Software Requirements Specifications

for

AJCON, Applet to JSF Converter

Version 1.0

Prepared by Group Teaplet

Berkan KISAOĞLU

Anil SEVİM

Özge TOKGÖZ

05.12.2010

TABLE OF CONTENTS

1. Introduction

This Software Requirements Specification (SRS) document provides a complete description of all the functions and specifications of Applet to Java Server Faces (JSF) Convertor (AJCON) sponsored by Siemens Enterprise Communications (Siemens EC).

The expected audience of this document is Siemens EC and its partners who wants to convert their Applet projects to JSF ones.

1.1. Problem Definition

Java Applets can provide web applications with interactive features that cannot be provided by HTML. When Java enabled browser is used to view a page that contains an applet, the applets byte-codes are transferred to user's system and executed by browser's Java Virtual Machine (JVM). In those days, applet technology has become out of date. Meanwhile, with new Java 2 Enterprise Edition (J2EE) technologies, same functional requirements can be met with less dependency. JSF is one of these new technologies. In this project, we will deal with this problem, old applet technology based applications.

1.2. Purpose

This document includes software requirements for AJCON project for use of converting Applets to JSF, release number 1.0. Further requirements, features and functionalities of AJCON will be explained and analyzed in detail throughout this SRS document.

1.3. Scope

This is an SRS document for specifying AJCON project requirements. It describes the content, qualities and functions of the software to be developed.

This document provides characteristics and features of procedures to be followed in developing the software. During this process, visual diagrams and models will be used in order to help audience of this SRS. For further developments on this project, all software requirements are tried to be stated clearly.

1.4. User and Literature Survey

There exists no such software doing exactly the same functionality of ours. On the other hand, there are some softwares that have the capability of doing some functionalities of our project.

There exists a product named "HTML Applet to Object Tag Converter" and "Microsoft J# Browser Controls" which allows developers to migrate Java Applets that were written and run on a JVM, to .NET framework. There are also some other products converting an applet to other technologies such as "Awn".

Potential users of our product are the enterprises that have products using Java Applets. Today there are 39 Technopolises, 24 Technology Incubator, over 1100 IT companies in Turkey [2]. All of those companies will be potential users of our products.

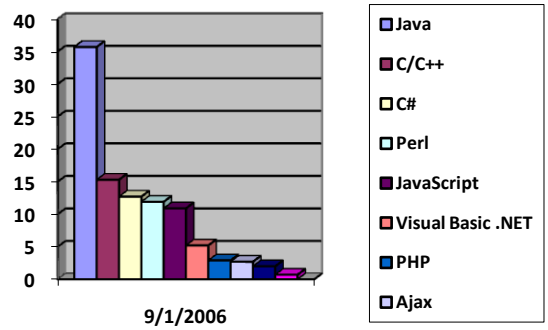


Figure 1

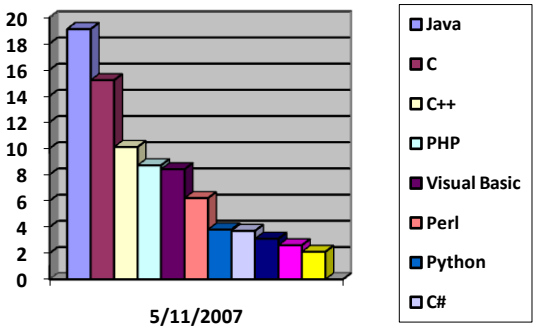


Figure 2

Also according to surveys made by some international organizations, it is proven that Java Technologies are the mostly used technology over other technologies [3]. First survey mentioned in Figure 1 is made by Dice.com, and the second survey mentioned in Figure 2 is made by Tiobe.com. All those surveys made among the developers to measure the popular technologies are in use by the developer's company. Since we are making a converter for Java Applets to Java Server Faces, there will be highly demand in market.

1.5. Definitions and Abbreviations

SRS	Software Requirements Specifications
AJCON	Applet to JSF Convertor
JVM	Java Virtual Machine
Siemens EC	Siemens Enterprise Communications
JSF	Java Server Faces
J2EE	Java 2 Enterprise Edition

1.6. References

[1]	IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
[2]	METU Technopolis, `http://www.metutech.metu.edu.tr`
[3]	Populer Programming Languages, `http://www.devtopics.com/most-popular-programming-languages/`
[4]	Java Naming Conventions, `http://www.oracle.com/technetwork/java/codeconventions-135099.html`

1.7. Overview

General outline of the remaining part of the document is so that:

- In section 2, an overall description of AJCON project is provided. First product perspective and product functions are given with related visual models: Component Diagram, Use Case Diagram. Then, technology constraints, assumptions and dependencies are stated.
- In section 3, specific external and internal requirements are stated. Interfaces are associated with the use cases drawn in section 2. Also, product function and system

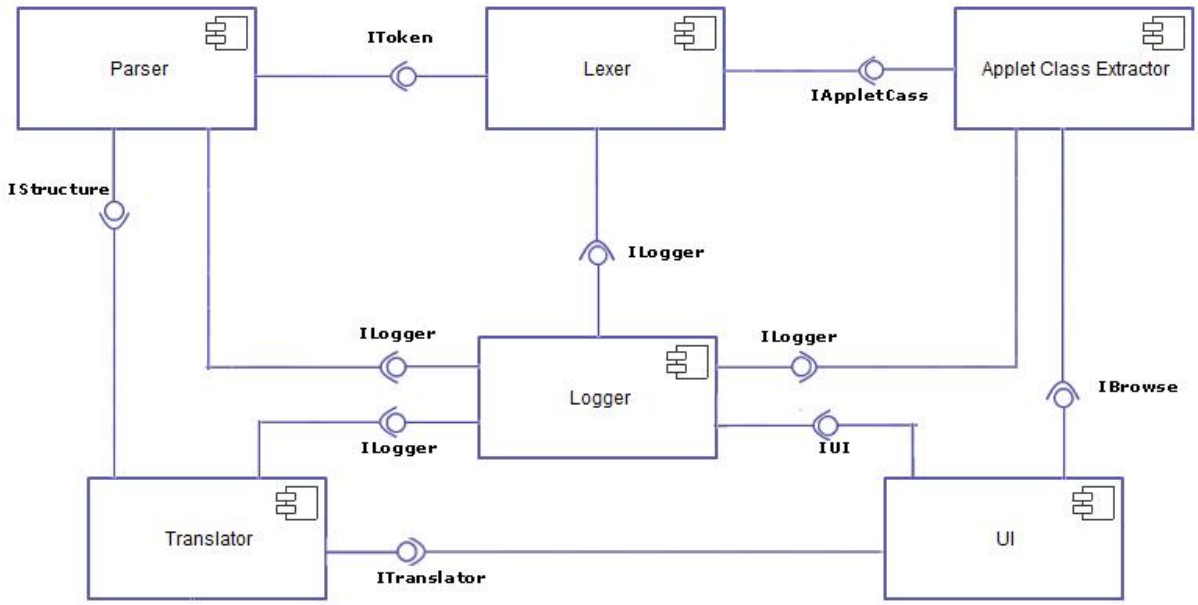
features are explained in detail. For each product function specific requirements and basic data flows are stated.

