INFORMATIKA
UPN "VETERAN" JAWA TIMUR

Software Design Document

Radical Rakhman Wahid

30 April 2019
24 Sya'ban 1440

Yang ada pada presentasi ini :

1 Prinsip *Software Design Document* yang Baik

2 *Overview Template Software Design Document*

Prinsip *Software Design Document* yang Baik

Berikut adalah lima prinsip dasar dari *software design document* yang baik :

1. Tahu tujuan awal dokumen dibuat
2. Gambarkan aliran data pada perangkat lunak
3. Sediakan banyak contoh kode program
4. Dokumentasikan proyek pada berbagai tingkatan
5. Catat hal-hal penting ketika mengembangkan *software*

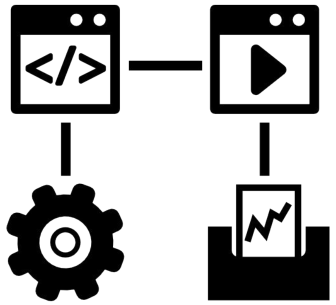
Slides selanjutnya akan menjelaskan lebih lanjut mengenai kelima prinsip dari *software design document* yang baik.

1. Tahu tujuan awal dokumen dibuat

Apa pun dokumentasinya, mengetahui siapa yang akan membaca dokumen dan mengapa mereka harus membacanya adalah suatu hal yang penting. Sebagai seorang pengembang kita harus tahu bahwa tidak semua pengguna merupakan orang yang memang berpengalaman dalam bidang teknologi informasi. Oleh karena itu pilihan bahasa dan kata-kata harus disesuaikan dengan si pembaca dan tujuan awal mengapa dokumen desain perangkat lunak ini ada.

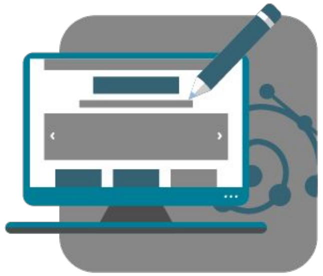
2. Gambarkan aliran data pada perangkat lunak

Ini akan memudahkan pembaca memahami alur kerja sebuah perangkat lunak. Selain itu juga gambaran aliran data bisa mencegah sesuatu yang tidak diharapkan kemunculannya seperti *bottleneck*.



3. Sediakan banyak contoh kode program

Jangan hanya memberikan satu contoh kode program secara penuh pada dokumentasi perangkat lunak. Pada dokumentasi yang baik hendaknya kode program dibagi-bagi sesuai dengan fungsinya masing-masing tidak ditampilkan utuh secara keseluruhan. Ini akan membuat pembaca bingung.



4. Dokumentasikan proyek pada berbagai tingkatan

Mengelola dokumentasi seperti ini memberi pembaca cara cepat untuk menemukan apapun yang mereka cari karena pada tingkatan tertentu ada dokumentasinya sendiri-sendiri.

5. Catat hal-hal penting ketika mengembangkan *software*

Ini akan mempersingkat waktu yang diperlukan untuk membuat dokumentasi. Tidak perlu waktu lama untuk menulis mengenai perangkat lunak yang dikembangkan karena hal-hal penting sudah tercatat dengan baik. Mencatat hal-hal penting juga dapat membantu dalam mengidentifikasi kekeliruan logika atau mengurangi kerumitan sebuah desain perangkat lunak.

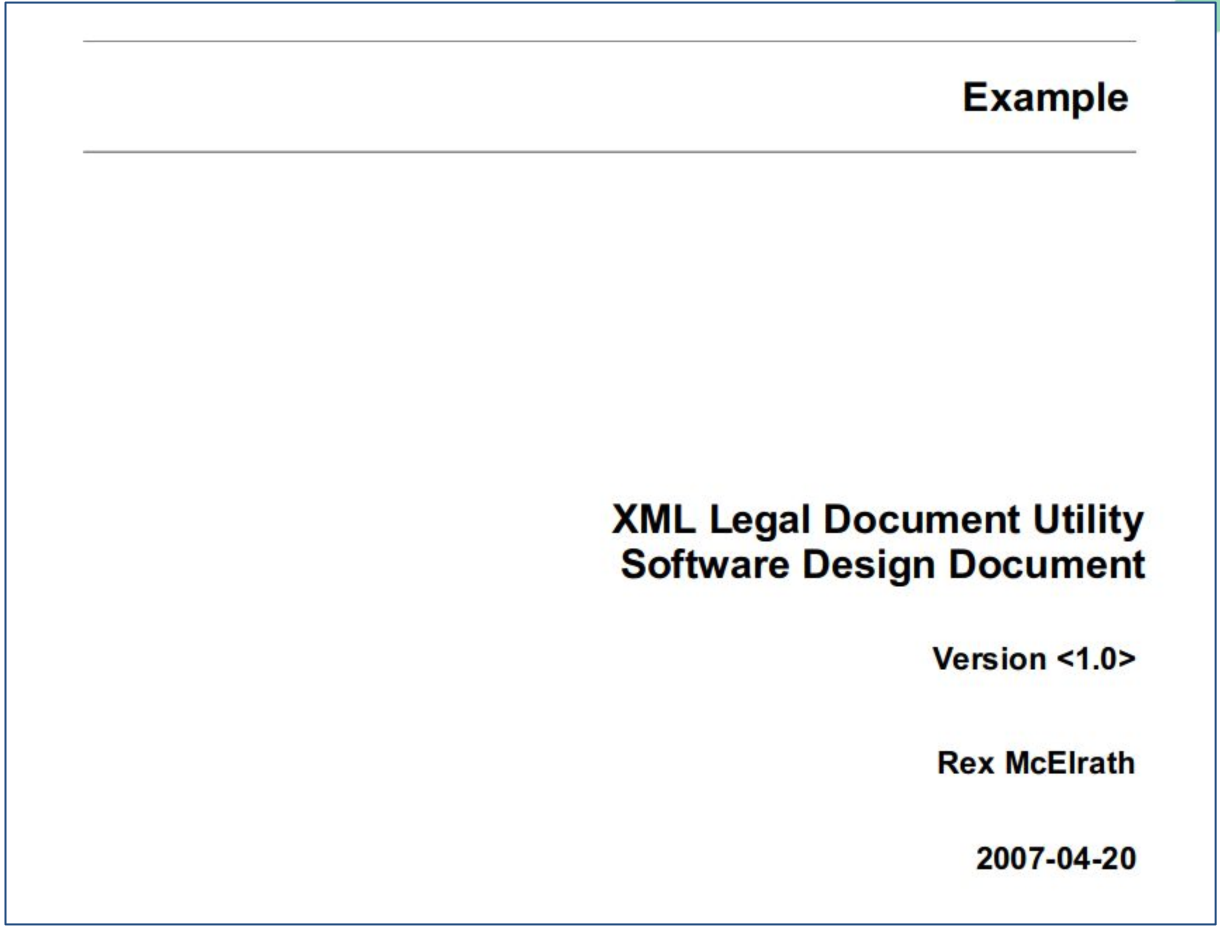
Overview Template Software Design Document

