



Acceptance Test Case Driven Component Selection Approach

BY

MUAADH ABDULGHANI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In
COMPUTER SCIENCE

MAY, 2014

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **MUAADH ABDULGHANI QAID** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Sajjad Mahmood

Dr. Sajjad Mahmood
(Advisor)

Adel Ahmed

Dr. Adel Ahmed
Department Chairman

Moataz Ahmed

Dr. Moataz Ahmed
(Member)

Salam A. Zummo

Dr. Salam A. Zummo
Dean of Graduate Studies

Mahmood Niazi

Dr. Mahmood Niazi
(Member)

1/6/14
Date



©MUAADH ABDULGHANI QAID

2014

Dedication

To my parents, wife, brothers, sisters.

ACKNOWLEDGMENTS

Thanks and all praise is to Allah (azzawagal) who created me, blessed me with health and who grant me success and give me the strength and motivation to pursue this degree.

I would like to state my deep appreciation and gratitude to my thesis advisor Dr. Sajjad Mahmood who has always been providing kindest support, continual encouragement and supervision.

I also would like to acknowledge my committee members Dr. Moataz Ahmed and Dr. Mahmood Niazi for their valuable advices and guidance.

I also acknowledge KFUPM and Information and Computer Science (ICS) department as well as all faculties who taught me.

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	V
TABLE OF CONTENTS.....	VI
LIST OF TABLES.....	VIII
LIST OF FIGURES.....	X
LIST OF ABBREVIATIONS.....	XI
ABSTRACT.....	XII
ملخص الرسالة.....	XIV
CHAPTER 1 INTRODUCTION.....	1
1.1 Introductory Background	1
1.2 COTS Based Development Overview	3
1.3 An Acceptance Testing Overview	5
1.4 Thesis Objective	6
1.5 Thesis Outline	7
CHAPTER 2 LITERATURE REVIEW	9
2.1 Acceptance Testing and Requirements Clarification.....	9
2.2 The General COTS Selection Process	10
2.3 COTS Evaluation Strategies	11
2.4 Component Selection Approaches	12
CHAPTER 3 THE PROPOSED METHODOLOGY	16

3.1	Requirements Modeling.....	17
3.2	Acceptance Test Cases Generation.....	19
3.3	COTS Searching and Filtering.....	23
3.4	Configure and Evaluate COTS.....	26
3.5	COTS Selection.....	27
CHAPTER 4 CASE STUDY.....		29
4.1	Overview.....	29
4.2	Requirements Modeling.....	31
4.3	Acceptance Test Cases Generation.....	32
4.4	COTS Searching and Filtering.....	37
4.5	Configure and Evaluate COTS.....	43
4.6	COTS Selection.....	46
CHAPTER 5 CONCLUSION.....		49
5.1	Summary and Contributions of the Thesis.....	49
5.2	Limitations and Future Work.....	50
REFERENCES.....		51
APPENDIX A: DETAILED EVALUATION'S TEST SUITES.....		56
APPENDIX B: FILTERING EVALUATION'S RESULTS.....		80
VITAE.....		83

LIST OF TABLES

Table 1 Keywords used in Action table	21
Table 2 Action Fit table for testing buying books	22
Table 3 Row Fit table for testing searching books	22
Table 4 Column Fit table for testing a discount on books	23
Table 5 MSS goals' model.....	31
Table 6 Descriptions of CLG1TS's fixtures	33
Table 7 CLG1TS's fixtures.....	34
Table 8 CLGs and their potential candidate components	38
Table 9 Detailed information about candidate components	40
Table 10 Shortlist of the most suitable components for the MSS case study	42
Table 11 CLG1TS evaluation's results.....	44
Table 12 CLG2TS evaluation's results.....	44
Table 13 CLG3TS evaluation's results.....	44
Table 14 CLG4TS evaluation's results.....	44
Table 15 CLG5TS evaluation's results.....	45
Table 16 CLG6TS evaluation's results.....	45
Table 17 CLG7TS evaluation's results.....	45
Table 18 CLG8TS evaluation's results.....	45
Table 19 Fitness of candidate components	47
Table 20 CLGs and their best-fit components	48
Table 21 Description of CLG1TS's fixtures.....	56
Table 22 CLG1TS's fixtures.....	57
Table 23 Description of CLG2TS's fixtures.....	60
Table 24 CLG2TS's fixtures.....	61
Table 25 Description of CLG3TS's fixtures.....	62
Table 26 CLG3TS's fixtures.....	63
Table 27 Description of CLG4TS's fixtures.....	66
Table 28 CLG4TS's fixtures.....	67
Table 29 Description of CLG5TS's fixtures.....	69
Table 30 CLG5TS's fixtures.....	70
Table 31 Description of CLG6TS' fixture.....	75
Table 32 CLG6TS' fixture.....	76
Table 33 Description of CLG7TS' fixture.....	77
Table 34 of CLG7TS' fixture	77
Table 35 Description of CLG8TS' fixture.....	78
Table 36 CLG8TS' fixture.....	78
Table 37 Filtering Evaluation's Results for Polling for meeting components.....	80

Table 38 Filtering Evaluation's Results for Notification components	80
Table 39 Filtering Evaluation's Results for File upload components	81
Table 40 Filtering Evaluation's Results for Image upload components	81
Table 41 Filtering Evaluation's Results for Rich WYSIWYG Editing components	82

LIST OF FIGURES

Figure 1 CBS development phases (adapted from [28]).....	5
Figure 2 Component selection activities (adapted from [37])	11
Figure 3 Overview of the proposed methodology	17
Figure 4 Goals model.....	19
Figure 5 Snapshot from Node Notify website	24
Figure 6 Generate and evaluate configurations	27
Figure 7 Activities and outputs in AccepCotSel.....	30
Figure 8 Drupal's search engine	39
Figure 9 Snapshot from Drupalmodules' website	39

LIST OF ABBREVIATIONS

Fit: Framework for Integrated Test

BAs: Business Analysts

SUT: System Under Test

CBS: Component Base System

COTS: Commercial Off-The-Shelf

ABSTRACT

Full Name : Muaadh Abdulghani Qaid
Thesis Title : Acceptance Test Case Driven Component Selection Approach
Major Field : Information and Computer Science
Date of Degree : May, 2013

Component Based System (CBS) development is a systematic reuse approach that helps in developing software by integrating existing components. Component selection plays a vital role in the success of the CBS. Lately, researchers have suggested using acceptance test cases as a functional specification to better understand software requirements. In this thesis, we present an acceptance test case driven component selection process that provides guidelines for CBS developers to select candidate components that best match the required functionalities of a CBS. We use acceptance test cases to understand functional requirements of a CBS-to-be. The acceptance test case driven component selection process consists of five phases: (i) requirements modeling (ii) acceptance test cases generation (iii) searching and filtering the candidate components (iv) configuration and evaluation of the filtered components (v) selection the most suitable component. We use the goal modeling technique to specify requirements of a CBS in phase (i). In phase (ii), we use Framework for integration test (FIT), an open sources framework for generating acceptance test cases, to create acceptance test cases for CBS-to-be. In Phase (iii), we use keyword search to identify suitable candidate components for all concrete goals from existing COTS repositories and filtering them based on a proposed equation. In phase (iv), we develop configurations for the short listed components and evaluate them against acceptance test cases. Finally, in phase (v), the configured components are

ranked based on acceptance test case satisfaction scores and the highest ranked components will be selected.

ملخص الرسالة

الاسم الكامل: معاذ عبدالغني قائد نعمان

عنوان الرسالة: منهج يعتمد على استخدام حالات اختبار القبول في اختيار مكونات ملائمة لبناء نظام

التخصص: علوم الحاسب الآلي

تاريخ الدرجة العلمية: مايو 2014

تعتبر طريقة بناء الأنظمة القائمة على المكونات (سي بي إس) من الطرق التي تتبنى عملية إعادة الإستخدام للبرمجيات حيث يتم بناء أنظمة مكونة من دمج مكونات موجودة مسبقاً. وتلعب عملية اختيار المكونات دور أساسي في عملية بناء الأنظمة القائمة على المكونات. أقترح الباحثون استخدام اختبارات قبول النظام لتوضيح الوظائف المطلوبة من النظام من قبل العملاء. وقد كشف العديد منهم أن استخدام حالات اختبار القبول كمواصفات وظيفية تيسر فهم متطلبات النظام للمطور. ونحن نعتقد أن المعلومات في حالات اختبار القبول يمكن أن تكمل وثائق الأنظمة القائمة على المكونات ويمكن أن تكون مفيدة في سياق عملية اختيار المكونات لبناء هذه الأنظمة. في هذه الأطروحة، نقدم طريقة لإختيار مكونات تعتمد على حالات اختبار القبول وهذه الطريقة تعطي مبادئ توجيهية للمطورين لإختيار أفضل المكونات الموجودة والتي تتوافق مع الوظائف المطلوبة من النظام المزمع بناءه من هذه المكونات المختارة. تتكون طريقة اختيار المكونات المقدمة في هذا البحث والمعتمدة على حالات اختبار القبول من خمس مراحل (أ) نمذجة المتطلبات (ب) إنشاء حالات اختبار القبول (ج) البحث عن المكونات وترشيحها (د) تثبيت المكونات المرشحة وتقييمها (هـ) اختيار المكونات الأكثر ملائمة. نستخدم تقنيات النمذجة التي تمثل متطلبات النظام على شكل أهداف في المرحلة (أ). وفي المرحلة (ب)، نستخدم إطار اختبار التكامل (Fit)، وهو إطار مفتوح المصدر، لتوليد حالات اختبار القبول. وفي المرحلة (ج)، نستخدم كلمات البحث من كل هدف تقني للبحث عن المكونات في مستودعات المكونات ونرشحها بناء على معادلة أقترحناها. في المرحلة (د)، نثبت المكونات ومتطلباتها في بيئة التطوير ونقيمها بناء على تطبيق حالات اختبار القبول عليها. وأخيراً، في المرحلة (هـ)، ترتب المكونات التي تم تقييمها بحسب نتائج تطبيق حالات القبول ثم نختار المكونات التي لها قيمة أعلى.

CHAPTER 1

INTRODUCTION

1.1 Introductory Background

Software development community has been of the view that software does not have to be developed from scratch all the time. Systematic software reuse is recognized as a key technique to improve software development productivity and achieve high quality software [1][2][3]. One of the key motivations for software reuse is to reduce development effort and cost by reusing existing code and gains in quality by incorporating reliable components. In a nutshell, reuse-based software development focuses on strategies and techniques to enable developers to create new systems using existing software artifacts and software components [2]. A number of techniques have been proposed to help realize potential benefits associated with systematic reuse of software artifacts. For example, software developers have adopted reuse based techniques such as product lines [4][5][6], component-based systems [7][8], commercial off-the-shelf (COTS) based development [9]–[11], design patterns [12][13] and matching and modeling tools [14]. Furthermore, analysis and design software artifacts such as requirements specifications and acceptance test cases have also been used during reuse-based development of software.

One of the promising approaches that aim to not develop systems from scratch is Component Based System (CBS) development. CBS development helps in building software by integrating existed software components. A component is considered as main building block for a CBS.

In this thesis, we adopt the component definition as follows [15]: "A software product: developed by a third party (who controls its ongoing support and evolution); bought, licensed, or acquired for the purposes of integration into a larger system as an integral part, i.e. that will be delivered as part of the system to the customer of that system; which might or might not allow modification at the source code level, but may include mechanisms for customization; and is bought and used by a significant number of systems developers".