- In section 4, Data objects that will be managed/manipulated by the software are described.
- In section 5, behavior of the system stated with an activity diagram and state transition diagram about the general states that system will possibly in.
- In section 6, general planning and estimation about the project mentioned. First team structure mentioned in detail and the roles are defined. Then, estimation stated about the AJCON with a visual Gantt chart.
- In section 7, we ended up Software Requirements Specifications Document.

2. Overall Description

2.1. Product Perspective

AJCON project will be an independent and self-contained project developing by Group Teaplet. Project will not be a subsystem of an existing product. Main concern of the project is to convert a Java Applet to JSF. For this purpose, It includes several components: Applet Class Extractor, Lexer, parser, translator, logger, UI.



Component Diagram

Above, relations between components are shown. Logger Component provides ILogger interface to other components except UI component. This interface provides logging functionalities to related components. At the same time, Logger Component uses the IUI interface provided by UI Component. Logger will use this interface for sending log information to the UI.

UI Component also provides IBrowse interface to Applet Class Extractor Component. Inputs taken from UI Component are sent to Applet Class Extractor Component throughout this interface. Moreover, UI triggers Translator Component via ITranslator interface to start conversion.

There exist several interfaces between Applet Class Extractor Component, Lexer Component, Parser Component, and Translator Component respectively. Using these interfaces, one component provides its own processed data to another for its usage. At last, the output of Translator Component will be the converted project located in destination folder.

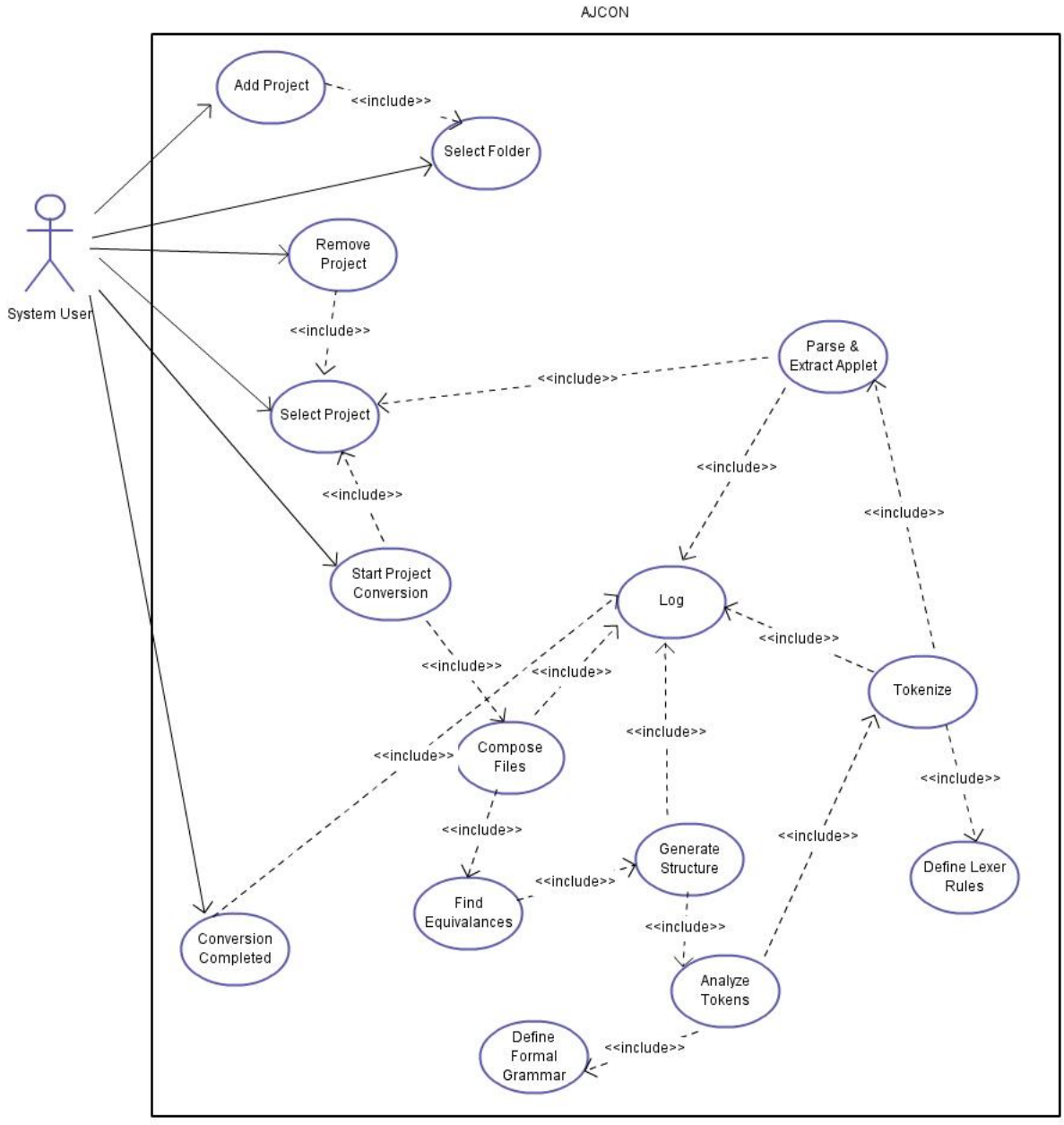
2.2. Product Functions

General functions which can describe the overall system functionalities are explained below.

- Applet Class Extractor Component
    - parseAndExtractApplet( ): Parses whole project and finds Applet classes among .java files.
- Lexer Component
    - defineLexerRules( ): Defining a set of possible character sequences that are used to form individual tokens or lexemes for Java Applet classes.
    - tokenize( ): The process of forming tokens from an input stream of characters (Java Applet classes) according to the pre-defined lexer rules.
- Parser Component
    - defineFormalGrammar( ): Defining a grammar for evaluating lexemes compatible with Java Applet classes.
    - analyzeTokens( ): Syntactic analysis of tokens according to pre-defined grammar.
    - generateStructure( ): Builds a hierarchical structure from analyzed group of tokens.
- Translator Component
    - findEquivalences ( ): Traverses the parse tree created by the parser component and finds the equivalences of applet elements in JSF.
    - composeFiles( ): According to found equivalences, compose JSF and other related java files.
- Logger Component
    - log(Object obj): Logs several information (Date/Time, Function of any component, process) of Object obj.
- UI Component
    - newProject( ): Adds a new project to convert.
    - selectProject( ): Selects a project from the project list to make operations.
    - removeProject( ): Removes the project.
    - selectFolder( ): Used for selecting source and destination folders.
    - startProjectConversion( ): Starts conversion operation

- conversionCompleted( ): Pops up a dialog box indicating that conversion is completed.

