytomine comes with its own user access control and user interface, and Cytomine needs to be modified so that those can be replaced with ILSR's counterparts.

The second of the open-source frameworks, Slide-Atlas by Kitware, is a web-based whole slide imaging platform. It supports whole slide displaying and annotating as well as management with per-user or per-group permissions [50]. It has dependency to OpenSlide library but it is not documented which image streaming protocols, if any, it uses. Slide-Atlas meets the web-based requirement and it is using OpenSlide, so it should support JPEG2000 images, although mentions of JPEG2000 could not be found from documentation. Slide-Atlas comes with simple set of annotation tools including text, circle and free form, which needs to be extended if the framework is chosen to implement ILSR's whole slide imaging feature. Slide-Atlas has dual comparison view similarly to IIPMooViewer, which would be useful in future ILSR development. The code base is well maintained and bug-tracker used, but adaptation and maintainability could cause a risk because lack of documentation of key areas, such as documentation about user software architecture.

The first of the closed-source solutions, JVS Web Viewer, is a web-based research whole slide framework that includes a viewer and set of annotation tools developed at the University of Tampere BioMediTech - Institute of Biosciences and Medical Technology [51]. In addition to JVS Web Viewer, the JVS microscope project also offers freely downloadable and usable DICOM linkable JPIP protocol compliant image server and viewer application for desktop usage. JVS Web Viewer is the viewer solution currently used for displaying whole slide images in ILSR. It is a web-based solution supporting JPEG2000 images. JVS Web Viewer also meets security standards as it uses https protocol and user authentication methods. Latest version of JVS Web Viewer also supports extensive set of annotation tools, although at the time of writing this thesis the annotation functionality was not integrated with the ILSR. Adaptation of JVS Web Viewer is easy as the application server is maintained by JVS Web Viewer's development team and main functionalities such as uploading, deleting and displaying a whole slide image are accessed through web-based API. JVS Web Viewer had functioned well as a whole slide viewer in ILSR while writing this thesis, but challenges related to closed-code solutions such as unknown maintainability, scalability and extensibility characteristics causes risks in ILSR's future development. As ILSR is research project constantly evolving, a rapid iteration loop is required when developing new features, which might become difficult to build when developing with external development team. Also visions and interests of future development plans of two development teams usually differ, which might cause problems in future ILSR development.

The second of the closed-source solutions, Medical Image Manager, or MIM, by HeteroGenius, is a web-based application for displaying, managing, and analyzing whole slide images [52]. MIM comes with server software, and a web-based user interface is used for managing, displaying, annotating and analyzing images. The platform is free to use and can be extended using with paid or free add-on modules. HeteroGenius also offers paid version of the platform called Medical Image Manager Pro, which comes with additional add-ons and support contract. MIM meets the web-based requirement and supports JPEG2000 file format. The system has a user access control but the connection is not using https protocol, and it is unknown how easy it would be to enable it. The viewer comes with a set of annotation tools, arrow, box, curve, ellipse and line, but lacks freehand tool. MIM has interesting extensions, which could be great assets in future ILSR development, such as 3D Pathology and Radiology add-ons. 3D Pathology add-on is used for stacking whole slide images for building a 3D volumetric model and Radiology Correlation add-on can be used for aligning whole slide images with radiology image volumetric data [53]. When the system was tested, there did not seem to be any tools for integrating MIM platform with external systems such as ILSR which causes possible risk.

The last of the closed-source solutions, Zoomify, is a commercial general purpose high-quality image viewer developed by Zoomify, Inc. [54]. Zoomify offers three different products to be purchased: Zoomify Express, Zoomify Pro and Zoomify Enterprise. The express version has only image viewer feature whereas the pro version gives more control over how the viewer works and rights to modify the viewer's source code [55]. The enterprise version offers extra features such as annotations, measurements, user access control and support for IIIF protocol [55]. Out of the three product versions of Zoomify, the enterprise version seems most suitable for implementing ILSR's whole slide imaging feature. Zoomify meets with the web-based requirement, but JPEG2000 is not supported natively. Zoomify announces on their website that they have implemented beta version of JPEG2000 support, but suspended the development before releasing it [56]. However, Zoomify does provide optional image conversion feature that supports JPEG2000 image formats [55]. The enterprise version comes with user authentication feature, but it is unknown if https is supported. Zoomify comes with simple set of annotation tools: freehand, rectangle and polygon tools are supported. Zoomify web-page states that viewer code could be modified once the pro or the enterprise version is purchased, but while writing this thesis neither of the versions were available for testing, so maintainability and adaptation requirements were not evaluated. Using Zoomify to implement ILSR's whole slide imaging feature would be a risk, as there are no documentation available of how to integrate Zoomify with external systems such as ILSR.