Analyzing the requirements and selecting the components are key phases in CBS development [16]. Appropriate selection of component is the key to the success of the CBS development [17][18][19]. An improper component selection will lead to undesirable effects, for instance introducing an additional cost, in maintenance and integration phases [18]. Considerable effort has been done to determine the challenges related to CBS requirements analysis and component selection. For example, Lee et al. [20] showed a component identification approach with a focus on low coupling and high cohesion values. Similarly, Jain et al. [21] presented a component selection approach based on clustering algorithm. The clustering algorithm depends on a set of predefined rules and heuristics for the clustering.

Traditionally, software systems have failed because late discovering the mismatching between the customers' expectations and the developed system functionalities [22][23]. Specifying the systems functionalities through acceptance test cases has been used for requirements clarification [23]–[26]. Melnik et al. [25] have revealed that the using acceptance test cases as a functional specification facilitate developer understanding for requirements. Lately, Ricca [23] did many experiments which show that Framework for Integration Test (Fit) tables [27] reduce the requirements understandability efforts. We believe that acceptance test case information can complement CBS documentation and can be helpful in the context of components selection process.

This thesis presents an approach for components selection for CBS based on acceptance test cases for selecting components that best-fit the required functionality by the CBS. The goal of our approach is to illustrate the benefits of using acceptance testing throughout the component selection phase in the CBS development. The component selection approach which is developed in this thesis consists of five phases: (i) requirements modeling (ii) acceptance test cases generation (iii) Search candidate COTS and filtering (iv) generate and evaluate configurations (v) select configuration.

1.2 COTS Based Development Overview

CBS development is a methodology for building software applications by using existed components. Qureshi and Hussain [28] presented a CBS development process by adapting the object oriented methodology [29]. In addition to the object oriented methodology phases [30], the CBS development have three phases which are (1)

selecting components; (2) engineering and testing; (3) evaluation [28]. Figure 1 showing the CBS development life cycle overview [28]. The phases formulate the CBS development process described as follows:

Phase1: Communication, in this phase the objectives and requirements of the required CBS are gathered.

Phase2: Planning, in this phase the specification document of the required CBS is created. Furthermore, the risk and feasibility of the required CBS are assessed.

Phase3: Analyzing and component selecting, this phase contain two stages, namely, analysis and selection. The analysis stage concentrates on collecting the detailed requirements of the required CBS and analyzing the relationships between those requirements. The “component selection” stage in which the suitable components will be selected.

Phase4: Engineering and testing, in this phase the focus is on accommodating the chosen components and glue-code writing to put them together to build the required CBS.

Phase5: Evaluation, in this phase the customers verify that the CBS is as they requested or no.

In this thesis, we only focus on **phase3**, namely, Analyzing and component selecting.

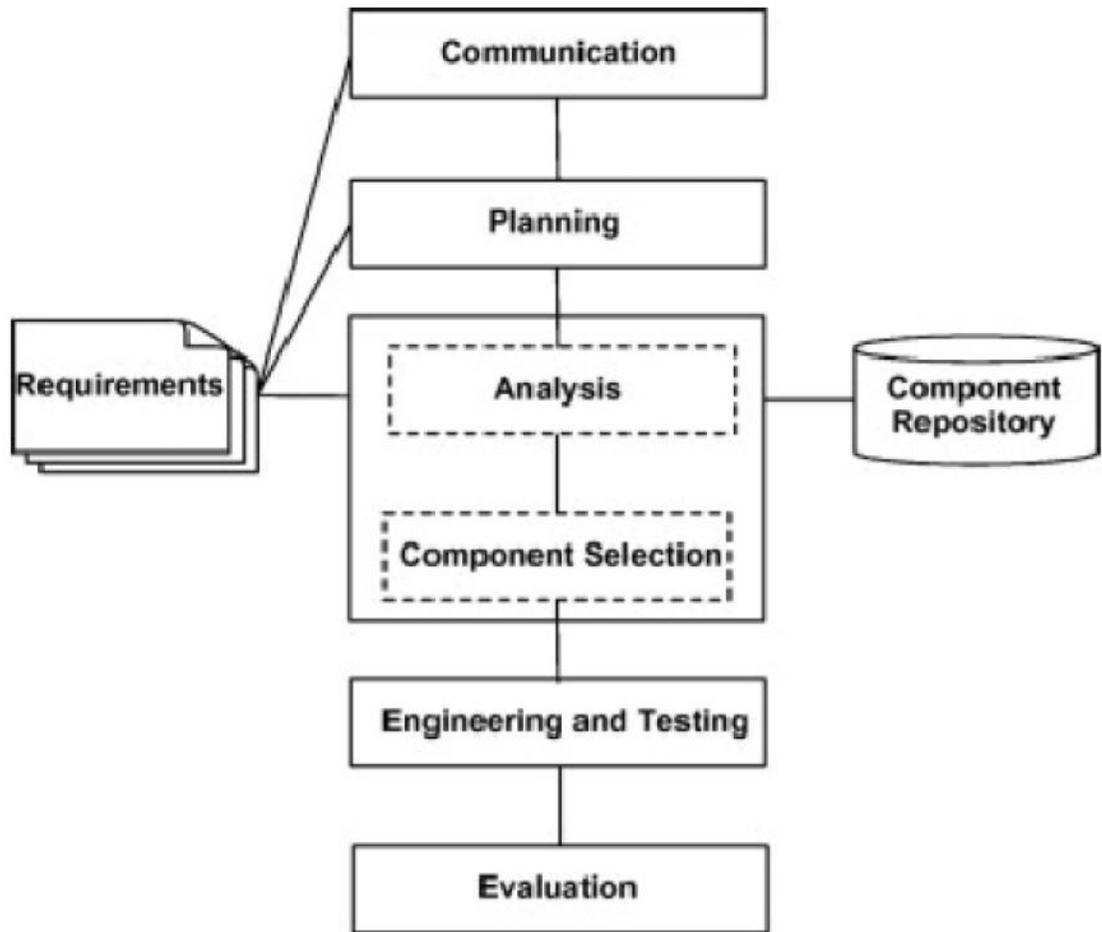


Figure 1 CBS development phases (adapted from [28])

1.3 An Acceptance Testing Overview

Acceptance testing is recognized as a validation activity that conducted by the customer to enable him to determine to accept the system or no [31]. It is done on the parts of the system or the entire system before delivering it to the customer. The acceptance test consists of many test cases, each case include inputs and outputs. In order to automate the testing, test cases are performed in a formal framework called Fit.

Framework for Integration Test (Fit) [27] is a framework for automation acceptance testing. It facilitates the communication between the business analysts and developer's communication, through using concrete examples about the needed functionality of software application. The tests' inputs and expected outputs are specified using tables called Fit tables. There are three types of Fit tables, specifically, the row tables, column tables, and action tables. Row tables are used for testing either the search or query results as expected or no [27]. Column tables are designed for testing the correctness the calculations of the software system. Action tables are used for testing the system sequence of actions. Additionally, Fit tables are just appropriate for modeling functional requirements in which there exists relations among inputs and outputs [23].

FitNess is an open source tool for creating, organizing and running the Fit tables. In FitNess, the acceptance tests are defined and executed in a collaborative environment by using web pages. FitNess uses yellow, green and red color codes [27] in test results. The yellow color indicates that the software generated an exception. The green color indicates that the value returned by the software is as expected. In the same way, the red color specifies that the value returned by the software is an incorrect value.

1.4 Thesis Objective

The aim of this thesis is to use acceptance testing information as functional specification to facilitate systematic selection of software components for a CBS. Our motivation is to use acceptance tests in conjunction with stakeholder's requirements to select suitable components for a CBS. The contributions of this thesis are as follows:

1. Develop an acceptance tests driven component selection technique for selecting components that match customer requirements.
2. Evaluate the effectiveness of the approach on a case study.
3. The case study result will be investigated and the conclusion and future work in this field will be introduced.

1.5 Thesis Outline

This thesis is organized as follows:

Chapter 2: Literature Review

In this chapter, we investigate the role of acceptance test cases in the requirements clarification. Furthermore we explored the current state of the art in the COTS selection. We presented the current COTS selection approaches and gave a necessary background for better understanding the COTS selection practices.

Chapter 3: Acceptance Tests Driven Component Selection Approach

In this chapter, we present an Acceptance tests driven Component Selection approach (AccepCotSel). The AccepCotSel approach consists of five phases, namely, requirements modeling, acceptance test cases creation, and search candidate COTS and filtering, generate and evaluate configurations, and select configuration.

Chapter 4: Validation of the Approach

In this chapter, we evaluated the usefulness of the presented approach by applying it to a case study. The aim is to show that the proposed approach is suitable to facilitate selection of the software components that match stakeholder requirements.

Chapter 5: Conclusions and Future Work

We summarize our work and discuss the threats of validity. Furthermore, future works is also presented.

CHAPTER 2

LITERATURE REVIEW

The Focus of the CBS development is developing systems from previously existed COTS products, instead of building systems from scratch like in traditional systems. In the CBS development, the appropriate COTS products have to be selected earlier [32]. The selection is important for successful CBS [17][33]. The literature contains several approaches that are proposed to select suitable COTS components for the CBS.

In this chapter, we investigate the role of acceptance test cases in the requirements clarification. Furthermore we explored the current state of the art in the COTS selection. Specifically, we presented the current COTS selection approaches and gave a necessary background for better understanding the COTS selection practices.

2.1 Acceptance Testing and Requirements Clarification

The feasibility of the test driven approach in development has been proved by a lot of empirical studies. One of the main advantages of test driven approach for development is helping in producing high quality source codes [34]. Furthermore, It has shown that it helps in software quality improvement [35]. For more information about the test driven approach in development and its usefulness refer to [23], [36].

The main motivation for our work understands more the effectiveness of the testing based development methods in Component Base Software (CBS). Our work goes after the previous studies which are done by [23], [25]. Melnik et al. [25] did empirical

experiments and shown that Fit tables is playing a great role in the functional requirements specifying. It is also improving understanding of the requirements. Furthermore, Ricca et al. [23] has shown by doing empirical experiments that acceptance test cases in the form of Fit tables has a significant contribution in clarifying the requirements.

We follow the same idea of [23], [25] to investigate the helpfulness of Fit tables, however, our concentration is investigating the potential usefulness of Fit tables in the process of selecting the suitable components in CBS development. As far as we know, none of the research work in CBS development explores the using of acceptance test cases in supporting the components selection.

2.2 The General COTS Selection Process

The COTS selection process is a methodology for selecting one or more suitable COTS products. Sami [37] presented general framework for COTS selection model that is followed by the most COTS selection approaches. Figure 2 gives an overview of the COTS selection life cycle.

The COTS selection process consists of five steps, specifically, defining criteria, searching, screening, evaluation, and component selection.

Step1: Defining criteria, focus in the definition of the evaluation criteria depending on the system constraints and requirements.

Step2: The candidate components will be searched for.

Step3: Screening is carried out to find the shortlist from the searched candidate components which will be used in the next step.

Step4: The shortlisted components will be evaluated.

Step5: The evaluation data will be analyzed and the fitting component will be selected.