Menurut *The University of Western Australia* templat *software design document* memiliki setidaknya ada empat belas bagian, yakni :

1. Sampul depan
2. Riwayat revisi
3. Daftar isi
4. Pengantar
5. Glosarium
6. *Use cases*
7. Ikhtisar desain
8. *System object model*
9. Deskripsi objek
10. Kolaborasi objek
11. Desain data
12. Model dinamis
13. Persyaratan non fungsional
14. Dokumentasi tambahan

Bagian empat sampai empat belas merupakan isi dari daftar isi, berikut adalah penjabarannya :

Sampul Depan Dokumen



Example

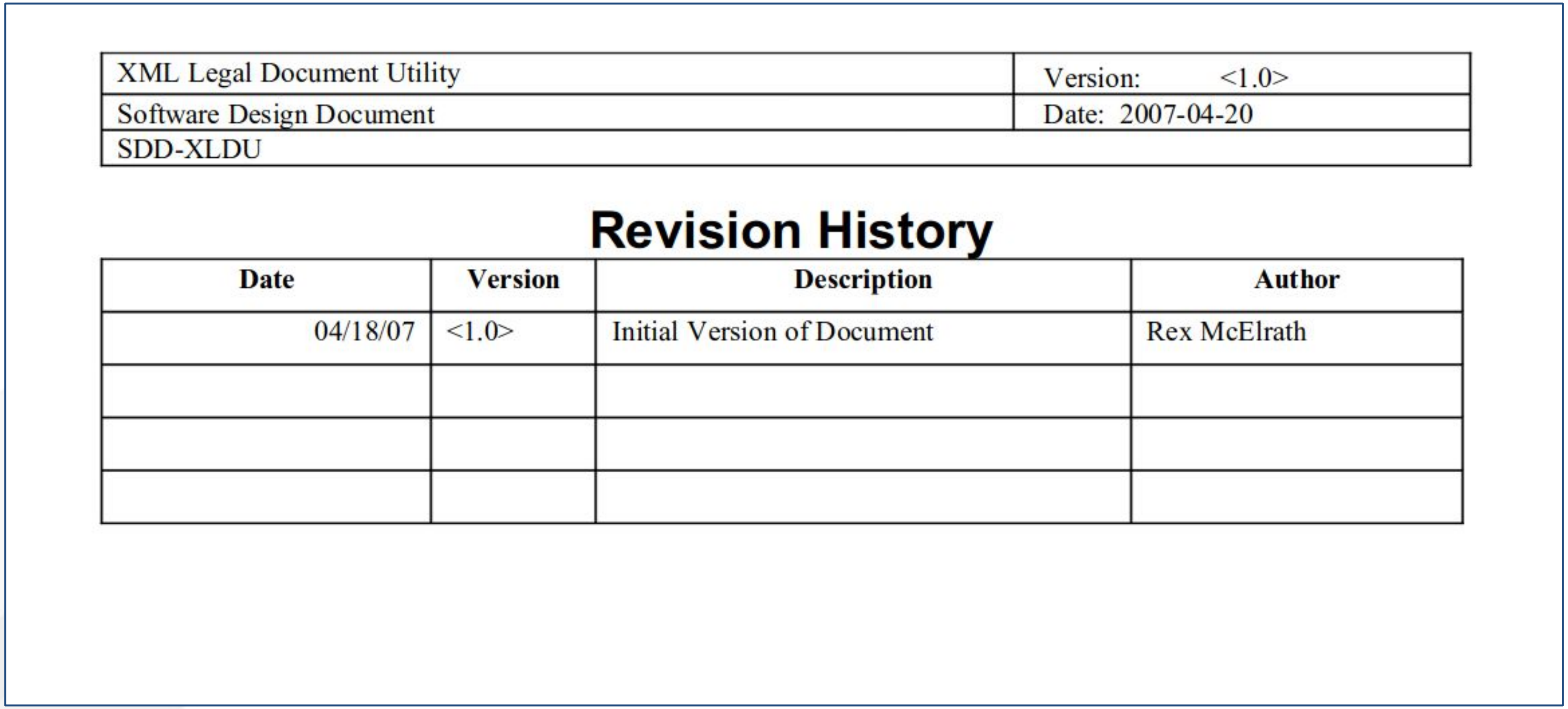
XML Legal Document Utility
Software Design Document

Version <1.0>

Rex McElrath

2007-04-20

Riwayat revisi



XML Legal Document Utility	Version: <1.0>
Software Design Document	Date: 2007-04-20
SDD-XLDU	

Revision History

Date	Version	Description	Author
04/18/07	<1.0>	Initial Version of Document	Rex McElrath