There is no specific user class; any user can use the system. Below there exists use case diagram of AJCON.



Use Case Diagram

2.3. Constraints, Assumptions and Dependencies

- Constraints: In our project, JSF 2.0 will be used in converted project.
- Assumptions: We assume that project which will be converted should be syntactically correct and runnable. Also, that project should include at least one applet class. When environment is considered, we assume that JRE should be installed on that running computer.
- Dependencies: JSF will depend on Java SE 5 (or higher) which is required to run this project.
- Safety and Security Considerations: While converting, there will be no changes in source project. Output project will be in a different directory. Therefore, if any error occurs while conversion, original project will remain the same.

There is no time, memory, OS, network and hardware limitation.

3. Specific Requirements

3.1. Interface Requirements

3.1.1. External Interface Requirements

3.1.1.1. User Interfaces

In our project, user interaction between the systems will not be so complicated because most actions will be relying on background processes.

(There are initial screen designs below. They may evolve with alternate requirements.)

While using the system, firstly user will see main user interface of the project.

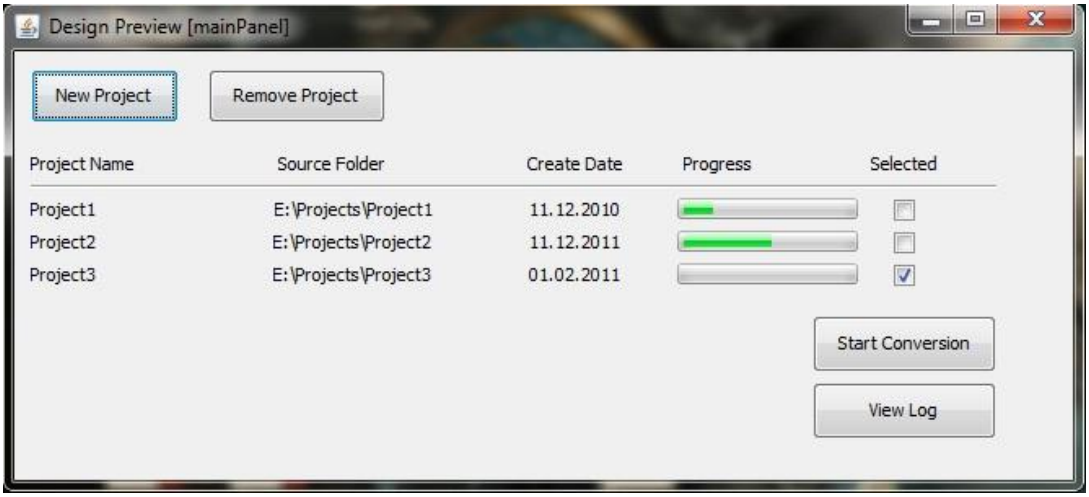


Figure 3 – Main Interface

In this UI, user can manage projects to be converted (select, add, remove project) by UI components (button, check box). Moreover, user can initiate conversion process after selecting project. By using this UI, user can also view information about projects and progress of each project's conversion. By a button, user can view log of the conversion in another window.

Figure 4 – Log Interface

When user wants to add new project on the main window, new user interface is opened. In this UI, user can browse his/her computer file system to find project. Also, user should select destination folder to place output project due to safety considerations.

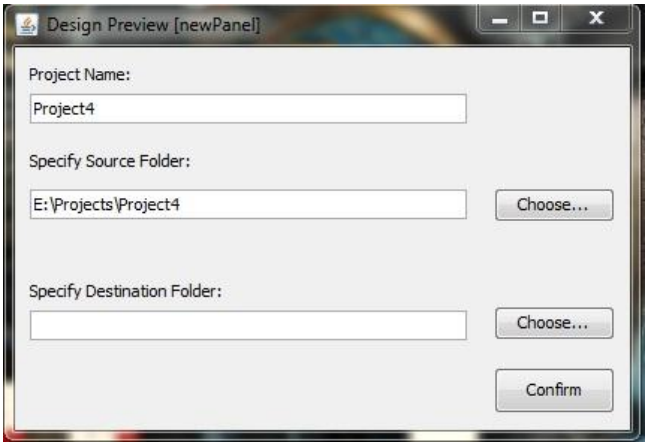


Figure 5 – Project Interface

As shown in component diagram (section 2.1), UI component of the system only interacts with Logger and Applet Class Extractor components. When user adds a project and selects destination folder, these parameters are taken as input of system in UI component. When user selects "Start Conversion", UI will communicate with Applet Class Extractor component and send inputs to this component. Considering Logger component, log information for each project will be sent to UI component and user can see them if he/she wants. Also, progress information will be taken from Logger component.

Associations with Use Cases

Use case – User Interface relations will be explained below.

- Main Interface
    - Add Project: User will be able to add new project to system. This use case includes "Select Folder" use case.
    - Select Project: User will be able to select a project among a list of projects.
    - Remove Project: User will be able to remove selected project from the list. This use case includes "Select Project" use case.

  - Start Project Conversion: User will be able to convert operation of selected projects. This use case includes “Select Project” and “Compose Files” use case to initiate conversion.
  - Conversion Completed: User will be able to be alerted when conversion completed. This use case includes “Log” use case to get information about completion.
- Log Interface
  - Log: User will be able to see the information about the operations information.
- Project Interface
  - Select Folder: This use case is replaced by “Specify Source Project Folder” or “Specify Destination Project Folder”.

The remaining use cases shown in 2.2 have not directly an interaction with user. These are all related to system functions and these are explained in section 3.2.

3.1.1.2. Other System Interfaces

As an external Interface, there exists only User Interface. Hardware Interfaces, Software Interfaces with other software products and Communication Interface does not exist. However, as software interface between external packages and our system will be decided later. (In Design Stage)

3.2. System Features with Functional Requirements

System features are organized according to components of AJCON. Details of each major function will be stated below.

3.2.1. UI Component Functions

3.2.1.1. newProject( )

3.2.1.1.1. Purpose of feature

This feature provides to add a new project to existing project list to convert. This feature has high priority among the others because, without adding a new project, conversion cannot start.