Third step – Final decision

The final step started with eight top candidates: IIPMooViewer, Helioviewer, OpenSeaDragon, Slide-Atlas, Zoomify, Cytomine, JVS Web Viewer and Medical Image Manager. The advantages and disadvantages of the remaining solutions were com-

pared against each other to form a final decision. The three remaining second category's solutions IIPMooViewer, Helioviewer and OpenSeaDragon were considered as a core elements for the new framework. Using core elements would be in align with the lean development process second principle, building quality in, as development team could ensure that the ILSR requirements would be fulfilled when implementing the new framework. However, as suitable frameworks meeting most of the requirements were available as well, it was decided that third (implementation using framework) or fourth (Integration with off-the-shelf framework) implementation approach were more suitable for ILSR whole slide imaging feature development, and hence the whole second implementation approach (implementation using core solution) and its remaining candidates were discarded. This decision narrowed down the number of selections from 8 to 5.

Slide-Atlas was discarded next from ILSR's whole slide imaging plans. It seemed too much of work for the developer team to adapt and maintain a project that does not meet maintainability and adaptation requirements by lacking documentation. Without proper documentation, it would be difficult to build overall vision of the used architecture and solutions and estimate the changes required. Also extending Slide-Atlases annotation tools would require extra work. Zoomify was also discarded as there was not enough information to evaluate it thoroughly, and it was difficult to tell how easy it would be to add a support to JPEG2000 images.

Finally the Cytomine, JVS Web Viewer and Medical Image Manager were the three most promising framework alternatives for the ILSR project. All of the frameworks were web-based and supported JPEG2000 images. All of the frameworks were supporting user authentication but JVS Web Viewer was the only one supporting usage of https protocol. All frameworks supported annotation tools, although Cytomine had most and Medical Image Manager the least extensive selection. Cytomine was the only framework that was open-source, allowing full control of modification to the development team. JVS Web Viewer had the best performance, according to quick test between the viewers, although frameworks could not be tested using similar hardware, as JVS Web Viewer was hosted by the external provider. Cytomine's license allows usage of the software as part of commercial application, whereas although JVS Web Viewer and Medical Image Manager are free to use, it is unknown if they can be used as part of commercial application. JVS Web Viewer is easier to adapt from all the frameworks, as its viewer part was already integrated with the ILSR. Cytomine needs some modification before its usage can be started. It is unknown how much effort is needed to integrate Medical Image Manager with the ILSR. Medical Image Manager has best extensibility for ILSR's needs, as it comes with add-ons for 3D modeling and radiology and adding custom add-ons is supported. JVS Web Viewers advantage is that it is DICOM ready and Cytomine comes with promising machine learning algorithm analysis tools.

All of the three frameworks were capable of implementing the ILSR whole slide imaging feature. Some framework met some of the requirements better than others, and other frameworks performed better in other areas, but it seemed plausible that any of the frameworks could be modified, within developer team or in collaboration with the

framework provider to meet all the requirements. In the end, Cytomine was picked as the framework that was going to be used for implementing the new whole slide imaging feature. The third implementation approach would give the developer team full control of how the framework, and its features would be modified and maintained in the future. When collaborating with third party providers, the risk of conflict of interest always needs to be taken into consideration. Furthermore, keeping the development iteration loop short between the two development teams can be challenging. Although maintaining the development of the framework would burden the small development team more than leaving the development responsibility to third party frameworks developer team, the work was considered to be within acceptable limits.

6. RESULTS

While writing this thesis, the integration process of Cytomine with the ILSR was in late development phase. As a proof of concept, an initial integration of ILSR and Cytomine was implemented and users were able to upload, delete and display whole slide images using ILSR front-end and secure https connection. Although the new feature was not yet deployed to production, everything points to the direction that Cytomine can and will be used for implementing the new whole slide imaging feature in ILSR, thus replacing previously used JVS Web Viewer.

Integration between the two systems required various development steps, including software modification and server maintenance related tasks. The main tasks were setting up Cytomine server, setting up Cytomine development environment, modifying ILSR for the integration, enabling https connection for Cytomine, migrating Cytomine for using ILSR's image storage and customizing Cytomine's user interface. The different tasks are described with found issues and challenges and how they were solved. After that the overall integration is evaluated based on the whole slide framework requirements presented in Table 5.1 and support of the future development of ILSR is discussed. Finally, the used evaluation method and its success in this project is evaluated.