Daftar isi

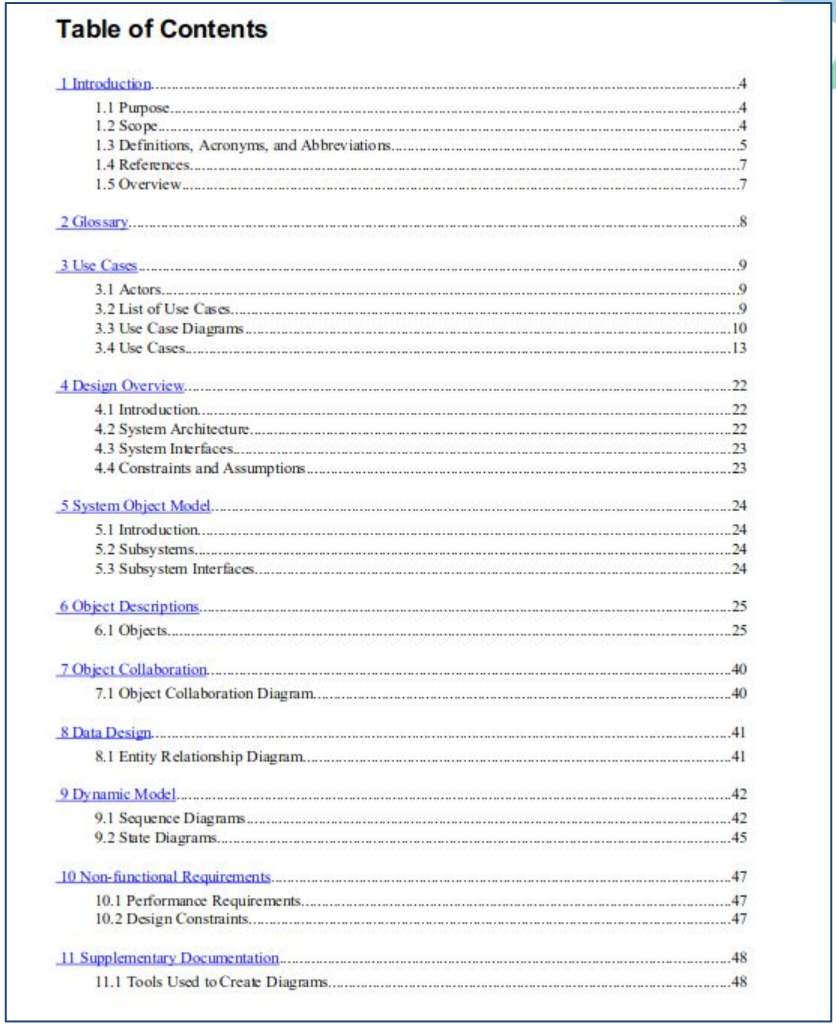
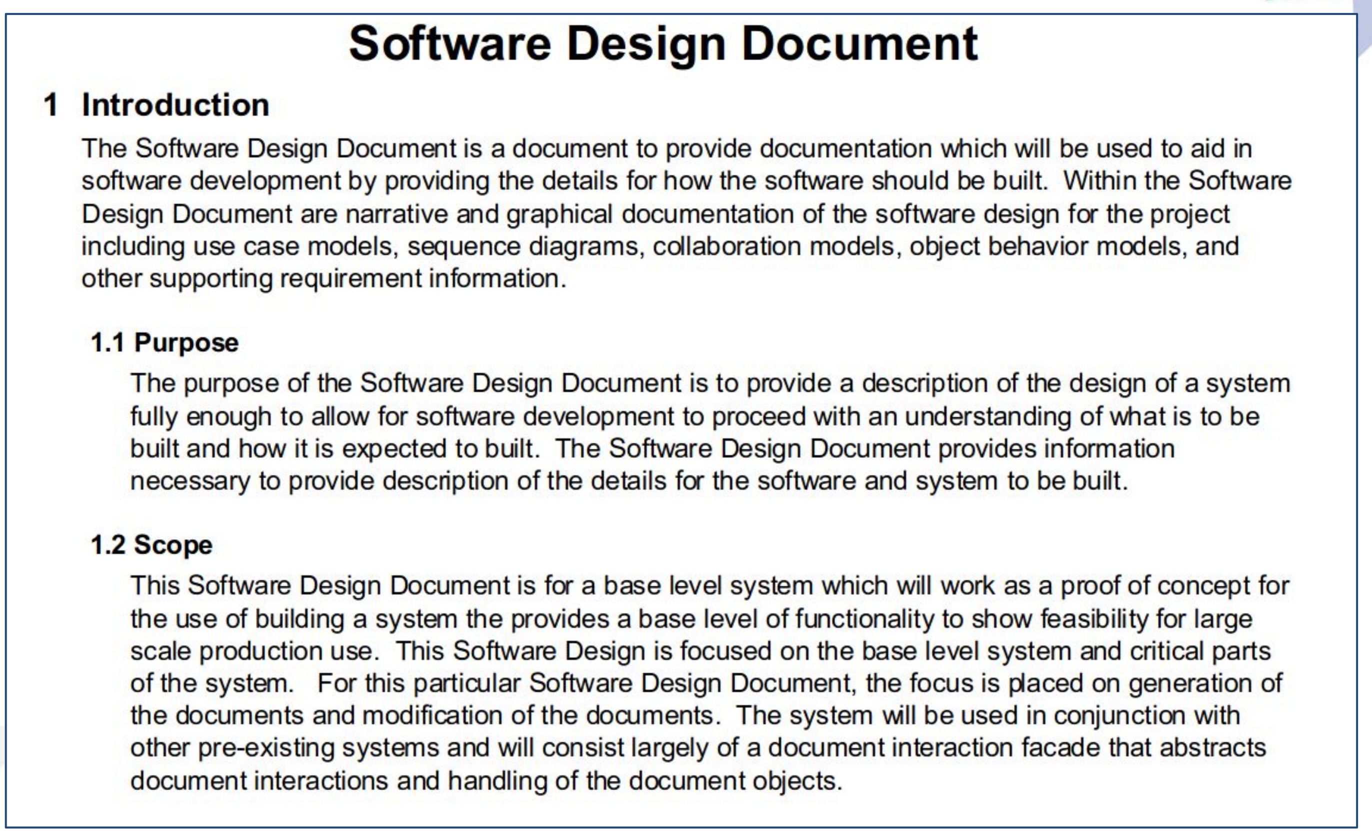


Table of Contents

Pengantar



Software Design Document

1 Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behavior models, and other supporting requirement information.

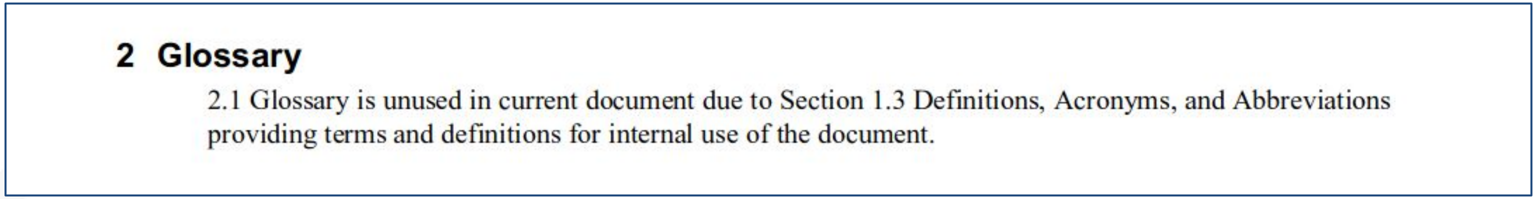
1.1 Purpose

The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to built. The Software Design Document provides information necessary to provide description of the details for the software and system to be built.

1.2 Scope

This Software Design Document is for a base level system which will work as a proof of concept for the use of building a system the provides a base level of functionality to show feasibility for large scale production use. This Software Design is focused on the base level system and critical parts of the system. For this particular Software Design Document, the focus is placed on generation of the documents and modification of the documents. The system will be used in conjunction with other pre-existing systems and will consist largely of a document interaction facade that abstracts document interactions and handling of the document objects.