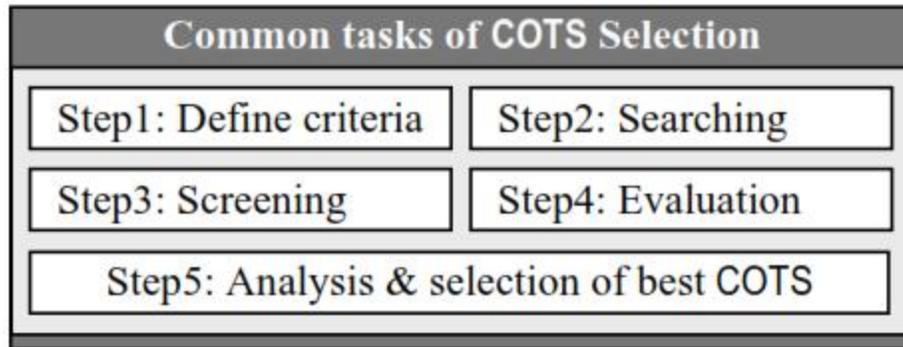


Figure 2 Component selection activities (adapted from [37])

2.3 COTS Evaluation Strategies

The main activity in the COTS selection is the COTS evaluation. In the COTS evaluation, the suitability of COTS components is determined. It provides the decision makers with the necessary information for choosing suitable COTS product (s) from the candidate products alternatives [38].

COTS are assessed by using criteria that correspond to the requirements and the constraints of the system. The literature includes many approaches for defining the evaluation criteria. For example, Kontio et al. [39] proposed to define a hierarchical evaluation criteria, in which the high level abstracts goals are refined to low level goals

with the help of using system requirements and architecture in addition to the products features.

Maiden et al. [17] presented an approach and suggested adding, repeatedly, discriminating criteria during the evaluation of the candidate COTS products

Many efforts are included in the literature that talks about using quality models as a basis to define the evaluation criteria. For instance, Franch et al. [40] presented an approach that uses the ISO/IEC 9126 quality model [41] for building COTS evaluation criteria.

There are three methods that are used for evaluating COTS products, namely, progressive filtering, puzzle assembly, and keystone identification [42]. It is possible to use multiple methods in the same project [42]. In the Progressive filtering, the less fit components are eliminated through defining discriminating criteria with a repeated products' evaluation. The most suitable COTS components are identified by running the first four steps of general selection process repeatedly. In the puzzle assembly many components are combined together as a puzzle pieces. The requirements of the puzzled products are considered at the same time. Keystone identification starts by selecting a specific requirement, and searching for the convenient components for the requirement. This technique makes the elimination of many products that not suit the crucial requirement, quick and easy.

2.4 Component Selection Approaches

Considerable effort has been done to determine the challenges related to the requirements analysis and the components selection in CBS development. Chung and Cooper [43]

proposed goal-oriented requirements engineering approach that concentrates on the requirement engineering of component selection process. They classified the requirements into two types, foreign requirements which are the features of the COTS products and native requirements that obtained from customers. They use the knowledge base to narrow the gap between the components and customer requirements. The approach highlights the significance of mapping product specification and system requirements; but, possible mismatching does not supported among both specifications.

Alves et al. [44] presents an approach to assess components according to their matching to the customer requirements. They used evaluation and resolution proposals for conflict managing and identification of the components. The candidate components evaluated using systematic criteria consists of quality attributes that have to be meet by the candidate components. The disadvantage of this approach is its complexity in case of having large alternatives of COTS.

Clark et al. [45] shows an iterative process for identification and selection components from the large repositories. In this approach, the integrator can search about components in the requirement analyzing stage. This method investigated the compatibility checks among components to ensure that the list of selected components for building the system is short.

Maiden and Ncube [17] proposed an approach for requirements acquisition and components selection. Their approach based on templates which are used progressively to evaluate the list of the candidate components to select suitable ones for a CBS. Repeatedly, the development team is acquiring the requirements using the templates, and

identifying and rejecting the components which are not matching with the specified requirements with the help of multi-criteria decision-making; after that, the selected components are explored in order to discover new requirements. Nevertheless, the process does not give enough details on requirements using in the components evaluation.

Cechich and Piattini [46] proposed an early procedure for measuring the suitability of COTS products. The procedure is using the functional features to reduce the candidate components. A number of metrics have been proposed for quantifying components. They presented a case study that illustrates how to calculate the functional measures.

Harman et al. [47] applied concepts from the field of search based software engineering for solving problems in the CBSs' components selection. They assign weight and cost values to each component. The weight contains customer desirability and anticipated revenue. In the same way, a component cost consists of the costs of acquisition and development. Finally, the search based software engineering is applied to the component selection and ranking problem after formulated it as a series of features that have weights and costs.

Fahmi and Choi [48] present a conceptual approach that incorporates the previous knowledge and decisions on components selection for reducing time and cost of the selecting components process. They use Case Based Reasoning (CBR) to achieve their goal. Nevertheless, the presented approach requires presenting the user requirements as a set of keyword functionalities.

Birkmeier and Overhage [49] presented a survey on the state of the art in components selection and identification and proposed a classification scheme for analyzing the strengths and weaknesses of different components selection and identification approaches. The survey concluded that to address the complexity of the components selection process, efforts from diverse areas is needed.

Lozano et al. [50] presented an approach that uses the Analytic Hierarchy Process (AHP) method [51] for components selection. The approach depends on the objectives of the project as the root of the decision tree with the intermediate level of the criteria and the different alternatives of it as leafs.

Khan and Mahmood [7] presented a graph based requirements clustering approach for components selection. The approach uses goal modeling to define the CBS' requirements, and a signed graph for representing the dependencies between the requirements. The related goals are clustered into groups based on the three proposed dependencies between them, namely, usage, non-functional, and threat dependencies. The components are selected based on the matching index for every cluster.

CHAPTER 3

THE PROPOSED METHODOLOGY

In this chapter, we present an acceptance test case driven component selection process that provides guidelines for CBS developers to select candidate components that best match the required functionalities of a CBS. We use acceptance test cases to understand functional requirements of a CBS-to-be. The acceptance test case driven component selection process, as shown in Figure 3, consists of five phases: (i) requirements modeling (ii) acceptance test cases generation (iii) COTS searching and filtering (iv) configure and evaluate COTS (v) COTS selection.

We use the goal modeling technique to specify requirements of a CBS in phase (i). In phase (ii), we use Framework for integration test (FIT), an open sources framework for generating acceptance test cases, to create acceptance test cases for CBS-to-be. In Phase (iii), we use keyword search to identify suitable candidate components for all concrete goals from existing COTS repositories. In phase (iv), we develop configurations for the short listed components and evaluate them against acceptance test cases. Finally, in phase (v), the configured COTS are ranked based on acceptance test case satisfaction scores and the highest ranked COTS will be selected. The five main phases are explained with more details in sections 3.1-3.5.

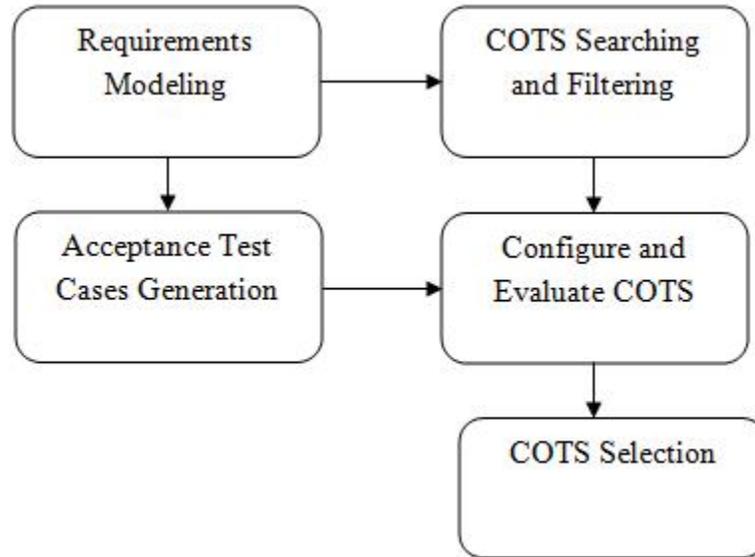


Figure 3 Overview of the proposed methodology

3.1 Requirements Modeling

Goal modeling [52][53] is a flexible approach for representing the requirements so that it is providing the analysts in CBS development the opportunity to shortlist the components that satisfy the required functionalities. The flexibility of requirements representation in CBS development is very important as it increases the probability of finding suitable matched components [16]. Many approaches use the goal-driven approach [53], [54] in modeling the CBS requirements as it helps in defining goals with different abstraction levels. For example, Sami [37] used the goal driven approach for defining the requirements for his component selection approach. He defined two types of goals, namely, strategic goals and technical goals. The strategic goals are high abstract goals which could not be satisfied directly by a COTS product. The technical goals are low

abstract goals which are resulted from decomposition of the strategic goals, and could be satisfied directly by a COTS product.

In this thesis , we adopt the goal definition as in [54]: “ an objective the composed system should meet. The goals can be stated at different levels of abstraction ranging from high-level strategic objectives of the organization to low level finer-grained goals stating technical objectives related to system design options”.

In this thesis, we adopt the hierarchical goal definition approach presented in [7]. In hierarchical goal definition approach, the CBS requirements are classified into two types: High-Level Goals (HLGs) and Concrete-Level Goals (CLGs). HLGs describe the requirements in CBS that are abstract and could not be used directly in components selection. It has to be refined to sub-goals that is so-called Concrete Level Goals (CLGs) that are directly employed as a basis of candidates components identification. There are two types of CLGs non-functional CLGs and functional CLGs. The functional CLG represents the required system behavior [54]. The non-functional CLG specifies choices of favorite functional behaviors. Each nonfunctional CLG is related to a particular functional CLG and has an effect on the picking of a candidate product for the related functional CLG.

According to the goal specification approach [54], the requirements of the CBS-to-be system start with selecting the business objective for it. The business objective of the system is decomposed into HLGs that collaborate to achieve it. Afterward, every HLG is decomposed into less abstract goals until the CLG are specified. The CLGs are the leaf nodes of the refinement hierarchy that is assigned to a single component that could be

part of the required system. The refinement of the CBS goal requirements depends on the required system [54]. The result of the refinement is a goal model that links HLGs and CLGs together (see Figure 4). The definition of the goal model should be carried out in parallel with components searching so that the analyst can define the concrete level goals in light of the customers' needs and components' features.

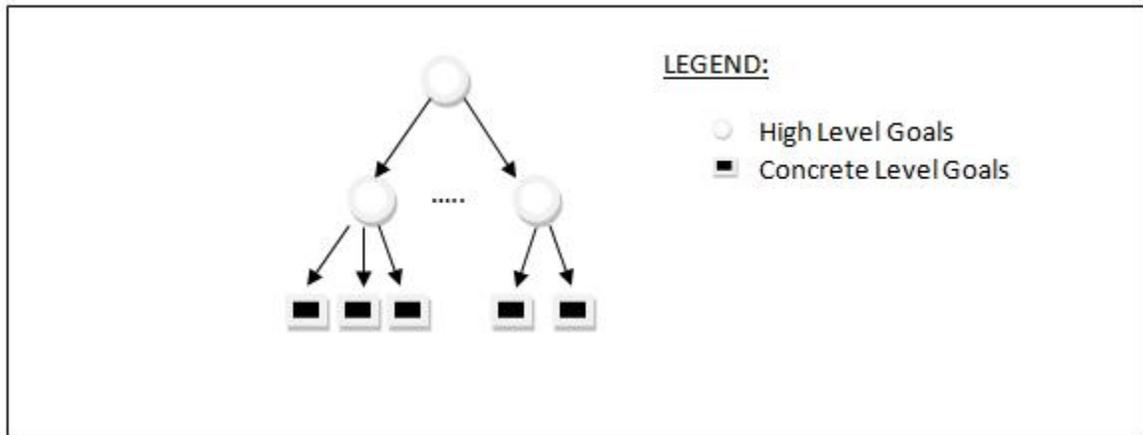


Figure 4 Goals model

3.2 Acceptance Test Cases Generation

The purpose of developing the test cases is to evaluate the configured candidate components. For each CLG, determined during requirements definition, a test suite is developed for evaluating its satisfaction by the candidate components in an accurate way.