3.2.1.1.2. Stimulus/Response Sequence

Data Flow:

3.2.1.1.2.1. Basic Data Flow

1. User starts AJCON.

2. User clicks “New Project” button.

3. A window that contains the details of the project will appear

4. User selects Source and Destination folders of AJCON. (selectFolder())

5. User confirms by clicking “Confirm” button.

3.2.1.1.2.2. Alternative Data Flow 1

5. User cancels by clicking “X” button.

3.2.1.1.2.3. Alternative Data Flow 2

4. User selects only Source folder of AJCON.

5. User confirms by clicking “Confirm” button.

6. An alert appears that indicates the Destination folder will be selected.

3.2.1.1.2.4. Alternative Data Flow 3

4. User selects only Destination folder of AJCON.

5. User confirms by clicking “Confirm” button.

6. An alert appears that indicates the Source folder will be selected.

3.2.1.1.2.5. Alternative Data Flow 4

4. User selects only Source folder of AJCON.

5. User cancels by clicking “X” button.

3.2.1.1.2.6. Alternative Data Flow 5

4. User selects only Destination folder of AJCON.

5. User cancels by clicking “X” button.

3.2.1.1.3. Associated Functional Requirements

According to the assumptions made in section 2.3, there exists at least one project contains an applet class.

3.2.1.2. selectFolder( )

3.2.1.2.1. Purpose of feature

This feature is used for selecting Source and Destination folders for adding new projects. This function has a high priority, because without selecting folder, adding new project operation cannot be completed.

3.2.1.2.2. Stimulus/Response Sequence

Data Flow:

3.2.1.2.2.1. Basic Data Flow

1. User clicks “Choose…” button.

2. A File/Folder Chooser will appear.

3. User selects folder.

3.2.1.2.2.2. Alternative Data Flow

3. User clicks “X” button and does not selects any folder.

3.2.1.2.3. Associated Functional Requirements

“New Project” window should be appeared to select folders. (newProject() function should be called before)

3.2.1.3. selectProject( )

3.2.1.3.1. Purpose of feature

This feature is used to select a project from the project list to make operations.

3.2.1.3.2. Stimulus/Response Sequence

Data Flow:

3.2.1.3.2.1. Basic Data Flow

1. User clicks an empty checkbox for each project to select.

2. Project will be selected.

3.2.1.3.2.2. Alternative Data Flow

1. User clicks a selected checkbox to deselect the project.

2. Project will be deselected.

3.2.1.3.3. Associated Functional Requirements

There exists at least one project in the project list.

3.2.1.4. removeProject( )

3.2.1.4.1. Purpose of feature

Removes the selected project from the list of projects. Removed project will no longer exist in the list.

3.2.1.4.2. Stimulus/Response Sequence

Data Flow:

3.2.1.4.2.1. Basic Data Flow

1. User clicks “Remove Project” button.

2. Project will no longer exist in the list.

3.2.1.4.2.2. Alternative Data Flow

1. User clicks “Remove Project” button.

2. An alert appears that indicates that at least one project should be selected.

3.2.1.4.3. Associated Functional Requirements

There exist at least one project should be selected in the list of projects. (selectProject())

3.2.1.5. startProjectConversion( )

3.2.1.5.1. Purpose of feature

This feature starts the conversion operation with a trigger coming from the UI.

3.2.1.5.2. Stimulus/Response Sequence

Data Flow:

3.2.1.5.2.1. Basic Data Flow

1. User clicks “Start Conversion” button.

2. Translator Component is waked up by UI.

3. Necessary parameters passed to Applet Class Extractor Component.

3.2.1.5.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.1.5.3. Associated Functional Requirements

There exits at least one project selected in the list of projects. (selectProject())

3.2.1.6. conversionCompleted( ):

3.2.1.6.1. Purpose of feature

Pops up a dialog box indicating that conversion is completed.

3.2.1.6.2. Stimulus/Response Sequence

Data Flow:

3.2.1.6.2.1. Basic Data Flow

1. Translator Component logs that the conversion is completed.

2. UI Component gets information about the completed conversion from Logger Component.

3. UI Component pops up a dialog box indicates that the conversion is completed.

3.2.1.6.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.1.6.3. Associated Functional Requirements

Translation operation should be done successfully and Translator component should log information about completion.

3.2.2. Applet Class Extractor Component Functions

3.2.2.1. parseAndExtractApplet( )

3.2.2.1.1. Purpose of feature

This feature parses the whole project files and extracts applet related files/classes.

3.2.2.1.2. Stimulus/Response Sequence

Data Flow:

3.2.2.1.2.1. Basic Data Flow

1. Applet Class Extractor Component gets the necessary parameters (folder paths, etc.) from the UI.

2. Expands all folders and looks for Applet related classes.

3. Passes those classes to Lexer Component.

3.2.2.1.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.2.1.3. Associated Functional Requirements

Necessary parameters that will come from the UI should be successfully acquired.

3.2.3. Lexer Component Functions

3.2.3.1. defineLexerRules( )

3.2.3.1.1. Purpose of feature

Defining lexer rules for Java Language to tokenize character sequences.

3.2.3.1.2. Stimulus/Response Sequence

Data Flow:

3.2.3.1.2.1. Basic Data Flow

1. Defines rules and assign keys to each token.

3.2.3.1.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.3.1.3. Associated Functional Requirements

Defined rules should be compatible with Java Language Syntax.

3.2.3.2. tokenize( )

3.2.3.2.1. Purpose of feature

The process of forming meaningful tokens according to pre-defined rules.

3.2.3.2.2. Stimulus/Response Sequence

Data Flow:

3.2.3.2.2.1. Basic Data Flow

1. Lexer Component gets the Applet Class files from Applet Class Extractor Component.

2. Tokenizes (create tokens) from those classes according to pre-defined lexer rules.

3. Passes obtained tokens to Parser Component.

3.2.3.2.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.3.2.3. Associated Functional Requirements

Lexer rules should be defined (defineLexerRules()) and Applet Class files should be acquired successfully from Applet Class Extractor Component (parseAndExtractApplet()).

3.2.4. Parser Component Functions

3.2.4.1. defineFormalGrammar( )

3.2.4.1.1. Purpose of feature

This feature defines formal grammar with created tokens according to Java Applet Class.

3.2.4.1.2. Stimulus/Response Sequence

Data Flow:

3.2.4.1.2.1. Basic Data Flow

1. Defines language grammar for generalizing tokens.

3.2.4.1.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.4.1.3. Associated Functional Requirements

Defined formal grammar should be compatible with Java Language Syntax.

3.2.4.2. analyzeTokens( )

3.2.4.2.1. Purpose of feature

Analyzes tokens according to pre-defined formal grammar and constructs meaningful structure.