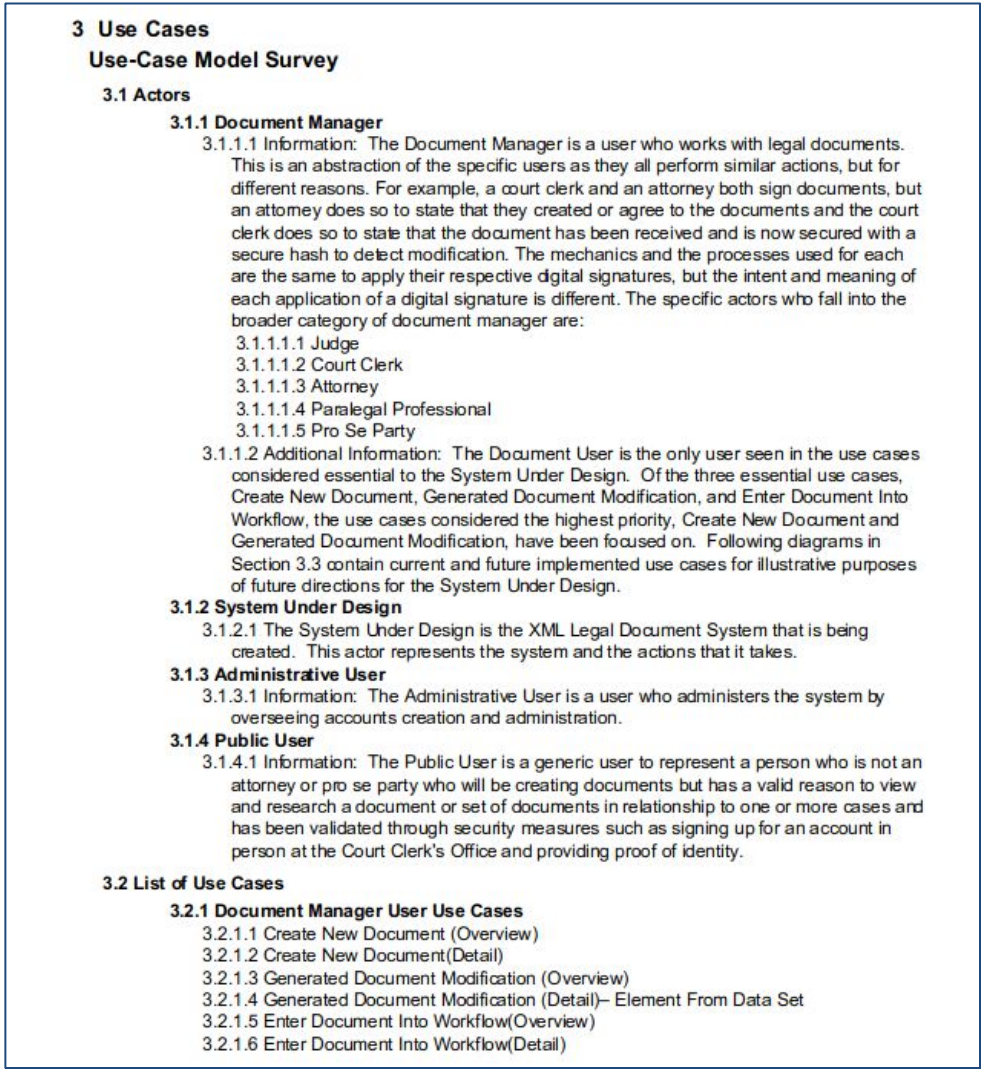
Glosarium



2 Glossary

2.1 Glossary is unused in current document due to Section 1.3 Definitions, Acronyms, and Abbreviations providing terms and definitions for internal use of the document.

Use cases



3 Use Cases

Use-Case Model Survey

3.1 Actors

3.1.1 Document Manager

3.1.1.1 Information: The Document Manager is a user who works with legal documents. This is an abstraction of the specific users as they all perform similar actions, but for different reasons. For example, a court clerk and an attorney both sign documents, but an attorney does so to state that they created or agree to the documents and the court clerk does so to state that the document has been received and is now secured with a secure hash to detect modification. The mechanics and the processes used for each are the same to apply their respective digital signatures, but the intent and meaning of each application of a digital signature is different. The specific actors who fall into the broader category of document manager are:

3.1.1.1.1 Judge
3.1.1.1.2 Court Clerk
3.1.1.1.3 Attorney
3.1.1.1.4 Paralegal Professional
3.1.1.1.5 Pro Se Party

3.1.1.2 Additional Information: The Document User is the only user seen in the use cases considered essential to the System Under Design. Of the three essential use cases, Create New Document, Generated Document Modification, and Enter Document Into Workflow, the use cases considered the highest priority, Create New Document and Generated Document Modification, have been focused on. Following diagrams in Section 3.3 contain current and future implemented use cases for illustrative purposes of future directions for the System Under Design.

3.1.2 System Under Design

3.1.2.1 The System Under Design is the XML Legal Document System that is being created. This actor represents the system and the actions that it takes.

3.1.3 Administrative User

3.1.3.1 Information: The Administrative User is a user who administers the system by overseeing accounts creation and administration.

3.1.4 Public User

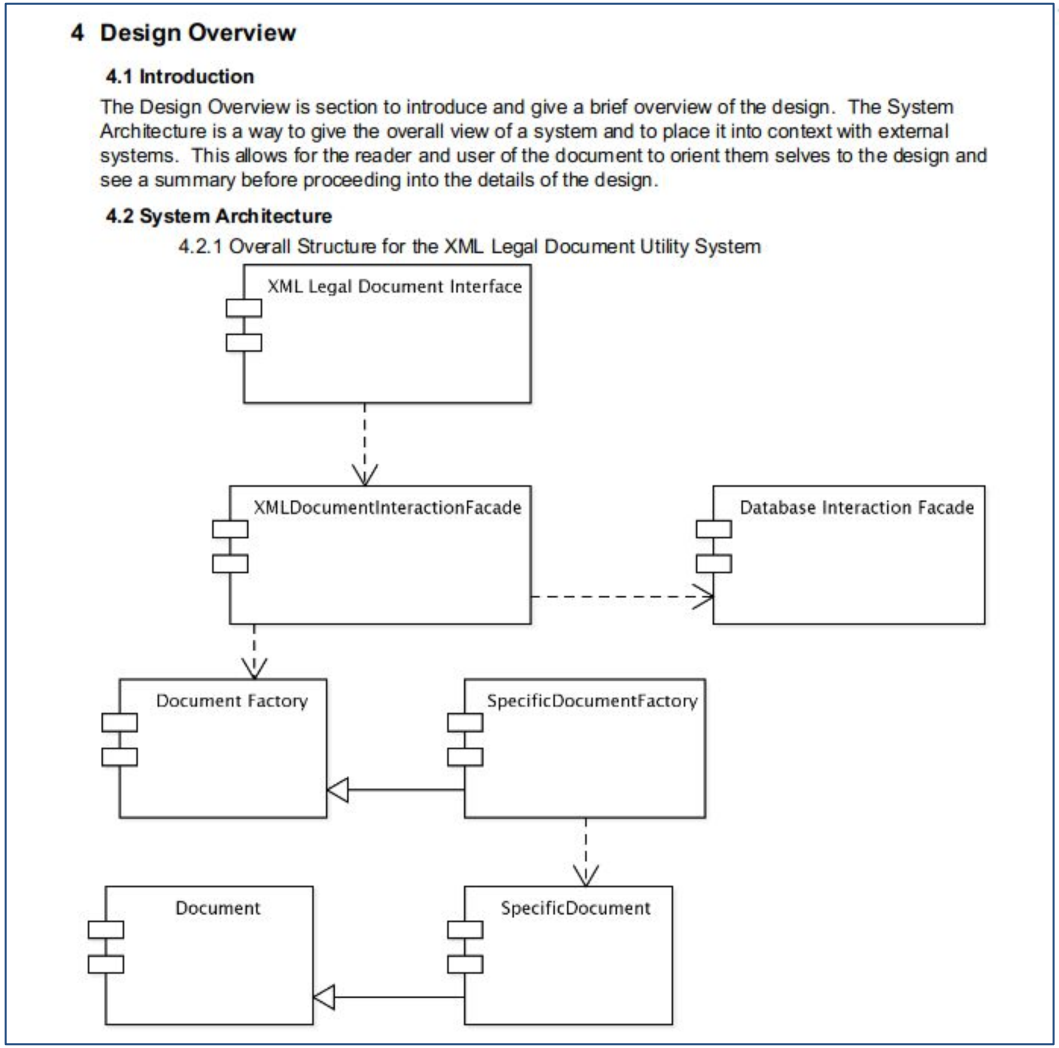
3.1.4.1 Information: The Public User is a generic user to represent a person who is not an attorney or pro se party who will be creating documents but has a valid reason to view and research a document or set of documents in relationship to one or more cases and has been validated through security measures such as signing up for an account in person at the Court Clerk's Office and providing proof of identity.