6.1 Issues and Challenges in Integration

Cytomine server was set up first for evaluation and later for actual integration with the ILSR system. Cytomine is a complex system and has dependencies to wide set of different external software libraries and components. To ease up the deployment, Cytomine uses Docker virtualization platform. Docker enables running isolated virtual containers on a host system, which are like virtual machines, only lighter as they share the host operating systems kernel [57]. Every Cytomine core element such as core server, database servers and image management system servers are modules, and ran on different isolated Docker containers, making the deployment easier as there is no need of worrying conflicts between different module's dependencies. Only challenges occurred with learning how Docker system operates, as developer team did not have previous experience on it, setting the firewall rules to allow communication between different containers and one bug that was detected in Cytomine bootstrap script. The bug was reported to Cytomine developer team through bug tracker. The solution to the problem was given within same day and the bug was fixed in Cytomine repository at the next day from reporting the bug.

The next thing was to set up the Cytomine development environment in order to modify the code server module, which is responsible for providing the API and the web front-end for managing, displaying and annotating the images. Using Docker isolation helps with setting up the development environment as well. Other core modules are run exactly the same way as when Cytomine is ran on production mode, the only difference is that the core server module is run in development mode, enabling seeing debug messages and effects of code changes instantly. The only challenges were that development documentation for setting up the development environment was not clear on all the details. Once development environment was successfully set up, suggested fixes were sent to the Cytomine developer team, but the instructions were not fixed into Cytomine documentation while writing this thesis, a month after posting the suggested fixes.

Next task was to enable https connection between ILSR and Cytomine. By default Cytomine is running on http and there were no instructions of how to modify, or which settings to change in order for Cytomine to be ran using https connection. However, Cytomine uses nginx and Apache Tomcat, which are well known and used web server technologies, and their documentation provides instructions how to enable https protocol. Biggest challenges included analyzing the Cytomine system to identify the elements needing modification, learning how to configure the server software and to provide certificate files in format that was accepted by the server software.

The next challenge was migrating Cytomine for using ILSR's image storage. Both Cytomine and ILSR have their own ways of storing and organizing their image data. There are several reasons why the two systems needs to be modified to use one common image storage, such as making backups, tracing and issues with disk space. ILSR centralizes all the data to Microsoft SQL database server, which has a backup plan, and in order to keep data backups consistent, the whole slide image data is required to be stored there with the rest of the ILSR data. One solution would be store the image files into the ILSR database, and a second copy of images into Cytomine's image storage and keep the data synchronized. This, however, causes a risk that if data synchronization fails at any reasons, the data between the two systems becomes inconsistent. In the worst case scenarios this could mean that wrong whole slide images could be displayed for the user or annotations and metadata would be linked to wrong images. Another issue with storing duplicates is caused by the disk space usage. Whole slide images can get as big as one gigabyte, and storage space could become an issue as ILSR system will store thousands of whole slide images in the future.

Another solution would be migrating Cytomine for using ILSR's database as a storage. However, modifying Cytomine to use database as image storage instead of the filesystem can take fair amount of effort. After different opinions were considered, a MS SQL's filetable functionality was decided to be tested. Filetable is introduced in MS SQL 2012 and designed for storing big blob objects, such as images. Filetable stores data onto a file system instead of regular database table, but allows access to them through database transaction as well as using normal file system operations [58]. Filetables gives the benefit of database engine to manage all the data, including taking back-

ups, but still giving the applications ability to use files through the filesystem. While writing this thesis, an initial implementation of Cytomine using database's filetable via network share was implemented successfully on development environment, but not yet in production environment. Using filetables seems to be a promising technology for unifying the image data storages, although performance issues caused by accessing the whole slide images through a network connection still needs to be tested.

The last challenge of customizing Cytomine's user interface was still ongoing process while writing this thesis. Cytomine comes with its own user interface, which allows users to do tasks such as managing Cytomine projects and uploading, annotating and analyzing whole slides. In first stage of integrating ILSR with the Cytomine, only the image viewing functionalities are required, followed by other stages that gradually starts enabling other features, such as annotations. Cytomine user interface needs to be modified so that all unnecessary toolbars can be hidden from the user, leaving only the viewer element visible. Cytomine comes with documentation how to partially achieve this using user roles, although some code modifications are needed, as even the minimized role-based view in Cytomine still leaves some of the toolbars visible. However, modifying Cytomine seems doable with limited resources, the biggest challenge being learning the used software frameworks such as Backbone.js and AngularJS.

6.2 Suitability of the Framework

The initial version of the whole slide imaging feature implemented using Cytomine met with all the requirements presented in Table 5.1. The feature is purely web-based and operates with the latest versions of modern browsers, Google Chrome, Mozilla Firefox and Microsoft Internet Explorer, without requiring installation of external browser plugins or changing any browser settings. The system supports the same JPEG2000 whole slide images that were used before in ILSR, and Cytomine supports wide set of other image formats as well. The connection is completely secured using https protocol after modification done by the developer team. The feature provides easy to use image displaying, with support for touch-screen based systems such as mobile phones and tables. The annotation tool set provides multiple tools that allows users for annotating areas of interest accurately and easily. The code base is structured so that it is easily maintainable and extensively documented, although the solution is using many different software solutions, which increases the learning curve for the new developers.