3.2.4.2.2. Stimulus/Response Sequence

Data Flow:

3.2.4.2.2.1. Basic Data Flow

1. Parser Component gets tokens from Lexer Component.

2. Anaylzes the tokens.

3. Calls generateStructure() function.

3.2.4.2.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.4.2.3. Associated Functional Requirements

Formal grammar should be defined (defineFormalGrammar()) and tokens should be acquired successfully from Lexer Component (tokenize()).

3.2.4.3. generateStructure( )

3.2.4.3.1. Purpose of feature

Puts analyzed group of tokens into a hierarchical structure.

3.2.4.3.2. Stimulus/Response Sequence

Data Flow:

3.2.4.3.2.1. Basic Data Flow

1. Gathers the analyzed tokens.

2. Processes and puts tokens and values into structure.

3.2.4.3.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.4.3.3. Associated Functional Requirements

analyzeTokens() function should be called before.

3.2.5. Translator Component Functions

3.2.5.1. findEquivalences ( )

3.2.5.1.1. Purpose of feature

Traverses the structures created by Parser Component and find the equivalences of the tree nodes in a pre-defined dictionary.

3.2.5.1.2. Stimulus/Response Sequence

Data Flow:

3.2.5.1.2.1. Basic Data Flow

1. Opens the dictionary that is present in XML format.

2. Finds the equivalent expression.

3.2.5.1.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.5.1.3. Associated Functional Requirements

Nodes in the parse tree structure should be acquired successfully (generateStructure()).

3.2.5.2. composeFiles( )

3.2.5.2.1. Purpose of feature

Writes found equivalences into a new file as a new project.

3.2.5.2.2. Stimulus/Response Sequence

Data Flow:

3.2.5.2.2.1. Basic Data Flow

1. Gathers equivalences

2. Write into file in the Destination folder as JSF.

3.2.5.2.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.5.2.3. Associated Functional Requirements

There should be equivalences for nodes in the structure (findEquivalences()).

3.2.6. Logger Component Functions

3.2.6.1. log(Object obj)

3.2.6.1.1. Purpose of feature

This feature logs the information that comes with the object.

3.2.6.1.2. Stimulus/Response Sequence

Data Flow:

3.2.6.1.2.1. Basic Data Flow

1. Gathers object to log information about it.

2. Gathers information from the object.

3. Send log information to UI.

3.2.6.1.2.2. Alternative Data Flow

No alternative data flow exists.

3.2.6.1.3. Associated Functional Requirements

Log object should be acquired successfully to get information as a parameter.

3.3. Non-functional Requirements

3.3.1. Performance requirements

- There is no limitation on the number of existing projects.
- There is no limitation on the number of converting projects.
- To have a better performance, it is required to convert a small number of projects at a time.
- It is compatible with all types of computer hardware and software that meets the constraints, assumptions and dependencies stated in section 2.3.

3.3.2. Security Requirements

Since there is no specific user class for AJCON project, all the users have the software on their computers can use the product. So there is no security policy applied on the AJCON project.

3.3.3. Logical Database Requirements

There is no logical database requirement. All the necessary information will be kept in XML format.

3.3.4. Design constraints

Java will be used as the programming language while developing the product. Also external products might be used if needed.

There is no hardware constraint.

3.3.4.1. Standards Compliance

All the documents/reports of the project will conform to IEEE specifications.

As a naming convention for data in our programming section, standard Java Naming Conventions will be used. [4]

3.3.4.2. Software System Attributes

3.3.4.2.1. Adaptability

There will be no extra effort to adapt AJCON into different platforms. Since AJCON is a Java project, it will work on the computers machine independently.

3.3.4.2.2. Availability

Main requirement is to have AJCON is to be a customer of Siemens-EC.

3.3.4.2.3. Correctness

System will work correctly if all the requirements, assumptions and dependencies are met. AJCON will give the same result with the same inputs in different times. On the other hand, AJCON will evolve from converting simple applets to converting more complex. So, there is no 100% conversion.

3.3.4.2.4. Maintainability

AJCON project can respond new requirements as the time passes. New features can also be added.

3.3.4.2.5. Portability

There will be an installer for AJCON. It can be moved on any computer with the installer.

4. Data Model and Description

In this section, we will describe how we organize the data used to communicate between different components of system and external environment.

4.1. Data Description

In our project, we will not use any kind of database structure. Instead, we will keep our data in simple file structures, e.g. XML file.

4.1.1. Data objects

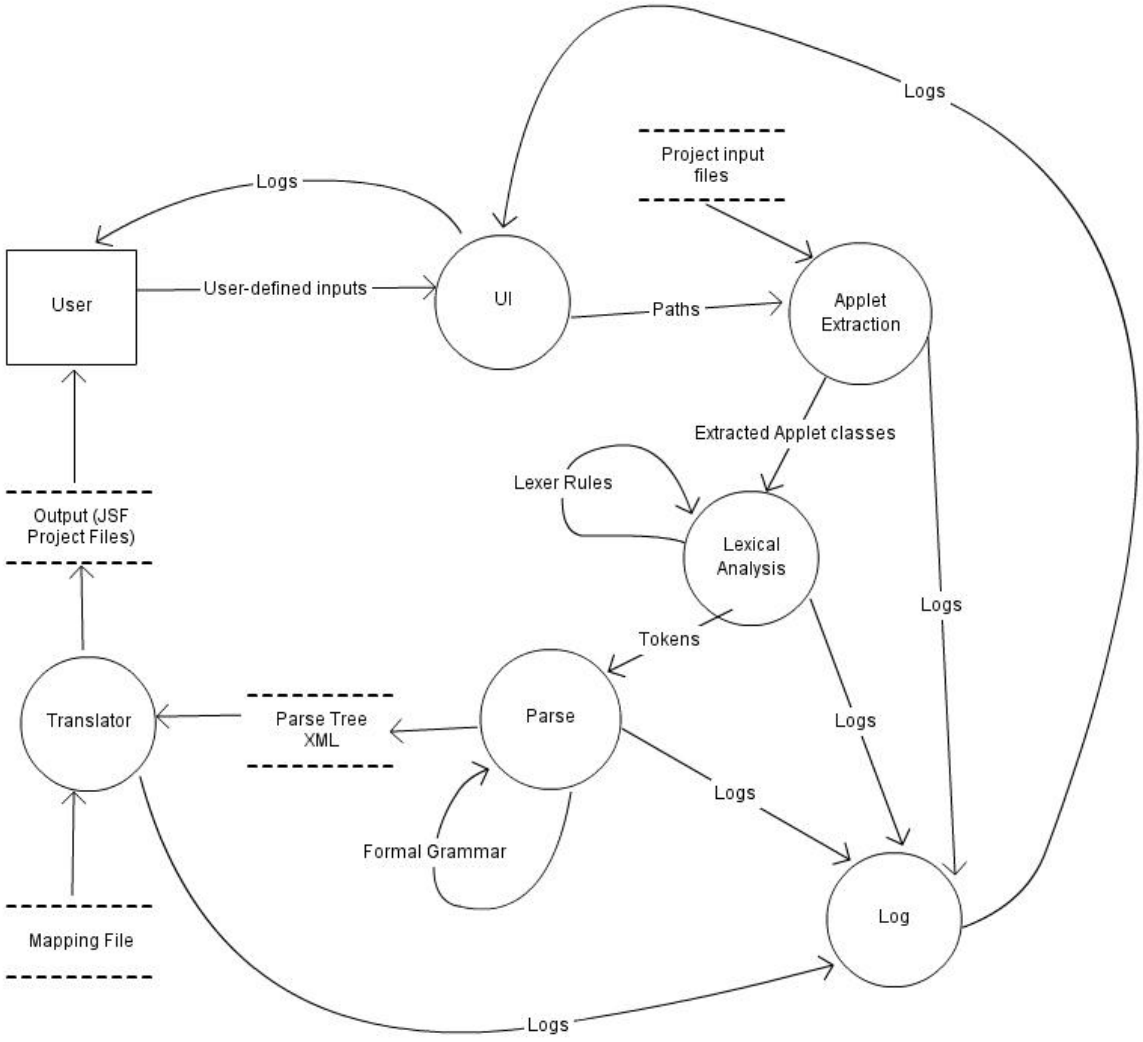
Data objects used in this project can be divided two as internal data objects that flow among internal system components, and external data objects processed between system and external environments.