We use FIT framework [27] because it is a famous framework for developing and executing the acceptance tests. Acceptance tests are developed in the FIT framework in tabular forms called Fit tables. Fit tables are used for specifying inputs and expected

outputs; it also provides the ability to check the tests automatically and reporting the final testing result.

There are three types of tables supported by FIT to enable test cases to be expressed as clearly as possible, namely, Action, Column, and Row tables. Different test cases are represented by different tables depending on its nature. Action tables are used to test the things happen on actions. It is particularly practical to represent test cases for the functionalities where frequent interactions with a user are expected. The rows in an Action tables are executed sequentially where each row (except the header) is considered an action. Each row consists of a numbers of fields. The first field of each row contains one of four commands that indicate the type of action to be performed. Table 1 provides a brief explanation of each command. The subsequent fields are used to include the necessary information to execute the command. Table 2 shows an example Action table for testing a functional requirement in which the user buy books and the total price is accumulated.

Table 1 Keywords used in Action table

Command	Usage
Start	Starts the given application. It is useful to execute this command to ensure the initial state of the system.
Enter	This command is used to create input. The following field contains the name of input to be entered, which is followed by another field containing the actual data value
Press	This command is used to execute functions in the given application. The following field contains the name of the function to be executed. Note that it is often the case that a function is executed by performing a GUI operation, such as a push of a button.
Check	The purpose of this command is to evaluate whether the output rendered by the application is the same as the expected output. The following field contains the name of the data to be evaluated, which is followed by another field containing the expected data value.

The Row tables are used to check whether the outputs of a query or search as expected or no. The header of a Row table indicates the name of the data structure to be evaluated. The remaining rows include the expected data elements. Table 3 shows an example Row table for testing a functional requirement in which the user search about specific books.

Table 2 Action Fit table for testing buying books

Fit.ActionFixture		
Start	BuyBooks	
check	total	0
enter	Book1's price	10
press	Buy	
check	total	10
enter	Book2's price	5
press	Buy	
check	Total	15

Table 3 Row Fit table for testing searching books

DisplayedBooks		
<i>Book</i>	<i>Auther(s)</i>	<i>Price</i>
Fit for Developing Software	Rick Mugridge , Ward Cunningham	\$54.01
ATDD by Example	Markus Gärtner	\$26.46

Column tables are used for formula evaluations where input is provided to a function and its output is evaluated. Column tables are useful to evaluate functionalities that do not include user interactions. The header of a Column contains the name of the function under test. The following row contains headers of data name columns and the tested function. Input columns start from the left and are followed by output columns on the

right. Table 4 shows an example Column table for testing a functional requirement in which a 10 percent discount is provided whenever the book price is greater than \$50.

Table 4 Column Fit table for testing a discount on books

CalculateDiscount		
Book	<i>Price</i>	<i>Discount()</i>
Fit for Developing Software	\$54.00	\$5.40
ATDD by Example	\$26.00	\$0.00
Agile Testing: A Practical Guide	\$100.00	\$10.00

3.3 COTS Searching and Filtering

The third step of our approach is COTS searching and filtering process. The main aim of this step is shortlisting the most suitable COTS candidates. Automatic search method is used. The automatic search means using strings to the search on the specialized components repositories' search engines and online search engines like Google and Bing [42], [49]. For a CLG we use keyword search to find the candidate components. Then, COTS filtering is done to filter out the less-fit COTS that resulted from searching.

The detailed information about the candidate components is collected through screening of candidate components documentations on the vendors' websites and the websites that offer evaluation for the components. An example component website is shown in Figure 5. The component's website contains various information about the component such as requirements, configuration, maintenance status, development status, usage statistics and versions.

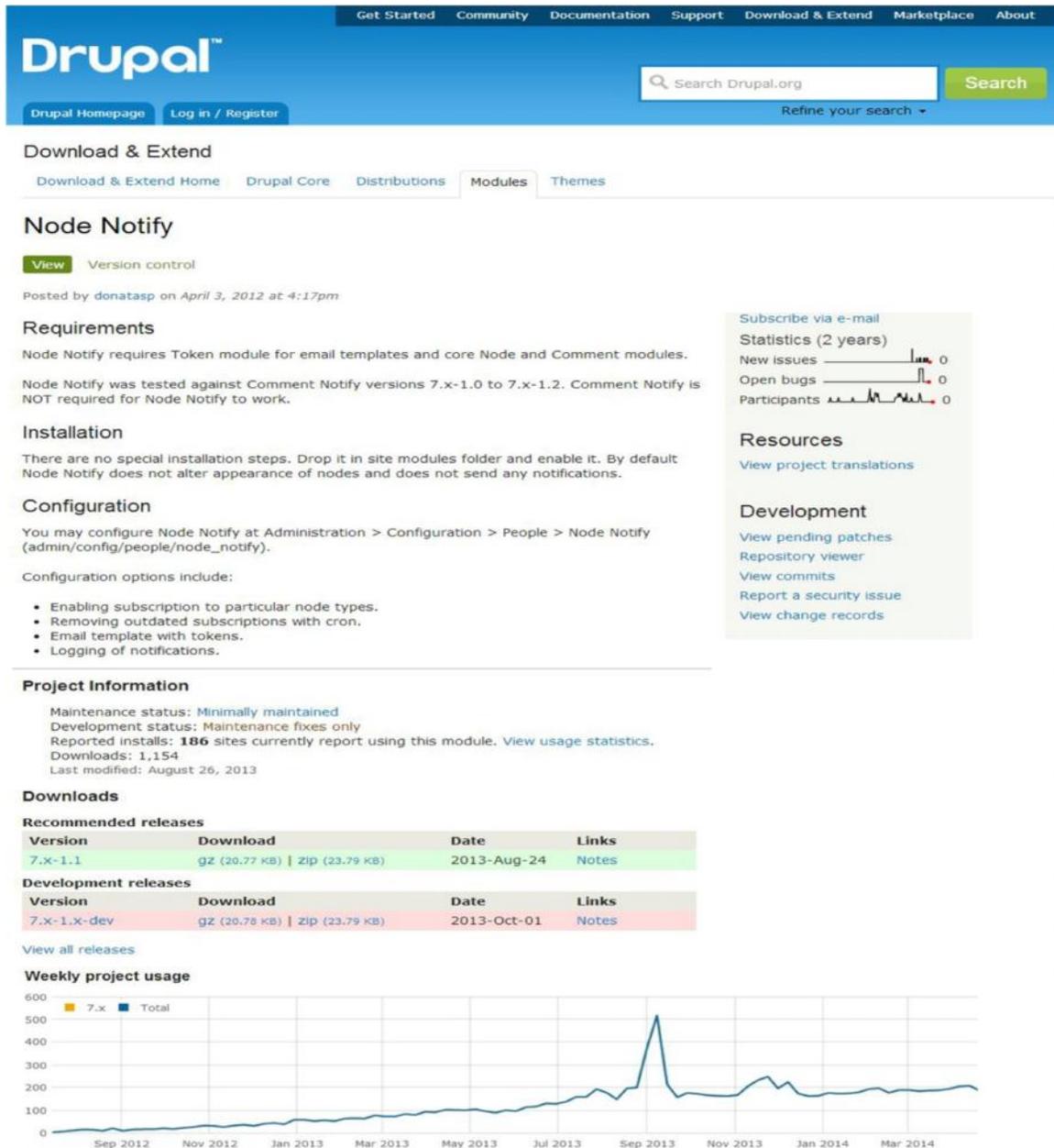


Figure 5 Snapshot from Node Notify website

We defined a function for filtering the candidate components called Filtering Scoring (FS). The FS value for a component $COTS_i$ is computed according to equation 3.1 below.

$$FS(COTS_i) = \frac{c_i * (1 + i_i)}{1 + r_i + b_i} \quad (3.1)$$

Where:

r_i is normalized value of the number of required components by $COTS_i$

b_i is normalized value of the number of the reported open bugs in $COTS_i$

i_i is normalized value of the number of the reported installations of $COTS_i$ by the customers

c_i is the compatability of the $COTS_i$ with the development environment, if the compatable with the development enviroment its vlaue will be 1 otherwise it takes 0 value.

Data is normalized to the range from 0 to 1 as in equation 3.2 [55]. We use variable X as an example in the calculations below. The normalized value of x_i for variable X in the i^{th} row is calculated as:

$$Normalized(x_i) = \frac{x_i - X_{min}}{X_{max} - X_{min}} \quad (3.2)$$

Where:

X_{min} is the minimum value for variable X

X_{max} is the maximum value for variable X

If X_{max} is equal to X_{min} then $Normalized(X_i)$ is set to 0.5.

The main goal of filtering is to decrease candidate components' number to be two candidate components for each CLG for the subsequent detailed evaluation for each CLG and accordingly minimize the effort and time of the complete evaluation process.

3.4 Configure and Evaluate COTS

In this phase, we generate configurations for each component resulted from the COTS searching and filtering phase, and subsequently apply the acceptance test cases, identified in phase 3.2, to evaluate the candidate components. Figure 6 give an overview of the configure and evaluate COTS activity. The following two steps are performed:

Step 1: Generate COTS' configuration. In this step, the candidate components are installed and adopted to the evaluation environment.

Step 2: Apply the test cases. The evaluation test cases are applied on the configured components, and the results are captured in the form of passed or failed test cases.

We defined a measure for the test cases implementation for each configuration called the Test Cases Satisfaction (TCS).

lets $TC_s = \{tc_1, tc_2, \dots, tc_m\}$ be a test cases set, and $CFG_s = \{cfg_1, cfg_2, \dots, cfg_n\}$ be a configurations set. The TCS value for a test case tc_r satisfaction by a configuration cfg_r has following values:

$$TCS(tc_r, cfg_r) = \begin{cases} = 1 & \text{iff } tc_r < Passed > cfg_r \\ = 0 & \text{iff } tc_r < Failed > cfg_r \end{cases} \quad (3.3)$$

The tc_r passed if the cfg_r satisfied it, and it failed if the cfg_r not satisfied it.

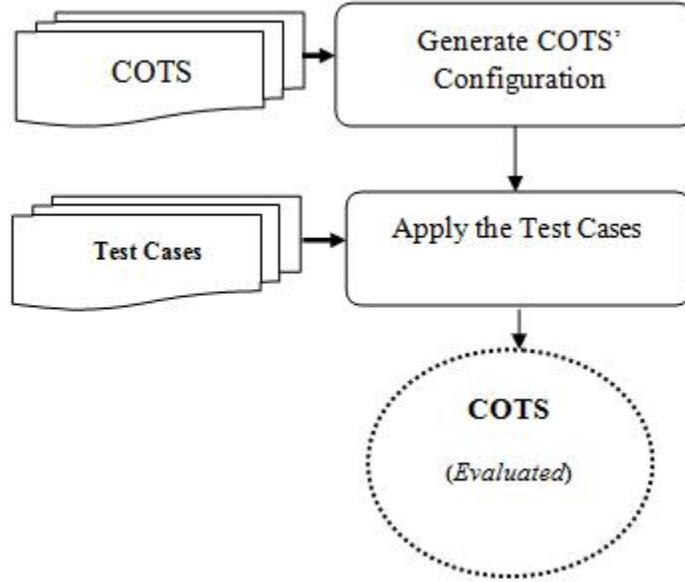


Figure 6 Generate and evaluate configurations

3.5 COTS Selection

Component selection is the final activity of our approach. In this phase, we defined a measure for the configured components scoring called the Matching Score (MS). The MS value for a configuration cfg_r is computed according to equation 3.4 below. Given the values of the TCS for the configuration cfg_r , the MS is computed as following:

$$MS(cfg_r) = \frac{\text{Number of passed test cases}}{\sum_{i=1}^n TCS(tc_i, cfg_r)} \quad (3.4)$$

The results of the COTS selection phase are COTS with their related fitness ranking from which the COTS with highest rank should be selected for each CLG. If the components having the same ranking, then the one with lower value resulted from dividing its reported open bugs by its installations is selected.