The system supports scalability, as the image processing can be distributed between different image servers. The viewer performed slightly slower than previous viewer, the JVS Web Viewer, in initial tests using different server support, but still within acceptable limits. A test was ran where image server's computational resources were increased from single CPU core to eight cores and memory was increased from four gigabytes to eight gigabytes, but it did not seem to increase the performance significantly. The optimization possibilities of Cytomine needs to be studied in the future. Cytomine is licensed using Apache Licence version 2.0 which allows freely use, modify, distribute

and sell the software as part of commercial software, which is suitable for ILSR's needs. The Cytomine framework was fairly easy to adapt, although familiarizing with technologies such as Docker, used web server software and Linux operating system caused a bit of learning curve and the user interface modification was still ongoing process while writing this thesis. Cytomine supports image data analysis tools that could be used in ILSR's development in the future and allows creating own custom analysis applications.

Cytomine is a suitable solution for implementing ILSR's whole slide imaging feature, but the framework is also capable of enabling ILSR's future development plans mentioned in Section 3.1, at least with some modifications. The first of the ILSR's development plans is using annotations stored in ILSR in order to record regions of interest in molecular analysis of tissues and for providing automated access to other metadata about the whole slide image. Cytomine comes with extensive set of annotation tools and it comes with machine learning algorithms that could be used for automating aspects of image annotation. The second plan includes using annotations as guide for cutting sub-sections from the section using LCM, in order to later extract and later study relevant biomolecules such as DNA or RNA from the cut sub-section. Cytomine supports this process as annotated whole slide images can be used as guide for the person responsible of operating the LCM device or cutting with scalpel. The automated process where Cytomine would drive the LCM device needs to be studied further.

The new framework supports maintaining tissue microarray data as it is and Cytomine's Magic Wand annotation tool could possibly be used for automatic detection of microarray spots. Magic Wand tool could possibly be also used for detecting the cut sections edges when generating 3D wireframe models of the prostate and tumour in the future. The registration of whole slide image data with other imaging modality data needs to be studied further and possibly requires implementation of JPIP protocol and DICOM support. Fortunately, Cytomine is built in a modular way and adding support for new streaming protocol would not require re-factoring the whole application, instead adding and configuring a new image server component would be sufficient.

6.3 Suitability of the Evaluation Method

The new evaluation method was found useful during the evaluation process, as it helped charting the field of whole slide imaging solutions and finding the right implementation approach to develop the new feature. The developer team felt that enough different solutions were included into the evaluation process and out of all the solutions, the most sustainable solution was found within acceptable amount of time. However, there is always room for improvement. The biggest issues with the evaluation method that could be improved in the future, were:

- time wasted in evaluating solutions too thoroughly in early phase,
- failure to utilize social networking better,
- underestimating the adaption workload,
- failure to discard implementation approaches sooner.

The first issue, **time wasted in evaluating solutions too thoroughly in early phase**, occurred when evaluating second category's solutions, especially Helioviewer and IIPMooViewer solutions. Both of the solutions were ranked as possible best candidates in early phase, before even evaluating all the third and fourth categories solutions. This was partly caused by the fact, that the identify step was not yet completed, when the evaluation of these solution was already started. At least two improvements could be done to prevent this happening in the future: enough time should be spent on identify phase, so that good coverage of different solutions is acquired and the solution and/or implementation approach should be discarded in earlier phase if they do not seem suitable for solving the problem. Although identify step should be done first, in practice it is ongoing process until the best candidate is picked and integration process has been started. It is a challenge to know when enough solutions has been acquired, especially when trying to find solutions to very specific business problems. One solution to find the best coverage seemed to be using many different search methods. For example using Google searches covered only part of the solutions. For example, when using search term "jpeg2000 server" on Google search, Cytomine's web site was not included in the top 100 page hits, and when terms "web-based whole slide imaging application" was used, Cytomine was 91st search result. For comparison, it is shown that "91% of searchers do not past page 1 or search results, and over 50% do not go past the first 3 results on page 1" [59]. When finding whole slide imaging frameworks, searching published papers using search engines like Google Scholar and NCBI search was found very useful.

Another improvement for avoiding wasting time to evaluating solutions too thoroughly in the early phase could be using scoring system introduced in SEI's evaluation method. The criteria for initial evaluation rounds could be determined and high enough score could be required in order for the solution to pass to next evaluation rounds. This would help evaluators to stay objective about the solutions and force finding other solutions in case of lack of suitable candidates.