3.2 List of Use Cases

3.2.1 Document Manager User Use Cases

3.2.1.1 Create New Document (Overview)
3.2.1.2 Create New Document(Detail)
3.2.1.3 Generated Document Modification (Overview)
3.2.1.4 Generated Document Modification (Detail)– Element From Data Set
3.2.1.5 Enter Document Into Workflow(Overview)
3.2.1.6 Enter Document Into Workflow(Detail)

Ikhtisar Desain



4 Design Overview

4.1 Introduction

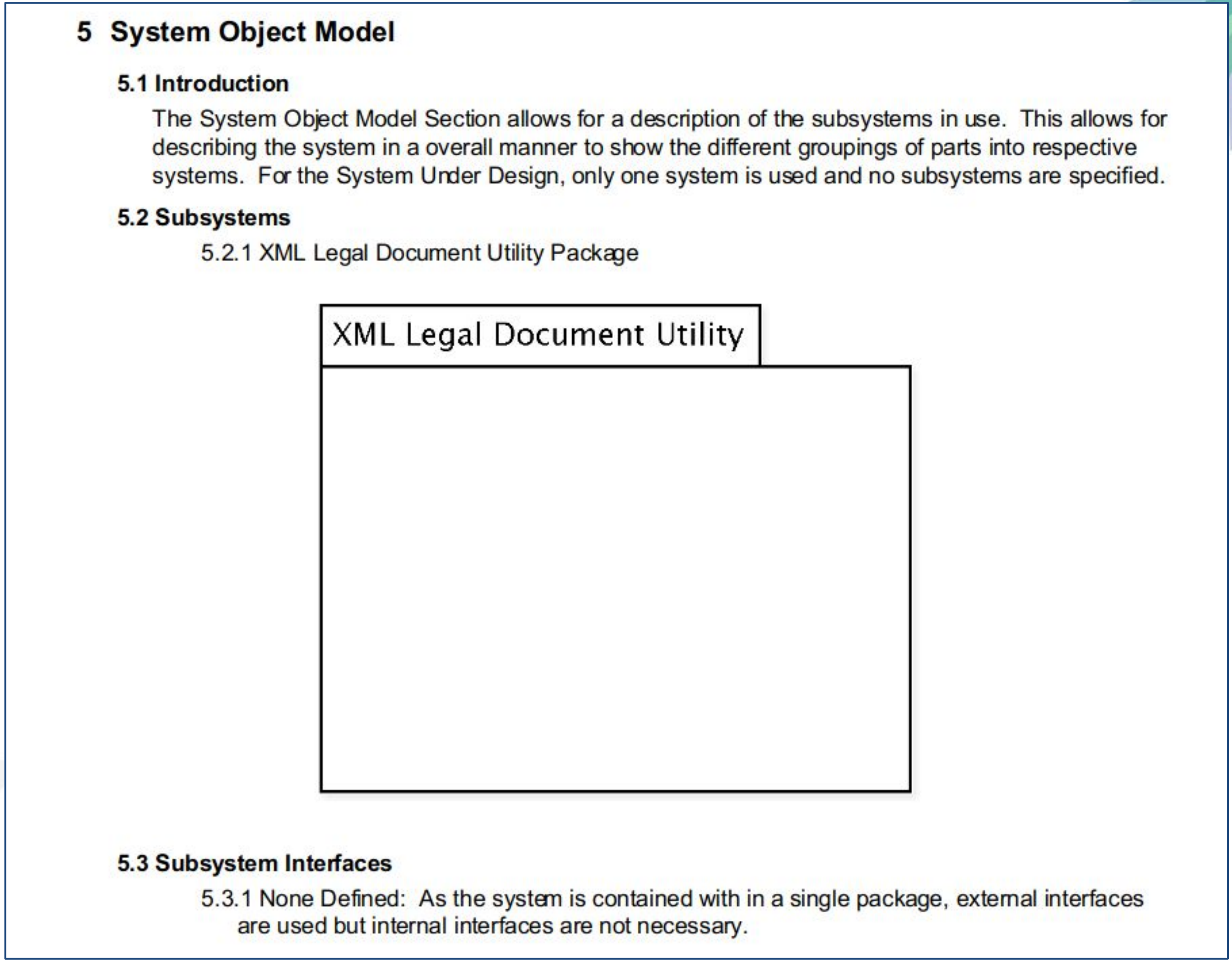
The Design Overview is section to introduce and give a brief overview of the design. The System Architecture is a way to give the overall view of a system and to place it into context with external systems. This allows for the reader and user of the document to orient them selves to the design and see a summary before proceeding into the details of the design.

4.2 System Architecture

4.2.1 Overall Structure for the XML Legal Document Utility System

XML Legal Document Interface
XMLDocumentInteractionFacade
Database Interaction Facade
Document Factory
SpecificDocumentFactory
Document
SpecificDocument

System object model



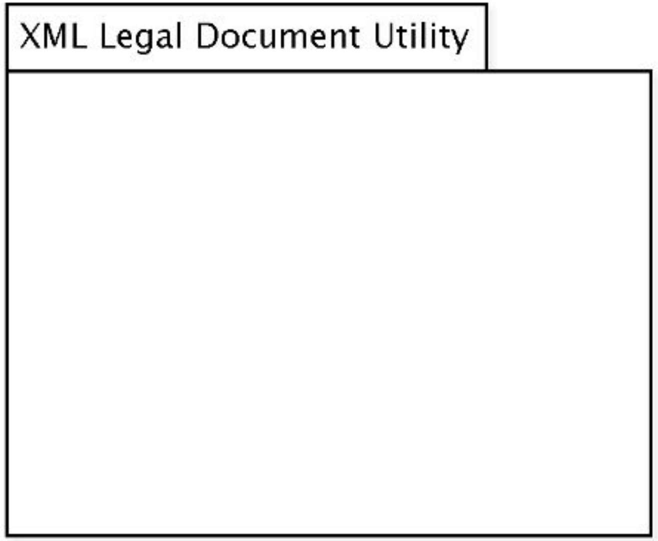
5 System Object Model

5.1 Introduction

The System Object Model Section allows for a description of the subsystems in use. This allows for describing the system in a overall manner to show the different groupings of parts into respective systems. For the System Under Design, only one system is used and no subsystems are specified.