CHAPTER 4

CASE STUDY

4.1 Overview

In this chapter, we present a case study to demonstrate how our acceptance test case driven component selection approach provides guidelines for CBS developers to select candidate components that best match the required functionalities of a CBS.

In this case study, we wanted to select components that will be used to build a Meeting Scheduling System (MSS). MSS aim is to facilitate the process of scheduling meetings between different people with different preferences. The workflow for the scheduling process starts with the initiator person creating meeting proposal and specifying the meeting agenda, date range, durations, and the potential participants. The system sends notifications to the specified potential participants. The participants then view the meeting proposal and select their suitable time. After the participants submit their availability times the initiator study the input from the participants and make a decision regarding to the meeting time and date, the system notifies the participants about the specific date and time of the meeting.

In our case study, we have elicited four high-level goals and the corresponding eight concrete-level goals from the MSS literature and used ‘Drupal modules’ as a repository for identifying candidate components. We have selected Drupal Module as it is a popular

open source framework for web application development and the repository provides a large number of components, systematically classified in 47 categories.

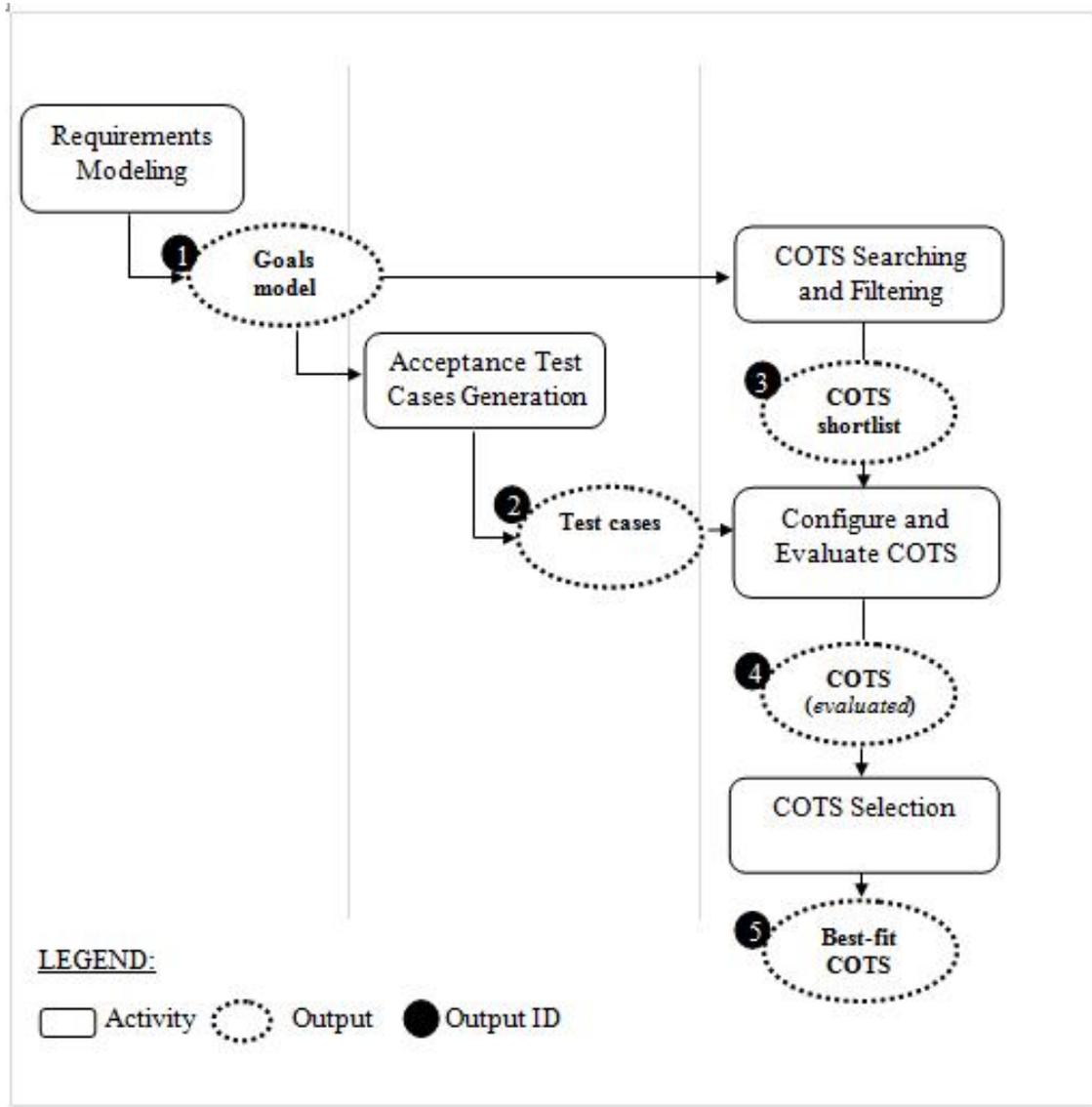


Figure 7 Activities and outputs in AccepCotSel

Figure 7 presents an expanded edition of the AccepCotSel approach as we applied during MSS case study. The figure contains similar activities to Figure 3, and expanded by adding the activities' outputs. Figure 7 is used as a reference during this chapter.

4.2 Requirements Modeling

The objective of this phase to define the goals model (① Figure 7) that will be used as a basis for suitable COTS selection process. Table 5 showing the MSS goals' model. The goals model consists of 13 goals: 5 HLGs defined at 3 hierarchical levels, and 8 CLGs at the fourth level. To clarify the goals model structure, as an example, the business objective at level 1 “Effective meeting scheduling” is refined into different goals at level 2, for instance, “Maximum attendance” and “Content management” goals. The “Maximum attendance” goal, at level 2, is also refined into one goal at level 3, namely, “Meeting scheduled from request”. The “meeting scheduled from request”, in level 3, is refined into two CLG goals which are “Polling for meeting” and “Email notification”.

Table 5 MSS goals' model

Business objective	Refined HLG	Refined HLG	CLG
Effective meeting	Maximum attendance	Meeting scheduled from request	CLG1: Polling for meeting
			CLG2: Email notification
	Content management		CLG3: File upload
			CLG4: Image upload
			CLG5: WYSIWYG text editing
			CLG6: Content searching
	Contact management		CLG7: Contact MSS administrator
			CLG8: Contact between MSS users

4.3 Acceptance Test Cases Generation

The acceptance test cases are generated to check the existence of the functionalities of each CLG. Best-fit components are selected based on the test cases. Acceptance test cases generation results in suites of test cases for each CLG in the Fit tables' format against which components are evaluated by testing how well the components satisfy the test's suites.

The tests suites are generated based on the MSS goals model as well as the gained knowledge from exploring the literature [54], [56] (2 in Figure 7). Finally, the test cases created with a total of 51: 17 for CLG1, 3 for CLG2, 6 for CLG3, 4 for CLG4, 17 for CLG5, 2 for CLG 6, 1 for CLG7, and 1 for CLG8. To give an idea about the test cases in this case study, we present the CLG1 Test Suite (CLG1TS), while the all test suites are given in Appendix A. CLG1TS is a test suite for testing CLG1: Polling for meeting. Table 6 describes the CLG1TS's Fixtures and respective CLG1TS's action and row fixtures are presented in Table 7.

Table 6 Descriptions of CLG1TS's fixtures

Fixture	Purpose
Fit.ActionFixture1	The purpose of this first fixture is to set up the title, location, description, dates, and related time slots to make polling for meeting.
Fit.ActionFixture2, 3, 4	These fixtures allow the participants to select the suitable date and time slots from the suggested ones in the polling request.
Fit.RowFixture1, 2, 3	The purpose of these fixtures is to assume there are three participants with their selected dates and time slots and check for their existence
Fit.RowFixture4	The purpose of these fixtures is to assume there are four polling requests in the system and check for their existence
Fit.ActionFixture5	These fixtures test the removing of a polling request
Fit.RowFixture5	The purpose of these fixtures is to assume there are three polling requests (not containing the removed one in the previous fixture) in the system and check for their existence

Table 7 CLGITS's fixtures

Fit.ActionFixture1		
Enter	Polling Title	Department meeting
Enter	Description	We want to discuss the department new policy
Enter	Location	Building 808 Room 207
Press	Date	10-5-2014
Enter	Time slot	2-3, 3-4, 4-6
Press	Date	11-5-2014
Enter	Time slot	2-3, 3-4, 4-6
Enter	Participants names	Participant1, Participant2, Participant3,
Check	Polling created	True

Fit.ActionFixture2		
Enter	Participant name	Participant1
Press	Date	10-5-2014
Enter	Time slot	2-3
Press	Date	11-5-2014
Enter	Time slot	2-3, 3-4
Check	Participant1 select date and time slot	True

Fit.ActionFixture3		
Enter	Participant name	Participant2
Press	Date	10-5-2014
Enter	Time slot	2-3
Press	Date	11-5-2014
Enter	Time slot	3-4
Check	Participant2 select date and time slot	True

Fit.ActionFixture4		
Enter	Participant name	Participant3
Press	Date	10-5-2014
Enter	Time slot	2-3
Press	Date	11-5-2014
Enter	Time slot	2-3
Check	Participant3 select date and time slot	True

Fit.RowFixture1		
Participant name	Selected Date	Time Slot(s)
Participant1	10-4-2014	2-3
Participant1	11-4- 2014	2-3, 3-4

Fit.RowFixture2

Participant name	Selected Date	Time Slot(s)
Participant2	10-4-2014	2-3
Participant2	11-4- 2014	3-4

Fit.RowFixture3

Participant name	Selected Date	Time Slot(s)
Participant3	10-4-2014	2-3
Participant3	11-4- 2014	2-3

Fit.RowFixture4

Meeting Polling 1
Meeting Polling 2
Meeting Polling 3
Meeting Polling 4

Fit.ActionFixture5

Enter	Polling name	Meeting Polling 1
press	Remove polling	
check	Polling removed	True

	Fit.RowFixture5	
	Meeting Polling 2	
	Meeting Polling 3	
	Meeting Polling 4	

4.4 COTS Searching and Filtering

Automatic search, based on goals model, performed by executing search strings on the drupal repository (www.drupal.org see Figure 8). The candidate components are specified through the keyword search based on CLGs in the goals model; this resulted in identifying an initial set of 25 COTS candidates classified according to their related CLGs into 8 categories (Table 8).

Table 8 CLGs and their potential candidate components

CLGs	Candidate Modules	Total
Polling for meeting	Flag, Rate, Advanced Poll, Decisions, Election, Make Meeting Scheduler	6
Notification	Comment Notify, Node Notify, Watcher, Notify, Subscriptions, Notifications	6
File upload	IMCE for FileField, FileField Sources, Multiupload Filefield	3
Image upload	IMCE, Image Fupload, Multiupload Imagefield	3
Rich WYSIWYG Editing	BUEditor, widgEditor, OpenWYSIWYG Editor, Wysiwyg	4
Content searching	Search	1
Contact MSS administrator	Contact	1
Contact between MSS users	Contact	1
		25



Figure 8 Drupal's search engine



Figure 9 Snapshot from Drupalmodules' website

Then, detailed information for each candidate component is collected from: (i) their drupal's websites and (ii) other online sources that offer information about drupal's components; e.g. <http://drupalmodules.com> (see Figure 9). The detailed information for each candidate component is presented in Table 9.