The second issue, **failure to utilize social networking better**, could have been prevented by contacting the developer teams more actively, by contacting open-source user communities, and posting direct questions to forums. Often people are happy to share their knowledge can be found and by contacting other people gives fresh aspects for solving the business problem and people might provide additional information and insights that are not available in documentations. Also after the solutions is picked, the future might require developer team collaborating with the solution's developers, and early enquiries might act as the first link building a communication channel between the two teams.

The third issue, **underestimating the adaption workload**, occurred while integrating Cytomine with ILSR. Cytomine uses many different software frameworks and setting up and modifying it requires learning of various new skills. This is not blocking the developer team from integrating the Cytomine with ILSR, as the process is almost done, but the workload could been estimated better in the evaluation phase to avoid surprises.

The final issue, **failure to discard implementation approaches sooner**, could have been avoided by discarding first and possibly fourth implementation approaches in sooner stages. It was highly unlikely that the right implementation approach for the ILSR development team would been implementing the whole slide imaging feature from the scratch, due the massive workload. Also the purpose of evaluating different solutions was driven by need to develop whole slide imaging feature that would allow short development cycles and control over the feature. This is usually not the case with commercial software. However, evaluating the first categories solutions ended up being valuable, as solutions from other categories were using these first categories solutions, and this gave the developer team more tools to evaluate the performance of the solutions. Also this revealed a new aspect when evaluating ready-made customizable frameworks: it is beneficial not only to evaluate the top candidates thoroughly, but also to evaluate the underlying software components to avoid hidden surprises. Keeping the fourth categories solutions as part of the evaluation process was also not completely waste of time, as they acted as benchmarks for third categories solutions and gave insight of how others have solved their whole slide imaging business problems.

7. CONCLUSIONS

The goal of this study was to find means for developing major new software capabilities using available limited resources and at the same time not giving up of any of the existing or envisioned new software requirements. Small developer teams easily burden themselves with starting big projects when the planning phase is not done carefully or teams lack ability to leverage available technologies. Decision for implementing software components from the scratch needs to be justified as there are so many different software solutions available nowadays, and rebuilding something that is already available to be utilized can be seen as waste of resources. On the other hand, adopting a ready-made solution can also become a limitation for the future development plans or maintaining the project can become a burden, if solution's evaluation process is not performed carefully.

The new software evaluation method introduced in this thesis provided a solution to evaluate different software solutions and at the same time evaluate the right implementation approach for the developer team's needs. The three main advantages of the new evaluation method are:

1. The implementation approach is included in the evaluation,
2. The method includes open-source and closed-source solutions,
3. The method gives a wide perspective for solving the business problem,

The first advantage, **the implementation approach is included in the evaluation**, takes into account how much work is really needed when the new software is developed, an important concept that needs to be considered by the small development team. As available solutions are categorized by the amount of work they require for implementing the solution, a whole category's solutions can be quickly discarded if the implementation approach gets discarded.

The second advantage, **the method includes open-source and closed-source solutions**, means that the developer team does not need to be limited by the solution provider's business model and encourages keeping options open until the final decision is done.

The third advantage, **the method gives a wide perspective for solving the business problem**, is possible as the method is used for evaluating solutions of all sizes related to solving the business problems. The development team gets a good understanding of the different frameworks that are usually used for solving similar problems, and also understanding of what happens behind the curtains, as low-level libraries and core elements gets also involved into the evaluation method.

Although the evaluation method performed well when tested in developing a new whole slide imaging feature for ILSR's needs, the process can be improved. The results of the process were analysed and five concrete improvement suggestions were gathered:

1. make fast rejection,
2. spend time to identify the possible candidates,
3. contact people,
4. spend time evaluating and testing the top candidates,
5. evaluate risks.

The first improvement, **make fast rejection**, means that the solution should be discarded once enough information is gathered to make decision. Usually there are plenty of solutions available, and evaluating all of them thoroughly is wasteful. If solution fails at most critical requirements, meeting less important requirements usually will not improve the suitability so much that the solution would be useful. One possible way to add fast rejection to the evaluation method would be including SEI evaluation method's scoring system and identifying the most important requirements in the requirements gathering step.

The second improvement, **spend time to identify the possible candidates**, is essential. There are no organizations that wants to waste time first evaluating the second best solution thoroughly and then spend even more time integrating it with their own system. Enough time should be spend and several different channels should be used for finding enough software candidates in the first step of evaluation process. Although ideally all the candidates are identified in early phases of evaluation, in practice candidates can be found in any phase during the evaluation process.