- Internal Data Objects:
    - Extracted Applet Classes: Applet class names, used for conversion by other system components.
    - Lexer Rules: Created for tokenizing character sequences in Lexer component.
    - Tokens: Generated and used by Lexer component.
    - Formal Grammar: Generated and used by Parser component.
    - Logs: Sent from other components to Logger component and also from Logger component to UI component.
- External Data Objects:
    - User-defined Inputs: User must define source project folder path and destination project folder path. These data are used to get all project files included in source project folder path and generate output JSF project files in destination project folder path.
    - Project Input Files: These files are extracted from source project folder path after user defined input data. Then, Applet Class Extractor component uses these files to start conversion.
    - Parse Tree: Generated as XML files by Parser component. Parse trees are used by Translator component of system.

- Mapping File: Created for purpose of storing mappings between Applet components and JSF components as XML format.
- Output (JSF Project files): These files include configuration files, generated and unmodified Java files.
- Logs: Sent from UI component to user.

4.1.2 Relationships

Relationships among data objects which are described above and the system are shown in the data flow diagram below.



Data Flow Diagram

4.1.3 Complete data model

AJCON project does not use database. Therefore, ER Diagram for database modeling is not drawn. For data modeling of the system, data flow diagram is supplied in section 4.1.2.

4.1.4 Data dictionary

Since the project does not contain any database object, there will be no data dictionary refers to it.

5. Behavioral Model and Description

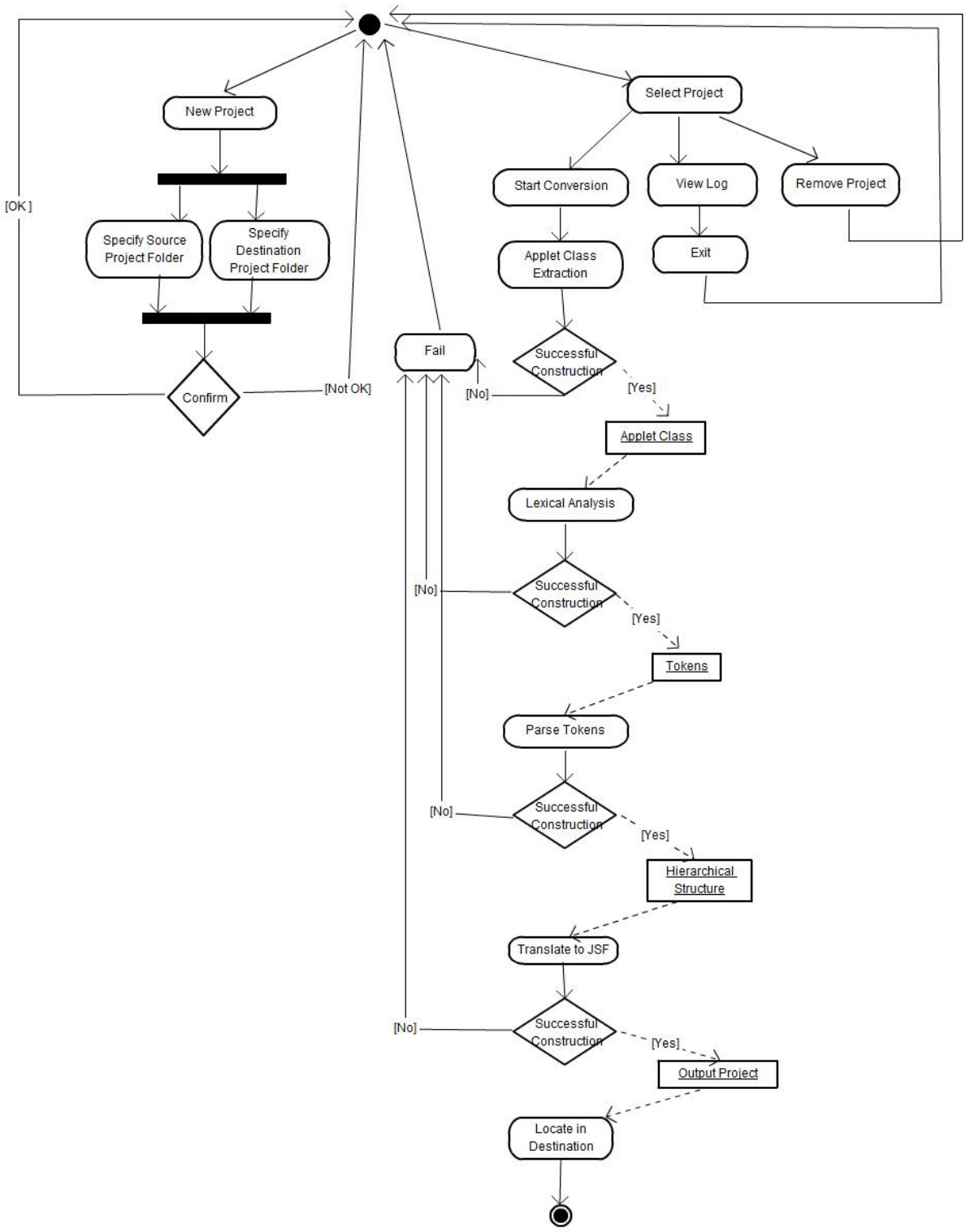
Behavior of the system will be explained with mostly diagrams.