Table 9 Detailed information about candidate components

CLG	Module	Drupal7	installations	Required modules	Reported Open Bugs
Polling for meeting	Flag	1	33,987	0	5
	Rate	1	9,71	3	99
	Decisions	0	351	1	31
	Make Meeting Scheduler	1	550	0	5
	Election	1	75	2	2
	Advanced Poll	1	4,12	2	87
Notification	Comment notify	1	13960	1	21
	Node notify	1	201	2	0
	Watcher	0	653	2	34
	Notify	0	2761	3	3

	Subscriptions	1	6031	3	12
	Notifications	0	10163	2	224
Rich WYSIWYG Editing	BUEditor	1	13049	0	0
	widgEditor	0	91	0	5
	OpenWYSIWYG Editor	0	211	0	20
	Wysiwyg	1	338266	0	69
File uploade	IMCE for FileField	1	6,984	2	2
	FileField Sources	1	46,574	1	30
	Multiupload Filefield	1	28,073	1	9
Image uploade	IMCE	1	345,276	2	3
	Image Fupload	0	8,110	2	111
	Multiupload Imagefield	1	25,352	1	7

Then, the proposed filtering function (chapter 3, equation 3.1) is used to filter out less suitable components from the initial list of 25. Finally, the short list of the most suitable

products was specified (3 in Figure 7). The resulted shortlist of components of filtering process is presented in Table 10, while the detailed evaluation's results are shown in Appendix B.

Table 10 Shortlist of the most suitable components for the MSS case study

CLG	Module	Developer	Website
Polling for meeting	Flag	socketwench	https://drupal.org/project/flag
	Make meeting Schedular	SebCorbin	https://drupal.org/project/makemeeting
	Notification	greggles	https://drupal.org/project/comment_notify
	Subscriptions	salvis	https://drupal.org/project/subscriptions
Rich WYSIWYG Editing	BUEditor	ufku	https://drupal.org/project/BUEditor
	Wysiwyg	sun	https://drupal.org/project/Wysiwyg
Image upload	Multiupload Imagefield	czigor	https://drupal.org/project/multiupload_imagefield_widget
	IMCE	ufku	https://drupal.org/project/imce
Files upload	FileField	quicksketch	https://drupal.org/project/filefield_sources

	Sources		
	Multiupload Filefield	czigor	https://drupal.org/project/multiupload_filefield_widget
Content searching	Search	Drupal	https://drupal.org/documentation/modules/search/
Contact MSS administrator	Contact	Drupal	https://drupal.org/documentation/modules/contact/
Contact MSS users	Contact	Drupal	https://drupal.org/documentation/modules/contact/

4.5 Configure and Evaluate COTS

Each candidate component resulted from the shortlisted list is evaluated after installing and configuring it to the drupal environment. The resulted candidate components from filtering phase are evaluated by implementing the proposed evaluation function TCS (chapter 3, equation 3.3). The resulted evaluations of components (4 in Figure 7) are presented in Table 11 to Table 18. For each component, the TCS value is shown. For instance, evaluating the Flag component against the Fixture1 or Test Case1 (TC1) belongs to the test suite CLG1TS resulted in TCS=0, which means Flag component does not satisfy the TC1.

Table 11 CLG1TS evaluation's results

	CLG1TS																	Total Passed Tests	Total Failed Tests
	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12	TC13	TC14	TC15	TC16	TC17		
Flag	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	10	7
Make meeting scheduler	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	17	0

Table 12 CLG2TS evaluation's results

	CLG2TS				
	TC1	TC2	TC3	Total Passed Tests	Total Failed Tests
Comment notify	0	1	1	2	1
Subscriptions	1	1	1	3	0

Table 13 CLG3TS evaluation's results

	CLG3TS							Total Passed Tests	Total Failed Tests
	TC1	TC2	TC3	TC4	TC5	TC6			
FileField Sources	1	1	1	1	1	1	6	0	
Multiupload Filefield	1	1	1	1	1	1	6	0	

Table 14 CLG4TS evaluation's results

	CLG4TS					Total Passed Tests	Total Failed Tests
	TC1	TC2	TC3	TC4			
Multiupload Imagefield	1	1	1	1	4	0	
IMCE	1	1	1	1	4	0	

Table 15 CLG5TS evaluation's results

	CLG5TS																	Total Passed Tests	Total Failed Tests
	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12	TC13	TC14	TC15	TC16	TC17		
BUEditor	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	6	11
Wysiwyg	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	7	10

Table 16 CLG6TS evaluation's results

	CLG6TS			
	TC1	TC2	Total Passed Tests	Total Failed Tests
Search (Core module)	1	1	2	0

Table 17 CLG7TS evaluation's results

	CLG7TS		
	TC1	Total Passed Tests	Total Failed Tests
contact (Core module)	1	1	0

Table 18 CLG8TS evaluation's results

	CLG8TS		
	TC1	Total Passed Tests	Total Failed Tests
contact (Core module)	1	1	0

4.6 COTS Selection

In this section, the outcomes from applying the component selection activity in the AccepCotSel approach are presented. The main objective of this phase is selecting the best-fit component for each CLG based on the fitness of the components. The components' fitness is calculated using Equation 3.4 from Chapter 3. Table 19 presenting the fitness of each candidate component.

As shown in Table 19, the “Make meeting Scheduler” component has higher fitness than the “Flag” component, so the first one should be selected. However, “FileField Sources” and “Multiupload Filefield” have the same fitness. In this situation, “Multiupload Filefield” should be selected because it has less value of open reported bugs/installations, only 0.0003 open reported bugs/installations, in comparison with the “FileField Sources” which has 0.0006 open reported bugs/installations. Table 20 presents the most suitable component for each CLG. In the case of having only one candidate component for a CLG (e.g. Search component for Content searching CLG), if this component has a non-zero MS value it should be selected, otherwise not.

Table 19 Fitness of candidate components

CLG	Module	MS
Polling for meeting	Flag	0.59
	Make meeting Scheduler	1
Notification	Comment notify	0.67
	Subscriptions	1
Files upload	FileField Sources	1
	Multiupload Filefield	1
Image upload	Multiupload Imagefield	1
	IMCE	1
Rich WYSIWYG Editing	BUEditor	0.35
	Wysiwyg	0.41
Content searching	Search	1
Contact MSS administrator	Contact	1
Contact MSS users	Contact	1

Table 20 CLGs and their best-fit components

CLG	Best-fit COTS
Polling for meeting	Make meeting Scheduler
Notification	Subscriptions
Files upload	Multiupload Filefield
Image upload	IMCE
Rich WYSIWYG Editing	Wysiwyg
Content searching	Search
Contact MSS administrator	Contact
Contact MSS users	Contact

CHAPTER 5

CONCLUSION

5.1 Summary and Contributions of the Thesis

In this thesis, we have presented a survey of the literature of the role of acceptance test cases in the requirements clarification. Furthermore, we explored the current COTS selection approaches and gave a necessary background for better understanding the COTS selection practices.

We have described an approach that uses acceptance testing information as functional specification to facilitate systematic selection of software components for a CBS. Our approach, called AccepCotSel, consists of five phases, namely, requirements modeling, acceptance test cases creation, and search candidate COTS and filtering, generate and evaluate configurations, and select the suitable COTS component. AccepCotSel approach uses the goal modeling technique to specify requirements of a CBS. It also uses the Framework for integration test (FIT), an open sources framework for generating acceptance test cases, to create acceptance test cases for CBS-to-be.

We have evaluated the effectiveness of the approach on a case study for selecting components to build MSS.

5.2 Limitations and Future Work

The approach validation results are specialized to the implemented case study, so the findings can't be general since they are only gained from a single case study. We plan to do additional case studies in order to confirm generalize our findings.

Further work is required to extend the Fit framework to be used in automating testing in the content management systems like Drupal.

There is a need to handle and analyze the acceptance test cases failure during the selection process and taking suitable decisions on that is needed, so we plan to develop a decision support framework for handing the mismatches between the candidate components and the acceptance test cases.

REFERENCES

- [1] M. Jha and L. O'Brien, "A comparison of software reuse in software development communities," in *5th Malaysian Conference in Software Engineering (MySEC)*, 2011, pp. 313–318.
- [2] R. W. Selby, "Enabling reuse-based software development of large-scale systems," *Softw. Eng. IEEE Trans.*, vol. 31, no. 6, pp. 495–510, 2005.
- [3] P. Mohagheghi and R. Conradi, "An empirical investigation of software reuse benefits in a large telecom product," *ACM Trans. Softw. Eng. Methodol.*, vol. 17, no. 3, p. 13, 2008.
- [4] G. Bockle, P. Clements, J. D. McGregor, D. Muthig, and K. Schmid, "Calculating ROI for software product lines," *Software, IEEE*, vol. 21, no. 3, pp. 23–31, 2004.
- [5] A. Pleuss, G. Botterweck, D. Dhungana, A. Polzer, and S. Kowalewski, "Model-driven support for product line evolution on feature level," *J. Syst. Softw.*, 2011.
- [6] J. Rubin and M. Chechik, "Combining related products into product lines," *Fundam. Approaches to Softw. Eng.*, pp. 285–300, 2012.
- [7] M. A. Khan and S. Mahmood, "A graph based requirements clustering approach for component selection," *Adv. Eng. Softw.*, vol. 54, pp. 1–16, 2012.
- [8] C. Becker and A. Rauber, "Improving component selection and monitoring with controlled experimentation and automated measurements," *Inf. Softw. Technol.*, vol. 52, no. 6, pp. 641–655, 2010.
- [9] J. Kontio, "A case study in applying a systematic method for COTS selection," *Proc. IEEE 18th Int. Conf. Softw. Eng.*, pp. 201–209, 1996.
- [10] P. Oberndorf, "COTS and Open Systems," *SEI Monogr. Use Commer. Softw. Gov. Syst. Softw. Eng. Institute. Carnegie Mellon Univ. Pittsburgh, Pennsylvania*, vol. 15213, 1998.
- [11] "SEI: Software Engineering Institute in Carnegie Mellon Univ." [Online]. Available: <http://www.sei.cmu.edu>.
- [12] N. Soundarajan and J. Hallstrom, "Responsibilities and rewards: Reasoning about design patterns," *Proc. 26th Int. Conf. Softw. Eng.*, 2004.