The third improvement, **contact people**, is related to identify step, but also to evaluation step. Groups such a solutions' developer teams, software communities, support forums, users and colleagues can all provide insight, aspects that were not considered and information that was not documented if they are contacted. The developer team just needs to reach out and ask to gain valuable new information.

The fourth improvement, **spend time evaluating and testing the top candidates**, means installing the top candidate solutions, testing them thoroughly by the developer team and end-users as well, if possible, studying code base, using load tests and considering suitability against all the requirements. Not all defects and limitations can be spotted in the testing phase, but developer team should do their best.

The fifth improvement, **evaluate risks**, was possibly the biggest lack of the introduced evaluation method. Non-functional risk analysis should be implemented as part of the method to answer questions similar than used in SEI's method: "Is the (software solutions) company well established?", "What is the longevity of the company?", "Is support offered?", "Is the vendor financially stable?", "How mature is the technology used?", and in case of closed-source solutions, "Is the vendor flexible to make changes to the software?". These questions are important when a solution is evaluated, regardless if is it open-source or closed-source. No-one knows the solution as well as its development team, and it is very beneficial that the developer team continues putting ef-

fort on developing the original solution, even if the organization has adapted and modified it to its own needs.

Even though the evaluation method has room for improvement, it is still considered beneficial to be used for evaluating software solutions by development teams, regardless of their size, in its current form. The evaluation method does not take into account team size as such, but usually small development teams can make more rapid decisions, as there are less communication overhead, and the evaluation method is based on doing fast rejections for solutions that are not considered useful. Solving the communication overhead in context of evaluation methods is interesting subject, but not in scope of this thesis.

The evaluation method is not specific for life science context and can be used for any type of software projects, when adjusted in the identify phase to use search tools most usable for the environment of the business problem. For example search engines designed for finding scientific publications such as Google Scholar and NCBI search were used when the evaluation method was used for solving the whole slide imaging problem, but other projects might benefit for using some other tools.

This study was considered useful for needs of ILSR development. When the project started, the world of whole slide imaging solutions were uncharted. The evaluation method revealed the solutions one by one, helped ranking them and estimating the amount of work required, and finally lead to selection of the solution that was considered the best sustainable solution for ILSR's needs. With lessons learned from this study, the evaluation method gets evolved into even more efficient method for evaluating available software and helping utilizing the limited resources of a small development team.

REFERENCES

[1] Medical Imaging Modalities. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.medicalimaging.org/about-mita/medical-imaging-primer/

[2] James, A. P & Dasarathy, B. V. 2014. Medical Image Fusion: A Survey of the state of the art. Information Fusion 19, September, pp. 4-19.

[3] DNA damage seen in patients undergoing CT scanning. [WWW]. [Accessed on 22.4.2016]. Available at: http://med.stanford.edu/news/all-news/2015/07/dna-damage-seen-in-patients-undergoing-ct-scanning-study-finds.html

[4] Rahmim, A. & Zaidi, H. 2008. PET versus SPECT: strengths, limitations and challenges. Nuclear Medicine Communications 29, pp. 193-207.

[5] Mease, R. C., Foss, C. A. & Pomper, M. G. 2013. PET imaging in prostate cancer: focus on prostate-specific membrane antigen. Current Topics in Mediclinical Chemistry 13, 8, pp. 951-962.

[6] Single Photon Emission Computed Tomography (SPECT). [WWW]. [Accessed on 16.4.2016]. Available at: http://www.heart.org/HEARTORG/Conditions/HeartAttack/SymptomsDiagnosisofHeartAttack/Single-Photon-Emission-Computed-Tomography-SPECT_UCM_446358_Article.jsp#.VxJuQPl974d

[7] SPECT vs. PET, Which is Best? [WWW]. [Accessed on 16.4.2016]. Available at: http://www.dicardiology.com/article/spect-vs-pet-which-best

[8] Farahani, N., Parwani, A. V. & Pantanowitz, L. 2015. Whole slide imaging in pathology: advantages, limitations, and emerging perspectives. Pathology and Laboratory Medicine International 7, pp. 23-33.

[9] Histotechniques. [WWW]. [Accessed on 16.4.2016]. Available at: http://library.med.utah.edu/WebPath/HISTHTML/HISTOTCH/HISTOTCH.html

[10] Bloom, K. J. IHC Staining Methods, Fifth Edition. California 2009, Dako North America. 172 p.

[11] Archival and Retrieval in Digital Pathology Systems. Madison WI 2011, Digital Pathology Association. 9 p.