5.2 Subsystems

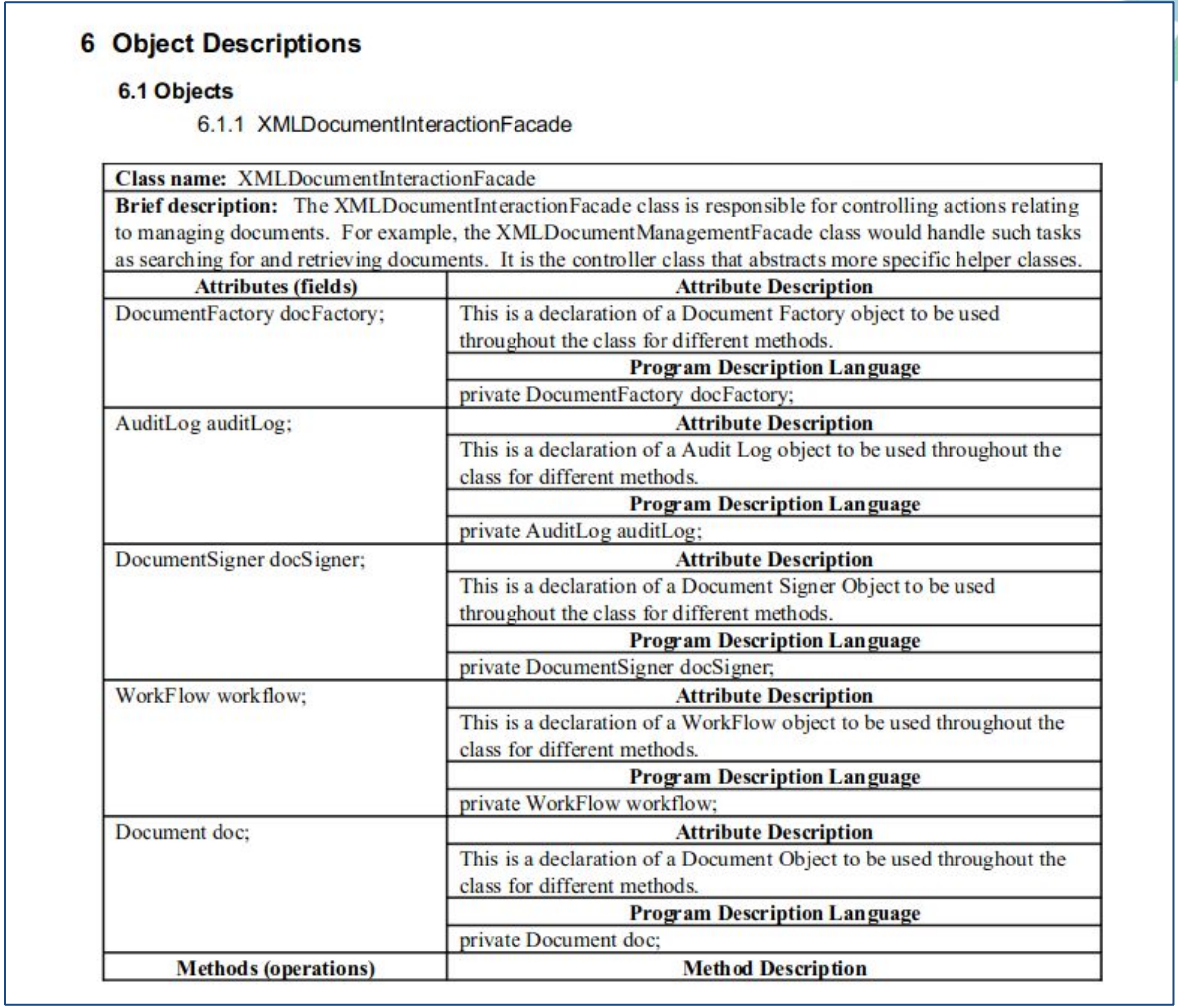
5.2.1 XML Legal Document Utility Package



5.3 Subsystem Interfaces

5.3.1 None Defined: As the system is contained with in a single package, external interfaces are used but internal interfaces are not necessary.

Deskripsi objek



6 Object Descriptions

6.1 Objects

6.1.1 XMLDocumentInteractionFacade

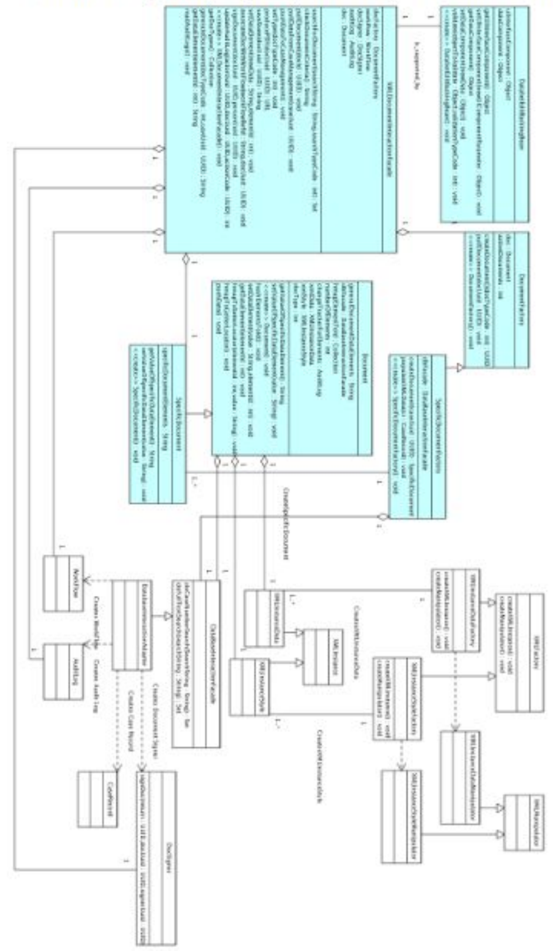
Class name: XMLDocumentInteractionFacade	
Brief description: The XMLDocumentInteractionFacade class is responsible for controlling actions relating to managing documents. For example, the XMLDocumentManagementFacade class would handle such tasks as searching for and retrieving documents. It is the controller class that abstracts more specific helper classes.	
Attributes (fields)	**Attribute Description**
DocumentFactory docFactory;	This is a declaration of a Document Factory object to be used throughout the class for different methods.
	Program Description Language
	private DocumentFactory docFactory;
AuditLog auditLog;	**Attribute Description**
	This is a declaration of a Audit Log object to be used throughout the class for different methods.
	Program Description Language
	private AuditLog auditLog;
DocumentSigner docSigner;	**Attribute Description**
	This is a declaration of a Document Signer Object to be used throughout the class for different methods.
	Program Description Language
	private DocumentSigner docSigner;
WorkFlow workflow;	**Attribute Description**
	This is a declaration of a WorkFlow object to be used throughout the class for different methods.
	Program Description Language
	private WorkFlow workflow;
Document doc;	**Attribute Description**
	This is a declaration of a Document Object to be used throughout the class for different methods.
	Program Description Language
	private Document doc;
Methods (operations)	**Method Description**