- [13] J. K. H. Mak, C. S. T. Choy, and D. P. K. Lun, "Precise modeling of design patterns in UML," in *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, 2004, pp. 252–261.
- [14] R. Kazman, G. Abowd, L. Bass, and P. Clements, "Scenario-based analysis of software architecture," *Software, IEEE*, vol. 13, no. 6, pp. 47–55, 1996.
- [15] "CeBASE: Center for Empirically Based Software Engineering." [Online]. Available: www.cebase.org.
- [16] S. Mahmood and R. Lai, "Analyzing component based system specification," in *Proc. of AWRE*, 2006.
- [17] N. A. Maiden and C. Ncube, "Acquiring COTS software selection requirements," *Software, IEEE*, vol. 15, no. 2, pp. 46–56, 1998.
- [18] K. R. P. H. Leung and H. K. N. Leung, "On the efficiency of domain-based COTS product selection method," *Inf. Softw. Technol.*, vol. 44, no. 12, pp. 703–715, 2002.
- [19] C. Alves and A. Finkelstein, "Investigating conflicts in COTS decision-making," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 13, no. 05, pp. 473–493, 2003.
- [20] J. K. Lee, S. J. Jung, S. D. Kim, W. H. Jang, and D. H. Ham, "Component identification method with coupling and cohesion," in *Software Engineering Conference, 2001. APSEC 2001. Eighth Asia-Pacific*, 2001, pp. 79–86.
- [21] H. Jain, N. Chalimeda, N. Ivaturi, and B. Reddy, "Business component identification-a formal approach," in *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International*, 2001, pp. 183–187.
- [22] M. H. M. O. B. R. C. Wohlin P. Runeson and A. Wesslen, "Experimentation in software engineering: an introduction," 2000.
- [23] F. Ricca, M. Torchiano, M. Di Penta, M. Ceccato, and P. Tonella, "Using acceptance tests as a support for clarifying requirements: A series of experiments," *Inf. Softw. Technol.*, vol. 51, no. 2, pp. 270–283, 2009.
- [24] K. Beck, *Test-driven development: by example*. Addison-Wesley, 2003.
- [25] G. Melnik, K. Read, and F. Maurer, "Suitability of fit user acceptance tests for specifying functional requirements: Developer perspective," *Extrem. Program. Agil. methods-XP/Agile Universe 2004*, pp. 638–663, 2004.

- [26] C. Marhold, C. Rohleder, C. Salinesi, and J. Doerr, "Clarifying Non-functional Requirements to Improve User Acceptance--Experience at Siemens," *Requir. Eng. Found. Softw. Qual.*, pp. 139–146, 2009.
- [27] W. Cunningham and R. Mugridge, *Fit for Developing Software: Framework for Integrated Tests*. Prentice-Hall, 2005.
- [28] M. R. J. Qureshi and S. A. Hussain, "A reusable software component-based development process model," *Adv. Eng. Softw.*, vol. 39, no. 2, pp. 88–94, 2008.
- [29] A. Schmietendorf, E. Dimitrov, and R. R. Dumke, "Process models for the software development and performance engineering tasks," in *Proceedings of the 3rd international workshop on Software and performance*, 2002, pp. 211–218.
- [30] I. Crnkovic and M. Larsson, "Challenges of component-based development," *J. Syst. Softw.*, vol. 61, no. 3, pp. 201–212, 2002.
- [31] IEEE, "IEEE Standard for Software Verification and Validation Plans," *IEEE Std 1012-1986*, 1986.
- [32] J. C. Dean and M. R. Vigder, "System Implementation Using Commercial off-the-shelf (COTS) Software," in *The 1997 Software Technology Conference (STC '97), Salt Lake City, Utah, USA*, 1997.
- [33] G. Ruhe, "Intelligent Support for Selection of COTS Products," in *Web, Web-Services, and Database Systems*, Springer, 2003, pp. 34–45.
- [34] B. George and L. Williams, "A structured experiment of test-driven development," *Inf. Softw. Technol.*, vol. 46, no. 5, pp. 337–342, 2004.
- [35] R. Kaufmann and D. Janzen, "Implications of test-driven development: a pilot study," in *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, 2003, pp. 298–299.
- [36] D. Janzen and H. Saiedian, "Test-driven development concepts, taxonomy, and future direction," *IEEE Comput.*, vol. 38, no. 9, pp. 43–50, 2005.
- [37] A. Sami, "DS for selecting COTS software products based on comprehensive mismatch handling," no. April, 2007.
- [38] D. Carney, "COTS evaluation in the real world," *SEI Interactive, Carnegie Mellon Univ.*, 1998.
- [39] J. Kontio, G. Caldiera, and V. R. Basili, "Defining Factors , Goals and Criteria for Reusable Component Evaluation Abstract : Figure 1 : The main phases in the OTSO process," pp. 1–12, 1996.

- [40] J. P. Franch, Xavier and Carvalho, "Using quality models in software package selection," *Software, IEEE*, vol. 20, pp. 34–41, 2003.
- [41] "ISO/IEC 9126-1:2001 Software engineering -- Product quality -- Part 1: Quality model," Geneva, Switzerland, 2001.
- [42] D. Kunda and L. Brooks, "Identifying and classifying processes (traditional and soft factors) that support COTS component selection: a case study," *Eur. J. Inf. Syst.*, vol. 9, no. 4, pp. 226–234, Dec. 2000.
- [43] L. Chung and K. Cooper, "Matching , Ranking , and Selecting Components : A COTS-Aware Requirements Engineering and Software Architecting Approach," in *International Workshop on Models and Processes for the Evaluation of COTS Components at ICSE*, 2004, pp. 41 –44.
- [44] C. Alves and J. Castro, "CRE: A systematic method for COTS components selection," in *XV Brazilian Symposium on Software Engineering (SBES)*, 2001.
- [45] J. Clark, C. Clarke, S. De Panfilis, G. Granatella, P. Predonzani, A. Sillitti, G. Succi, and T. Vernazza, "Selecting components in large COTS repositories," *J. Syst. Softw.*, vol. 73, no. 2, pp. 323–331, 2004.
- [46] A. Cechich and M. Piattini, "Early detection of COTS component functional suitability," *Inf. Softw. Technol.*, vol. 49, no. 2, pp. 108–121, 2007.
- [47] M. Harman, A. Skaliotis, K. Steinhöfel, and P. Baker, "Search--based approaches to the component selection and prioritization problem," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 1951–1952.
- [48] S. A. Fahmi and H.-J. Choi, "A study on software component selection methods," in *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, 2009, vol. 1, pp. 288–292.
- [49] D. Birkmeier and S. Overhage, "On component identification approaches--classification, state of the art, and comparison," *Component-Based Softw. Eng.*, pp. 1–18, 2009.
- [50] A. Lozano-tello and F. D. I. Lia, "BAREMO : How to Choose the Appropriate Software Component Using the Analytic Hierarchy Process 1," 2002, pp. 1–8.
- [51] P. T. Golden, Bruce L and Wasil, Edward A and Harker, "The analytic hierarchy process," *Springer*, 1989.
- [52] S. Dardenne, Anne and Van Lamsweerde, Axel and Fickas, "Goal-directed requirements acquisition," *Sci. Comput. Program.*, vol. 20, pp. 3–50, 1993.

- [53] C. Rolland, C. Souveyet, and C. Ben Achour, “Guiding goal modeling using scenarios,” *Softw. Eng. IEEE Trans.*, vol. 24, no. 12, pp. 1055–1071, 1998.
- [54] A. van Lamsweerde, *Requirements Engineering From System Goals to UML Models to Software Specifications*. Wiley, 2009.
- [55] A. Wardlaw, *Practical Statistics*. John Wiley & Sons Ltd, West Sussex, England, 2000.
- [56] A. Van Lamsweerde, R. Darimont, and E. Letier, “Managing Conflicts in Goal-Driven Requirements Engineering,” vol. 24, no. 11, pp. 908–926, 1998.

Appendix A: Detailed Evaluation's Test Suites

CLG1: Polling for meeting Test Suite (CLG1TS)

CLG1TS is a test suite for testing CLG1: Polling for meeting. Table 21 describes the CLG1TS's Fixtures and respective CLG1TS's action and row fixtures are presented in Table 22.

Table 21 Description of CLG1TS's fixtures

Fixture	Purpose
Fit.ActionFixture1	The purpose of this first fixture is to set up the title, location, description, dates, and related time slots to make polling for meeting.
Fit.ActionFixture2, 3, 4	These fixtures allow the participants to select the suitable date and time slots from the suggested ones in the polling request.
Fit.RowFixture1, 2, 3	The purpose of these fixtures is to assume there are three participants with their selected dates and time slots and check for their existence
Fit.RowFixture4	The purpose of these fixtures is to assume there are four polling requests in the system and check for their existence
Fit.ActionFixture5	These fixtures test the removing of a polling request

Fit.RowFixture5	The purpose of these fixtures is to assume there are three polling requests (not containing the removed one in the previous fixture) in the system and check for their existence
-----------------	--

Table 22 CLG1TS's fixtures

Fit.ActionFixture1		
Enter	Polling Title	Department meeting
Enter	Description	We want to discuss the department new policy
Enter	Location	Building 808 Room 207
Press	Date	10-5-2014
Enter	Time slot	2-3, 3-4, 4-6
Press	Date	11-5-2014
Enter	Time slot	2-3, 3-4, 4-6
Enter	Participants names	Participant1, Participant2, Participant3,
Check	Polling created	True

Fit.ActionFixture2		
Enter	Participant name	Participant1
Press	Date	10-5-2014
Enter	Time slot	2-3
Press	Date	11-5-2014

Enter	Time slot	2-3, 3-4
Check	Participant1 select date and time slot	True

Fit.ActionFixture3		
Enter	Participant name	Participant2
Press	Date	10-5-2014
Enter	Time slot	2-3
Press	Date	11-5-2014
Enter	Time slot	3-4
Check	Participant2 select date and time slot	True

Fit.ActionFixture4		
Enter	Participant name	Participant3
Press	Date	10-5-2014
Enter	Time slot	2-3
Press	Date	11-5-2014
Enter	Time slot	2-3
Check	Participant3 select date and time slot	True

Fit.RowFixture1		
Participant name	Participant1	
10-4-2014	2-3	

11-4- 2014	2-3	3-4
------------	-----	-----

Fit.RowFixture2	
Participant name	Participant2
10-4-2014	2-3
11-4- 2014	3-4

Fit.RowFixture3	
Participant name	Participant3
10-4-2014	2-3
11-4- 2014	2-3

Fit.RowFixture4	
Meeting Polling 1	
Meeting Polling 2	
Meeting Polling 3	
Meeting Polling 4	

Fit.ActionFixture5		
Enter	Polling name	Meeting Polling 1
press	Remove polling	
check	Polling removed	True

Fit.RowFixture5	
Meeting Polling	2
Meeting Polling	3
Meeting Polling	4

CLG2: Email Notification Test Suite (CLG2TS)

CLG2TS is a test suite for testing CLG2: Email Notification. Table 23 describes the CLG2TS’s Fixtures and respective CLG2TS’s action and row fixtures are presented in Table 24.

Table 23 Description of CLG2TS’s fixtures

Fixture	Purpose
Fit. ColumnFixture1	The purpose of this fixture is to notify participants about a new meeting through their emails.
Fit. ColumnFixture2	The purpose of this fixture is to notify a meeting’s initiator about a new comment on the meeting and notify a meeting’s participants who have commented about new comments after them.
Fit. ColumnFixture3	The purpose of this fixture is to notify meeting’s participants about an updating in the meeting through their emails.

Table 24 CLG2TS's fixtures

Fit.ColumnFixture1			
Initiator	Created meeting	Notify()	
InitiatorA	Meeting1	Participant 1 Participant 2 Participant 3	

Fit.ColumnFixture2		
Editor	Edit meeting	Notify()
InitiatorA	Meeting1	Participant 1 Participant 2 Participant 3

Fit.ColumnFixture3			
Participant	Selected meeting	Comment	Notify()
Participant 1	Meeting1	Comment test1	InitiatorA
Participant 2	Meeting1	Comment test2	InitiatorA Participant 1

CLG3: File uploads Test Suite (CLG3TS)

CLG3TS is a test suite for testing the file uploading CLG Table 25 describes the CLG3TS's Fixtures and respective CLG3TS's action and row fixtures are presented in Table 26.