[12] It all starts with a quality scan… [WWW]. [Accessed on 16.4.2016]. Available at: http://flagshipbio.com/news/it-all-starts-with-a-quality-scan/

[13] Campbell, W., Foster, K. & Hinrichs, S. 2013. Application of whole slide image markup and annotation for pathologist knowledge capture. Journal of Pathology Informatics 4.

[14] Curran, S., McKay, J. A., McLeod, H. L. & Murray, G. I. 2000. Laser capture microscopy. Molecular Pathology 53, 2, pp. 64-68.

[15] Hamilton, P. W., Wang, Y., Boyd, C., James, J. A., Loughrey, M. B., Hougton, J. P., Doyle, D. P., Kelly, P., Maxwell, P., McCleary, D., Diamond, J., McArt, D. G., Trunstall, J., Bankhead, P. and Salto-Tellez, M. 2015. Automated tumor

analysis for molecular profiling in lung cancer. Oncotarget 6, 29, pp. 27938-27952.

[16] Marée, R., Stévens, B., Rollus, L., Rocks, N., Lopez, X. M., Salmon, I., Cataldo, D. & Wehenkel, L. 2013. A rich internet application for remote visualization and collaborative annotation of digital slides in histology and cytology. Diagnostic Pathology 8, pp. 1-4.

[17] Bidgood, W. D. Jr., Horii, S. C., Prior F. W. & Van Syckle, D. E. 1997. Understanding and using DICOM, the data interchange standard for biomedical imaging. Journal of the American Medical Informatics Association 4, 3, pp. 199-212.

[18] ILSR Introduction, History, Standards, and Current Status. [WWW]. [Accessed on 16.4.2016]. Available at: http://ilsr-fogbugz.uta.fi/default.asp?W41

[19] Jawhar, M. T. 2009. Tissue Microarray: A rapidly evolving diagnostic and research tool. Annals of Saudi Medicine, 29, 2, pp. 123-127.

[20] Liu, W., Laitinen, S., Khan, S., Vihinen, M., Kowalski, J., Yu, G., Chen, L., Ewing, C. M., Eisenberger, M. A., Carducci, M. A., Nelson, W. G., Yegnasubramanian, S., Luo, J., Wang, Y., Xu, J., Isaacs, W. B., Visakorpi, T. & Bova, G. S. 2009. Copy number analysis indicates monoclonal origin of lethal metastatic prostate cancer. Nature Medicine, 15, 5, pp. 559-65.

[21] Li, Y., Alsagabi, M., Fan, D., Bova, G. S., Tewfik, A. H. & Dehm, S. M. 2011. Intragenic rearrangement and altered RNA splicing of the androgen receptor in a cell-based model of prostate cancer progression. Cancer Research 71, 6, pp. 2108-2117.

[22] Heintzelman, N. H., Taylor, R. J., Simonsen, L., Lustig, R., Anderko, D., Haythornthwaite, J. A., Childs, L. C. & Bova, G. S. 2013. Longitudinal analysis of pain in patients with metastatic prostate cancer using natural language processing of medical record text. Journal of the American Medical Informatics Association 20, 5, pp. 898-905.

[23] Nickerson, M. L., Im, K. M., Misner, K. J., Tan, W., Lou, H., Gold, B., Wells, D. W., Bravo, H. C., Fredrikson, K. M., Harkins, T. T., Milos, P., Zbar, B., Linehan, W. M., Yeager, M., Andresson, T., Dean, M. & Bova, G. S. 2013. Somatic alterations contributing to metastasis of a castration-resistant prostate cancer. Human Mutation 34, 9, pp. 1231-1241.

[24] Applying Lean to Software Development, an Excerpt from The Art of Software Development. [WWW]. [Accessed on 16.4.2016]. Available at: http://fyi.oreilly.com/2009/01/chapter-2-applying-lean-to.html

[25] Defect Prevention: Reducing Costs and Enhancing Quality. [WWW]. [Accessed on 16.4.2016]. Available at: https://www.isixsigma.com/industries/software-it/defect-prevention-reducing-costs-and-enhancing-quality/

[26] Are 64% of Features Really Rarely or Never Used? [WWW]. [Accessed on 16.4.2016]. Available at: https://www.mountaingoatsoftware.com/blog/are-64-of-features-really-rarely-or-never-used

[27] How to Evaluate Open Source Software / Free Software (OSS/FS) Programs. [WWW]. [Accessed on 24.4.2016]. Available at: http://www.dwheeler.com/oss_fs_eval.html

[28] Bandor, M. S. Quantitative Methods for Software Selection and Evaluation. Pittsburgh, Pennsylvania 2006, Carnegie Mellon University. Technical Note CMU/SEI-2006-TN-026. 12 p.