Kolaborasi Objek

7 Object Collaboration

7.1 Object Collaboration Diagram

7.1.1 This is a diagram depicting the object relationships. Items in blue are described in Section 6.

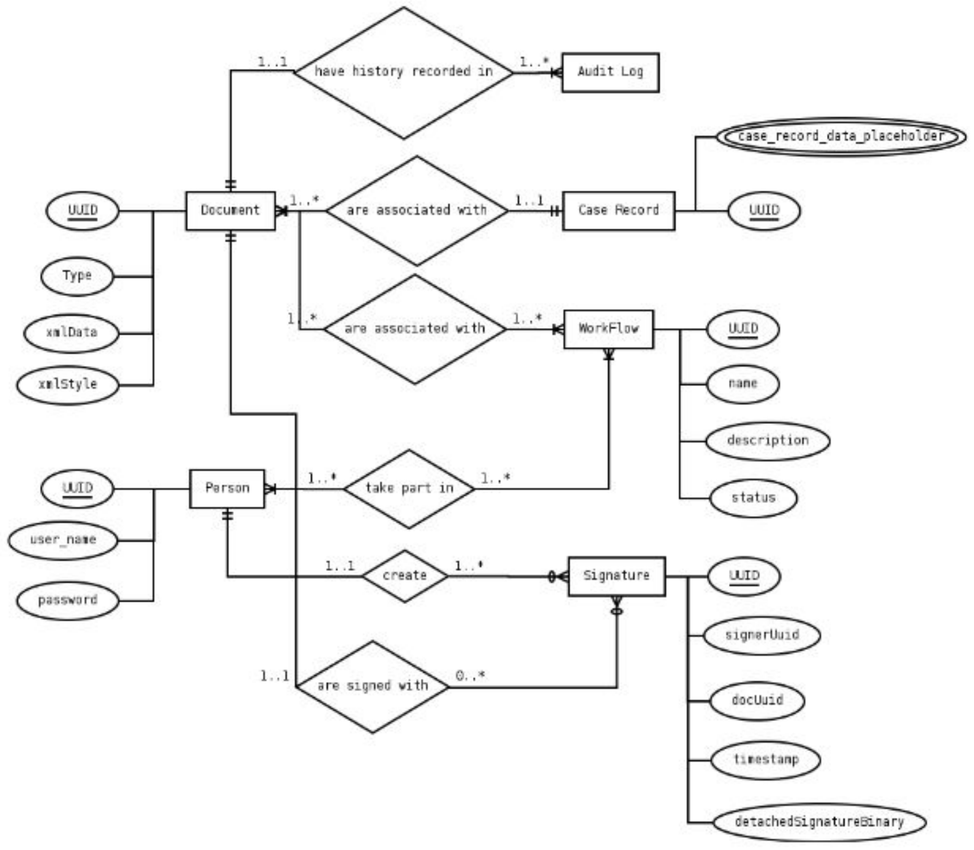


Desain data

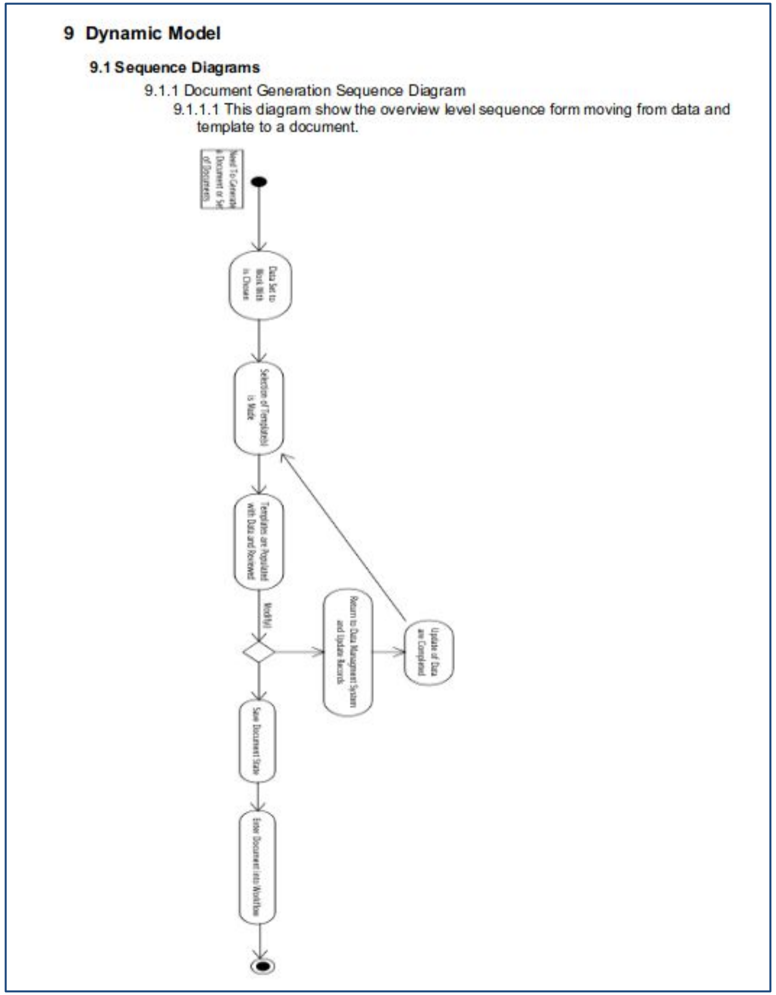
8 Data Design

8.1 Entity Relationship Diagram

8.1.1 Basic Entity Relationship Diagram



Model dinamis



9 Dynamic Model

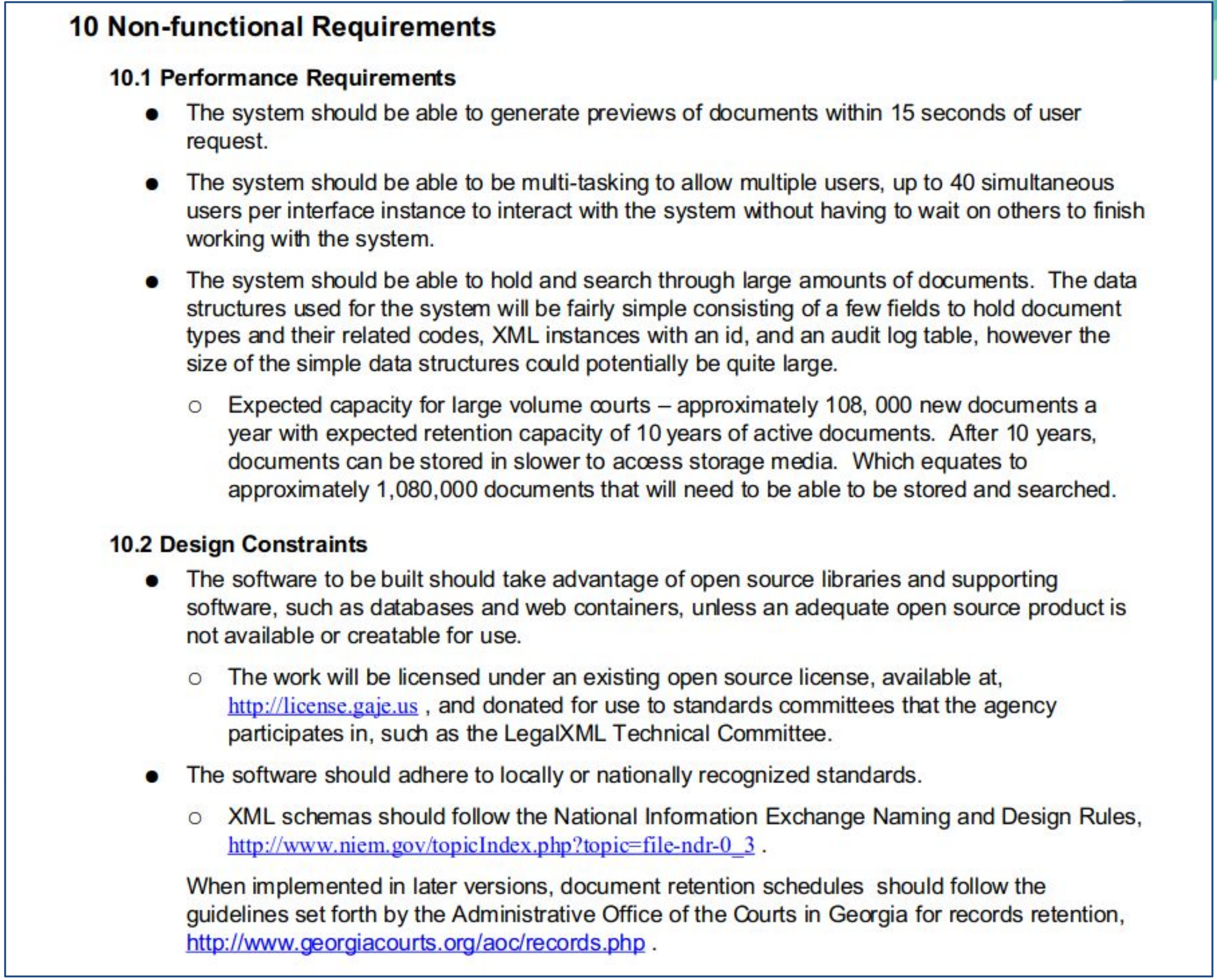
9.1 Sequence Diagrams

9.1.1 Document Generation Sequence Diagram

9.1.1.1 This diagram show the overview level sequence form moving from data and template to a document.

Need To Generate a Document or Set of Documents
Data Set to Work With is Chosen
Selection of Template(s) is Made
Templates are Populated with Data and Reviewed
Workflow
Return to Data Management System and Update Records
Update of Data are Completed
Save Document State
Enter Document into Workflow

Persyaratan non fungsional



10 Non-functional Requirements

10.1 Performance Requirements

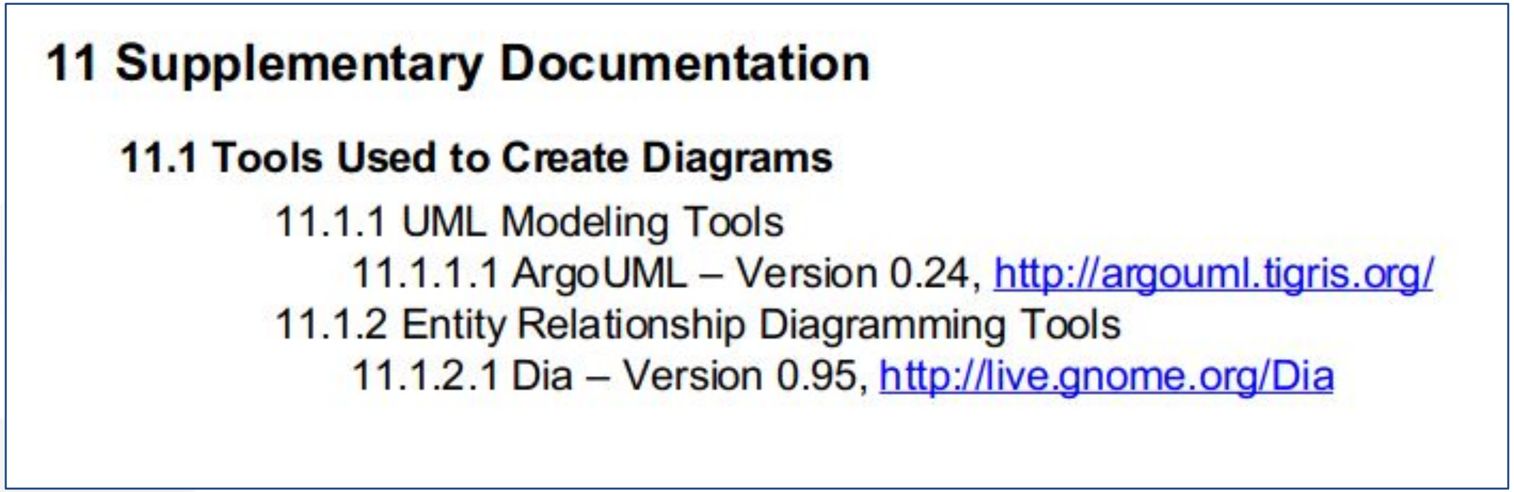
- The system should be able to generate previews of documents within 15 seconds of user request.
- The system should be able to be multi-tasking to allow multiple users, up to 40 simultaneous users per interface instance to interact with the system without having to wait on others to finish working with the system.
- The system should be able to hold and search through large amounts of documents. The data structures used for the system will be fairly simple consisting of a few fields to hold document types and their related codes, XML instances with an id, and an audit log table, however the size of the simple data structures could potentially be quite large.