Table 25 Description of CLG3TS's fixtures

Fixture	Purpose
Fit.ActionFixture1	This fixture tests the uploading of pdf files by the users.
Fit.ActionFixture2	This fixture tests the uploading of MS word files by the users.
Fit.ActionFixture3	This fixture tests the uploading of text files by the users.
Fit.ActionFixture4	This fixture tests the uploading of multiple pdf files by the users.
Fit.ActionFixture5	This fixture tests the uploading of multiple docx files by the users.
Fit.ActionFixture6	This fixture tests the uploading of multiple text files by the users.

Table 26 CLG3TS's fixtures

Fit.ActionFixture1		
Press	Browse	
press	Select file	ABC1
press	File Type	pdf
Press	Upload	
check	Uploaded	True

Fit.ActionFixture2		
Press	Browse	
press	Select file	ABC1
press	File Type	docx
Press	Upload	
check	Uploaded	True

Fit.ActionFixture3		
Press	Browse	
press	Select file	ABC1
press	File Type	txt

Press	Upload	
check	Uploaded	True

Fit.ActionFixture4		
Press	Browse	
press	Select file	ABC1
press	File Type	pdf
press	Select file	ABC2
press	File Type	pdf
Press	Upload	
check	Uploaded	True

Fit.ActionFixture5		
Press	Browse	
press	Select file	ABC1
press	File Type	docx
press	Select file	ABC2
press	File Type	docx
Press	Upload	
check	Uploaded	True

Fit.ActionFixture6		
Press	Browse	
press	Select file	ABC1
press	File Type	txt
press	Select file	ABC2
press	File Type	txt
Press	Upload	
check	Uploaded	True

CLG4: Image uploads Test Suite (CLG4TS)

CLG4TS is a test suite for testing the image uploading CLG. Table 27 describes the CLG4TS's Fixtures and respective CLG4TS's action and row fixtures are presented in Table 28.

Table 27 Description of CLG4TS's fixtures

Fixture	Purpose
Fit.ActionFixture1	This fixture test the uploading of jpg images by the MSS users.
Fit.ActionFixture2	This fixture tests the uploading of png images by the MSS users.
Fit.ActionFixture3	This fixture tests the uploading of gif images by the MSS users.
Fit.ActionFixture4	This fixture test the uploading multiple images of different types by the users.

Table 28 CLG4TS's fixtures

Fit.ActionFixture1		
Press	Browse	
press	Select image	ABC1
press	Image Type	jpg
Press	Upload	
check	Uploaded	True

Fit.ActionFixture2		
Press	Browse	
press	Select image	ABC1
press	Image Type	png
Press	Upload	
check	Uploaded	True

Fit.ActionFixture3		
Press	Browse	
press	Select image	ABC1
press	Image Type	gif
Press	Upload	

check	Uploaded	True
-------	----------	------

Fit.ActionFixture4		
Press	Browse	
press	Select image	ABC1
press	Image Type	gif
press	Select image	ABC2
press	Image Type	png
press	Select image	ABC3
press	Image Type	jpg
Press	Upload	
check	Uploaded	True

CLG5: WYSIWYG Text Editing Test Suite (CLG5TS)

CLG5TS is a test suite for testing the WYSIWYG Text Editing CLG. Table 29 describes the CLG5TS's Fixtures and respective CLG5TS's action and row fixtures are presented in

Table 30.

Table 29 Description of CLG5TS's fixtures

Fixture	Purpose
Fit.ActionFixture1, 2, 3, and 4	These fixtures test the decoration of tests with making it bold, italic, underlined, and changing the text color.
Fit.ActionFixture5, 6, and 7	These fixtures test the alignment of texts (i.e. left, right, and center alignment)
Fit.ActionFixture8, 9, 10, 11, 12, and 13	These fixtures test changing the texts font and size
Fit.ActionFixture14, 15	These fixtures test the functionality of inserting an image and table to the texts.
Fit.ActionFixture16, 17	These fixtures test the functionalities of copy, cut, and paste texts.

Table 30 CLG5TS's fixtures

Fit.ActionFixture1		
Enter	Text	Meeting scheduling
press	Select	Meeting scheduling
press	Bold	
check	Textbold	True

Fit.ActionFixture2		
Enter	Text	Meeting scheduling
press	Select	Meeting scheduling
press	Italic	
check	Text Italic	True

Fit.ActionFixture3		
Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Underline	
check	TextUnderline	True

Fit.ActionFixture4		
---------------------------	--	--

Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Font color	Red
check	Text color changed	True

Fit.ActionFixture5		
Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Align	left
check	Text left Aligned	True

Fit.ActionFixture6		
Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Align	center
Check	Text center Aligned	True

Fit.ActionFixture7		
Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling

Press	Align	right
Check	Text right Aligned	True

Fit.ActionFixture8		
Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Font type	Proxima Nova
Check	Text font changed	True

Fit.ActionFixture9		
Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Font type	Gotham
Check	Text font changed	True

Fit.ActionFixture10		
Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Font type	Roboto
Check	Text font changed	True

Fit.ActionFixture11

Enter	Text	Meeting scheduling
press	Select	Meeting scheduling
press	Font type	Inconsolata
check	Text font changed	True

Fit.ActionFixture12

Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Font type	Avenir
check	Text font changed	True

Fit.ActionFixture13

Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Font Size	14
check	Text font Size changed	True

Fit.ActionFixture14

Press	Table
-------	-------

Enter	Rows	3
Enter	Columns	4
Check	Table Created	True

Fit.ActionFixture15

Press	Insert image	
Press	Select	ImageA
check	Image inserted	True

Fit.ActionFixture16

Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Copy	
Press	Paste	
Check	Text pasted	True

Fit.ActionFixture17

Enter	Text	Meeting scheduling
Press	Select	Meeting scheduling
Press	Cut	

Press	Paste	
Check	Text pasted	True

CLG6: Content Searching Test Suite (CLG6TS)

CLG6TS is a test suite for testing the content searching CLG. Table 31 describes the CLG6TS's fixtures and respective CLG6TS's action and row fixtures are presented in Table 32.

Table 31 Description of CLG6TS' fixture

Fixture #	Type	Purpose
Fit.ActionFixture1		The purpose of this fixture is to search for a poll using a keyword searching
Fit.ActionFixture1		The purpose of this fixture is to search about events in a specific date.

Table 32 CLG6TS' fixture

Fit.ActionFixture1		
Enter	Search Text	polling
Press	Search	
Check	Search results displayed	True

Fit.ActionFixture2		
Enter	Search by date	Wed, 05/07/2014
Press	Search	
Check	Search results displayed	True

CLG7: Contact site administrator Test Suite (CLG7TS)

CLG7TS is a test suite for testing the contact site administrator CLG.

Table 33 describes the CLG7TS's fixture and respective CLG7TS's action fixture is presented in Table 34.

Table 33 Description of CLG7TS' fixture

Fixture #	Type
Fit.ActionFixture1	The purpose of this fixture is contacting site administrator. It tests how a user can communicate with the site administrator through providing his name, email, subject, and the message body, respectively.

Table 34 of CLG7TS' fixture

Fit.ActionFixture1		
Enter	Name	User1
Enter	Email	User1@yahoo.com
Enter	Subject	Request for privileges
Enter	Message Body	Please, give me privileges to create pollings
press	Send	
check	Message Sent	True

CLG8: Contact between users Test Suite (CLG8TS)

CLG8TS is a test suite for testing the contact between MSS users CLG. Table 35 describes the CLG8TS's fixture and respective CLG7TS's action fixture is presented in Table 36.

Table 35 Description of CLG8TS' fixture

Fixture #	Type
Fit.ActionFixture1	The purpose of this fixture is contacting between the site users. It tests how a user can communicate with another through providing his name, email, subject, message body, receiver name, and the receiver email.

Table 36 CLG8TS' fixture

Fit.ActionFixture1		
Enter	Name	User1
Enter	Email	User1@yahoo.com
Enter	Subject	Request for privileges
Enter	Message Body	Please, give me privileges to create polling
Enter	Receiver(s) Name	User2
Enter	Receiver(s) email	User2@yahoo.com

press	Send	
check	Message Sent to Receiver(s)	True

Appendix B: Filtering Evaluation's Results

Table 37 Filtering Evaluation's Results for Polling for meeting components

Module	Drupal (7)	Installations	Installations <i>Normalized</i>	Required Modules	Required modules <i>Normalized</i>	Open Bugs	Open Bugs <i>Normalized</i>	FS()
Flag	1	33,987	1.000	0	0.000	5	0.031	1.940
Rate	1	971	0.026	3	1.000	99	1.000	0.342
Advanced Poll	1	412	0.010	2	0.667	87	0.876	0.397
Decisions	0	351	0.008	1	0.333	31	0.299	0.000
Make meeting Scheduler	1	550	0.014	0	0.000	5	0.031	0.984
Election	1	75	0.000	2	0.667	2	0.000	0.600

Table 38 Filtering Evaluation's Results for Notification components

Module	Drupal (7)	Installations	Installations <i>Normalized</i>	Required Modules	Required Modules <i>Normalized</i>	Open Bugs	Open Bugs <i>Normalized</i>	FS()
Comment notify	1	13960	1.000	1	0.000	21	0.094	1.829
Node notify	1	201	0.000	2	0.500	0	0.000	0.667
Watcher	0	653	0.033	2	0.500	34	0.152	0.000
Notify	0	2761	0.186	3	1.000	3	0.013	0.000
Subscriptions	1	6031	0.424	3	1.000	12	0.054	0.693
Notifications	0	10163	0.724	2	0.500	224	1.000	0.000

Table 39 Filtering Evaluation's Results for File upload components

Module	Drupal (7)	Installations	Installations <i>Normalized</i>	Required Modules	Required Modules <i>Normalized</i>	Open Bugs	Open Bugs <i>Normalized</i>	FS()
IMCE for FileField	1	6,984	0.000	2	1.000	2	0.000	0.500
FileField Sources	1	46,574	1.000	1	0.000	30	1.000	1.000
Multiupload Filefield	1	28,073	0.533	1	0.000	9	0.250	1.226

Table 40 Filtering Evaluation's Results for Image upload components

Module	Drupal (7)	Installations	Installations <i>Normalized</i>	Required Modules	Required Modules <i>Normalized</i>	Open Bugs	Open Bugs <i>Normalized</i>	FS()
IMCE	1	345,276	1.000	2	1.000	3	0.000	1.000
Image Fupload	0	8,110	0.000	2	1.000	111	1.000	0.000
Multiupload Imagefield	1	25,352	0.051	1	0.000	7	0.037	1.014

Table 41 Filtering Evaluation's Results for Rich WYSIWYG Editing components

Module	Drupal (7)	Installations	Installations <i>Normalized</i>	Required Modules	Required Modules <i>Normalized</i>	Open Bugs	Open Bugs <i>Normalized</i>	FS()
BUEditor	1	13049	0.038	0	0.500	0	0.000	0.692
widgEditor	0	91	0.000	0	0.500	5	0.072	0.000
Open WYSIWYG Editor	0	211	0.000	0	0.500	20	0.290	0.000
Wysiwyg	1	338266	1.000	0	0.500	69	1.000	0.800

VITAE

Name : Muaadh Abdulghani Qaid

Nationality : Yemeni

Date of Birth : 1/1/1984

Tel. : +966 566239932

: +967 777035032

Academic Background :

- Received Bachelor of Information Systems– King Khalid University (KKU) – Abha, Saudi Arabia , 2008 with a GPA of 4.72/5.
- Joint the Information and Computer science department as full time student at King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia in September 2009.
- Completed Master of science (M.S.) in Computer science from King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia in June 2014, with a GPA of 3.56/4.0
- Email: muaadhnoman@gmail.com, g200905130@kfupm.edu.sa