[29] Top Tips For Selecting Open Source Software. [WWW]. [Accessed on 24.4.2016]. Available at: http://oss-watch.ac.uk/resources/tips

[30] Young, R. R. 2002. Recommended Requirements Gathering Practices. Cross-Talk, Apr 1, pp. 9-12.

[31] DICOM Supplement 106: JPEG 2000 Interactive Protocol. [WWW]. [Accessed on 16.4.2016]. Available at: http://dicom.nema.org/dicom/Conf-2005/Day-2_Selected_Papers/B302_Weisfeiler_Supplement%20106%20--%20JPIP.pdf

[32] The JasPer Project Home Page. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.ece.uvic.ca/~frodo/jasper/

[33] Kakadu Software. [WWW]. [Accessed on 16.4.2016]. Available at: http://kakadusoftware.com/

[34] OpenJPEG. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.openjpeg.org/

[35] OpenSlide. [WWW]. [Accessed on 16.4.2016]. Available at: http://openslide.org/

[36] Palmer, W., May, P. & Cliff, P. 2013. An Analysis of Contemporary JPEG2000 Codecs for Image Format Migration. Proceedings of the 10th International Conference on Preservation of Digital Objects. Available at: http://scape-project.eu/wp-content/uploads/2013/11/iPres2013_Palmer_JPEG2000Codecs.pdf

[37] OpenJPIP v2.1.0 Documentation. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.openjpeg.org/doxygen/openjpippage.html

[38] CADI software. [WWW]. [Accessed on 16.4.2016]. Available at: http://gici.uab.es/CADI/

[39] 2KAN. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.2kan.org/

[40] GSoC: JPEG2000 JPIP Server and Viewer Applet. [WWW]. [Accessed on 16.4.2016]. Available at: http://dltj.org/article/gsoc-jpip/

[41] webjpip.js. [WWW]. [Accessed on 16.4.2016]. Available at: https://github.com/MaMazav/webjpip.js/tree/master

[42] The Helioviewer Project. [WWW]. [Accessed on 16.4.2016]. Available at: http://wiki.helioviewer.org/wiki/Main_Page

[43] Mueller, D., Dimitoglou, G., Caplins, B., Ortiz, J. P. G., Wamsler, B., Hughitt, K., Alexanderian, A., Ireland, J., Amadigwe, D., & Fleck, B. 2009. JHelioviewer - Visualizing large sets of solar images using JPEG 2000. ArXiv e-prints. 19 p.

[44] IIPMooViewer. [WWW]. [Accessed on 16.4.2016]. Available at: http://iipimage.sourceforge.net/documentation/iipmooviewer/

[45] Internet Imaging Protocol. [WWW]. [Accessed on 16.4.2016]. Available at: http://iipimage.sourceforge.net/documentation/protocol/

[46] OpenSeaDragon. [WWW]. [Accessed on 16.4.2016]. Available at: http://openseadragon.github.io/

[47] Cytomine. [WWW]. [Accessed on 16.4.2016]. Available at: http://cytomine.be/

[48] Cytomine Development Documentation – Part 8: Image Server. [WWW]. [Accessed on 16.4.2016]. Available at: http://doc.cytomine.be/display/DEVDOC/Part+8%3A+Image+server

[49] Top 10 Apache License Questions Answered. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.whitesourcesoftware.com/whitesource-blog/top-10-apache-license-questions-answered/

[50] Slide-Atlas. [WWW]. [Accessed on 16.4.2016]. Available at: http://slideatlas.kitware.com/

[51] JPEG2000 Virtual Slide microscope. [WWW]. [Accessed on 16.4.2016]. Available at: http://jvsmicroscope.uta.fi/?q=about

[52] HeteroGenius Medical Image Manager. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.medicalimagemanager.com/static/MIM/index.html

[53] HeteroGenius MIM Add-ons and Features. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.medicalimagemanager.com/static/MIM/features/index.html

[54] Zoomify. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.zoomify.com/index.html

[55] Zoomify – Compare features. [WWW]. [Accessed on 16.4.2016]. Available at: http://www.zoomify.com/compare.htm

[56] Zoomify – Frequently asked questions: What support does Zoomify offer for JPEG2000? [WWW]. [Accessed on 16.4.2016]. Available at: http://www.zoomify.com/support.htm#a20081216_2154

[57] What is Docker? [WWW]. [Accessed on 16.4.2016]. Available at: https://www.docker.com/what-docker

[58] FileTables. [WWW]. [Accessed on 16.4.2016]. Available at: https://msdn.microsoft.com/en-us/library/ff929144.aspx

[59] How many Google searchers go to page two of their search results? [WWW]. [Accessed on 24.4.2016]. Available at: https://www.quora.com/How-many-Google-searchers-go-to-page-two-of-their-search-results