    - Expected capacity for large volume courts – approximately 108, 000 new documents a year with expected retention capacity of 10 years of active documents. After 10 years, documents can be stored in slower to access storage media. Which equates to approximately 1,080,000 documents that will need to be able to be stored and searched.

10.2 Design Constraints

- The software to be built should take advantage of open source libraries and supporting software, such as databases and web containers, unless an adequate open source product is not available or creatable for use.
    - The work will be licensed under an existing open source license, available at, http://license.gaje.us , and donated for use to standards committees that the agency participates in, such as the LegalXML Technical Committee.
- The software should adhere to locally or nationally recognized standards.
    - XML schemas should follow the National Information Exchange Naming and Design Rules, http://www.niem.gov/topicIndex.php?topic=file-ndr-0_3 .

When implemented in later versions, document retention schedules should follow the guidelines set forth by the Administrative Office of the Courts in Georgia for records retention, http://www.georgiacourts.org/aoc/records.php .

Dokumentasi tambahan



11 Supplementary Documentation

11.1 Tools Used to Create Diagrams

11.1.1 UML Modeling Tools

11.1.1.1 ArgoUML – Version 0.24, http://argouml.tigris.org/

11.1.2 Entity Relationship Diagramming Tools

11.1.2.1 Dia – Version 0.95, http://live.gnome.org/Dia

Referensi

http://lightcastletech.com/five-principles-to-good-documentation-writing-good-documentation-series/

http://robotics.ee.uwa.edu.au/courses/design/examples/example_design.pdf

Terima kasih!