5.1. Description for software behavior

An activity diagram showing the major events and states of system is stated below. In this diagram:

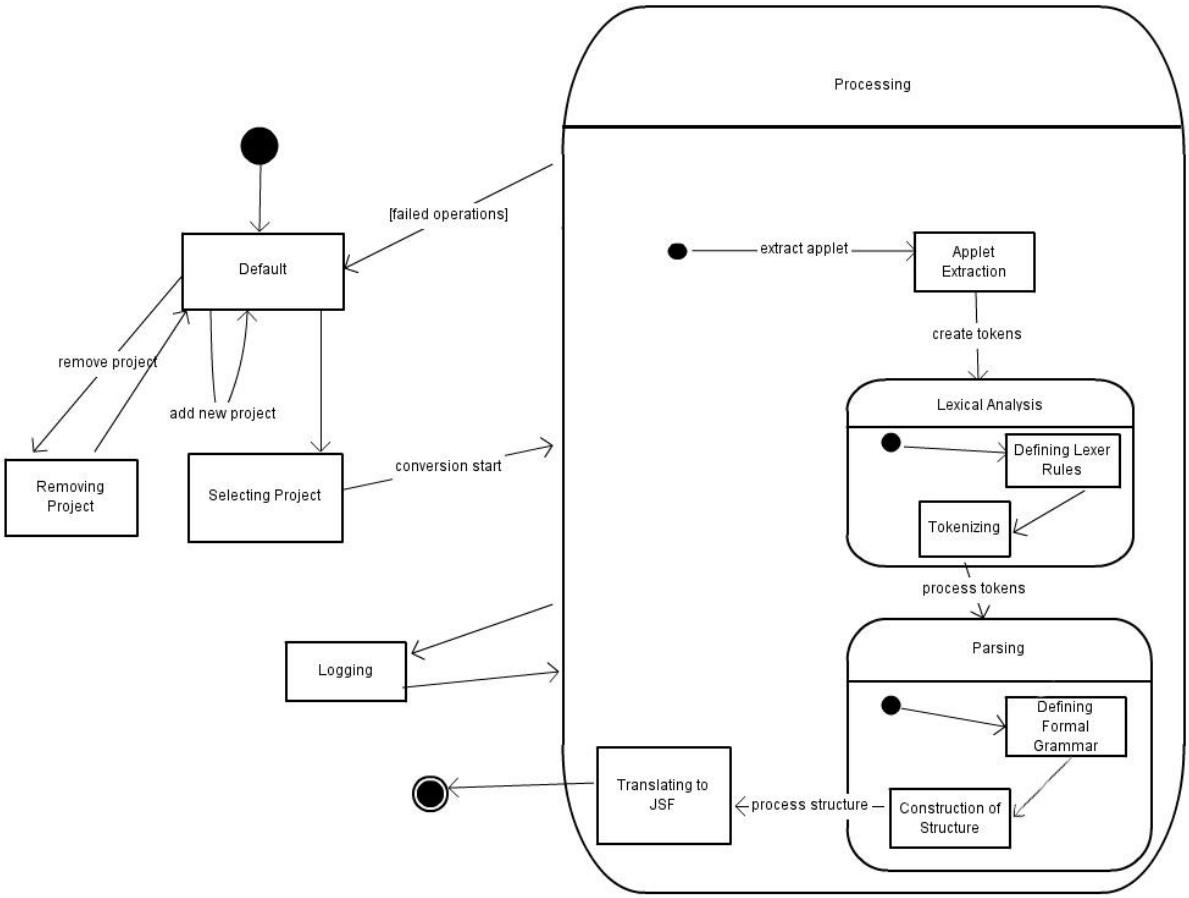
- Initial State is the state that nothing is done yet. System is at “Ready State”.
- Final State is the state that an input project is successfully converted to JSF.

Some of the activities generates objects in intermediate levels. Processing of those objects sequentially without any error will result in “Final State”.



Activity Diagram

5.2. State Transition Diagrams



State Diagram

Above is the state transition diagram of AJCON. General behavior of the system is shown in the activity diagram in section 5.1 and state transition diagram together.

6. Planning

6.1. Team Structure

In every professional software engineering project, there should be a team structure which represents a hierarchical structure among employees. Our team is relatively small compared to big industry-based projects and includes a project sponsor, an advisor, a team leader and 3 software developers.

Role	Description
Project Sponsor	Our project sponsor is Siemens EC. Siemens EC have the authority to make any changes on AJCON project in return of meeting any needs demanded by team members.
Advisor	Our project advisor is Assoc.Prof.Dr. Ferda Nur Alpaslan from Middle East Technical University, Computer Engineering Department. She guides the team on any difficulties.
Team Leader	Alper Teke from Siemens EC is our team leader. From time to time, he arranges meetings to check whether project process is continuing properly and gives new directions to follow if needed.
Front-End Software Developer	Berkan Kisaoglu is a front-end software developer in AJCON project. He is responsible from interface of the project, which connects the user to back-end software developers.
Back-End Software Developer-1	Ozge Tokgoz is a back-end software developer in AJCON project. She is mostly responsible from lexical analysis of any Java Applet project.
Back-End Software Developer-2	Anil Sevim is a back-end software developer in AJCON project. He is responsible from parsing tokens and generating JSF project.

6.2. Estimation (Basic Schedule)

We used COCOMO Model for estimation. First calculated the Function Point (FP), and then calculated other related information.

User Inputs:

- Source project folder
- Destination project folder

User Outputs:

- Logs
- Result of conversion (JSF Project)

User Inquiries:

- Start Conversion operation
- Add project
- Remove project
- Select project
- Select folder

Files:

- Result of generated structure in XML format(Parse Tree)
- Dictionary of equivalences in XML format
- Template web.xml file
- Template faces-config.xml file

External Interfaces:

- Main Interface
- Project Interface
- Log Interface

	Simple	Average	Complex		Simple Weight	Average Weight	Complex Weight	Total
#User Inputs	2	0	0		3	4	6	**6**
#User Outputs	0	0	2		4	5	7	**14**
#User Inquiries	0	5	0		3	4	6	**20**
#Files	0	0	4		7	10	15	**60**
#External Interfaces	0	3	0		5	7	10	**21**
							Unadjusted function point(UFC)	**121**

Complexity Adjustment Values Fi:

- Reliable backup and recovery 3
- Data communications 5
- Distributed processing 0
- Critical performance 5
- Heavily utilized operational environment 2
- On-line data entry 0
- Input transactions over multiple screens (on-line) 0
- Master file updates on-line 1
- Complex input/output/file/inquiries 5
- Complex internal processing 5
- Reusable code design 5
- Conversion and installation included in design 5
- Multiple installations for different organizations 2
- Design for facilitating change and ease of use 5

$\Sigma F_i =$ 43

FP = UFC x [0.65 + 0.01 x ΣFi]

= 121 x [0.65 + 0.01 x 43]

= 130.68

Project size estimation

Basically object oriented languages will be used, therefore

LOC = 130.68 fp x 30 loc/fp

= 3920 loc

KLOC = 4.0 kloc

Project effort estimation

AJCON project is an organic project. So basic effort is:

Effort = a^b KLOC(b^b) where a^b = 2.4 and b^b = 1.05 for an organic project.

Effort = 2.4 x $4^{1.05}$

= 10.3 Staff-Months

Project schedule estimation

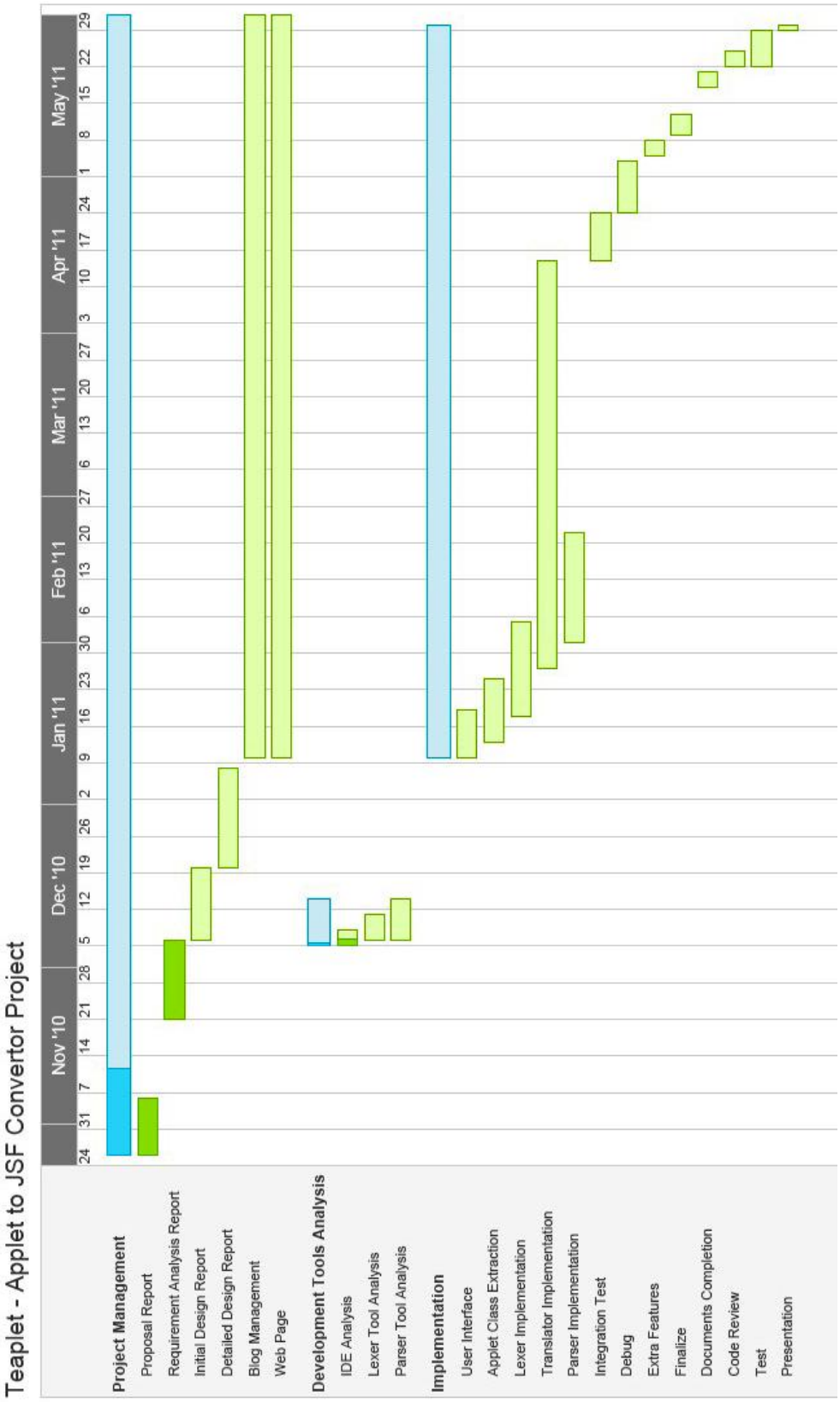
AJCON project is an organic project. So time is:

Time = c^b Effort (d^b) where c^b = 2.5 and d^b = 0.38 for an organic project.

Time = 2.5 x $10.3^{0.38}$

= 6.06 Months

~ 6.1 Months


Teaplet - Applet to JSF Convertor Project
Nov '10
Dec '10
Jan '11
Feb '11
Mar '11
Apr '11
May '11
24 31 7 14 21 28 5 12 19 26 2 9 16 23 30 6 13 20 27 6 13 20 27 3 10 17 24 1 8 15 22 29
Project Management
Proposal Report
Requirement Analysis Report
Initial Design Report
Detailed Design Report
Blog Management
Web Page
Development Tools Analysis
IDE Analysis
Lexer Tool Analysis
Parser Tool Analysis
Implementation
User Interface
Applet Class Extraction
Lexer Implementation
Translator Implementation
Parser Implementation
Integration Test
Debug
Extra Features
Finalize
Documents Completion
Code Review
Test
Presentation

6.3. Process Model

Every qualified software engineer should know that process managament is one of the most significant parts of software engineering concept, which makes process model considerably important.

Since 491 course is a step-by-step course, it seems that our process model should follow a sequential design process, which gives rise to waterfall model. However, being proposed and confirmed by Siemens EC veterans, AJCON is a relatively open-ended project. Since all requirements are not well-bounded and may shape according to Siemens EC and its customers, there will be different versions of both documentation and software. Because we need to update requirements and solutions through collaboration, our project should continue iteratively and incrementally, which ends up with agile software development model. By using this model, firstly, we will return back and update every single change on document. Then, modifications on software will be considered.

7. Conclusion

In this SRS document, complete description of AJCON system behaviour, features, data flowing and other requirements were stated. These reqirements will help the progression of the project in other stages. However, all of these may be subject to changes in further development stages.