OPEN SOURCE FRAMEWORK USAGE: AN INVESTIGATION OF THE USER'S INTENTION TO CONTINUE USING A FRAMEWORK

ALEXANDRU LEMNARU
B.Sc. Academy of Economic Studies, 2009

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTERS OF SCIENCE IN MANAGEMENT

Management
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

Abstract

To increase productivity, application developers are using tools that allow them to create higher quality applications faster. One such set of tools, open-source frameworks, allows application developers to reuse software artifacts and should increase application quality. However, given the vast number of open-source frameworks available, users must be able to differentiate among frameworks and select the one best suited for them. In this study, we expand the taxonomy of open-source frameworks and analyze the impact of the framework's characteristics, technical quality, and social pressure on perceived usefulness and continued framework usage intention. Our findings suggest that understandability and flexibility have a significant impact on perceived ease of use, while perceived usefulness is mainly determined by flexibility and efficiency. Our research can be used to understand what influences developers to continue using frameworks and to improve framework development.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction and Problem Statement

Designing and developing successful systems is one of the most important goals of systems analysis and design and of the information systems (IS) field (Boehm, 1999; DeLone & McLean, 2003; Frakes & Kang, 2005). From a management perspective, in a study of the top information technology (IT) management concerns for 2009, Luftman and Ben-Zvi (2010) identified IT cost reduction and IT reliability and efficiency among the top 10 priorities for IT managers. Given the effects of the economic recession, IT managers are looking for research on ways to improve software development quality, efficiency, and productivity (Luftman & Ben-Zvi, 2010).

From an application developer's perspective, given the growing complexity of both technology and requirements, new tools are required to help create applications faster and with greater quality (Srinivasan, 1999; van Gurp & Bosch, 2001). To that end, researchers (e.g., Boehm, 1999; Frakes & Kang, 2005; Sindre, Conradi, & Karlsson, 1995; Srinivasan, 1999) have suggested three major approaches to improve software development: (a) speed up development by using automation, (b) improve the software development process, and (c) reuse software.

In response, software developers have created and used different software reuse-based tools that help them develop better software packages (Polančič, Heričko, & Pavlič, 2011; Srinivasan, 1999). One such set of tools has been collectively termed object oriented frameworks, software frameworks or just frameworks. A framework can be defined as "a semicomplete application that contains certain fixed aspects common to all applications in the problem domain, along with certain variable aspects unique to each application generated from it" (Srinivasan, 1999, p. 24).

Frameworks address all three suggested major approaches to help improve software development productivity and quality. They provide access to a tested core set of functions (Srinivasan, 1999), in the form of software classes and libraries, which can be reused through an Application Programming Interface (API). This allows framework users to automate and optimize part of the development process by reusing software artifacts (van Gurp & Bosch, 2001). Each new application becomes an instance of the framework, which allows for reuse of classes and features of the framework, while also permitting framework users to create new features specific to each application (Srinivasan, 1999). These potential improvements in productivity and quality make framework-based development one of the most promising approaches for improving application development (Frakes & Kang, 2005; Polančič et al., 2011).

With the development of the Internet and new software development methodologies, frameworks have evolved as well, and a new type, open-source frameworks, has appeared (Weber, 2004). While these frameworks have similar characteristics and benefits to their commercial counter-parts, they are open-source software and share the characteristics of public source code (Polančič et al., 2011; Polančič, Heričko, & Rozman, 2010). To add functionality, users are able to modify and upgrade a framework, and then share their contributions with the framework's community, thus becoming framework developers themselves (Polančič et al., 2011; Polančič et al., 2010). While open-source frameworks only appeared in the last 10 years, their number and importance have grown significantly. In 2007, Polančič et al. (2011) identified over 5,000 open-source frameworks, with 10,000 software developers actively developing them.

While some studies (Frakes & Kang, 2005; Polančič et al., 2011) have reported on the advantages of using frameworks for application development, others (Srinivasan, 1999; van Gurp & Bosch, 2001) have identified a number of challenges that application developers must overcome when using frameworks. Additionally, Polančič et al. (2011, p. 1) found that "many frameworks and framework-related projects still fail, which indicates that frameworks still have unsolved problems."

Given the challenges related to framework technology and open-source framework usage, the question of what makes a framework successful needs to be studied (Polančič et al., 2010). In their updated IS success model, DeLone and McLean (2003) proposed that system use, net benefits and satisfaction are the three main dimensions of the success of an information system. Following this research, Polančič et al. (2010) suggested that the user's intention to continue using a framework is a suitable proxy for system use. Polančič et al. (2010) found that the user's intention to continue using a framework and perceptions of the usefulness of the framework are antecedents of net benefits and satisfaction. Thus, the authors concluded that the user's intention to continue using a framework and the user's perceptions of the usefulness of a framework are key determinants for framework success.

While this establishes links from perceived usefulness and continued usage intention to DeLone and McLean's (2003) dimensions of IS success, we know little about the antecedents of these factors. Previous studies (Polančič et al., 2011; Polančič et al., 2010; Polančič, Horvat, & Rozman, 2009) identified some of these antecedents, in particular some of the framework and individual characteristics, and then proposed that further research is required to validate their suggested antecedents, as well as to identify new ones. Thus, the following questions arise:

(a) What factors influence a framework user's intention to continue using an open-source framework?

(b) How should framework technical quality, which Polancic et al. (2010) proposed would increase perceived ease of use and perceived usefulness, be measured?

(c) What factors influence a framework user's perception of the usefulness of an open-source framework?

(d) Does social pressure (in the open-source environment) influence the framework user's intention to continue using a framework?

From a theoretical perspective, answering these questions will expand our understanding of frameworks and the factors influencing the intention to continue using a framework and the framework's success. Thus, we contribute to the existing literature in five important ways.

First, previous studies on open-source frameworks recommended additional studies to discover the antecedents to both the framework’s perceived usefulness and the intention to continue using a framework (Polančič et al., 2011). By studying possible antecedents, our study will add to the literature on the technology acceptance model (TAM) (Davis, 1989) and help framework developers and IT managers understand factors affecting the intent to continue using a framework.

Second, as suggested by King & He (2006), our study includes the construct of social pressure, which is new to the framework literature. Since the framework’s online community plays an important role in the development of the open-source framework, (Polančič et al., 2011), the community might also play an important role in the user’s intention to continue using a framework. We intend to measure the impact of social pressure on a framework's perceived usefulness, ease of use, and the user's continued usage intention.

Third, several studies on frameworks suggest a number of guidelines to improve the quality of frameworks (van Gurp & Bosch, 2001). However, there is a lack of research that has looked at these guidelines and analyzed the impact of the framework's technical quality on the perceived usefulness of the framework and the intention to continue using it. We intend to bridge this gap by analyzing the impact of the framework's technical quality on the user's (i.e., application developer's) intention to continue using the framework and the framework's perceived usefulness.

Fourth, several studies have extended the original TAM model to include antecedents or factors from other theories (King & He, 2006). Our study looks at the antecedents of continued framework usage intention and provides a contribution to the literature on post-adoption usage models. Based on the classification of King and He (2006), our study provides both type 1 and 2 extensions to the original TAM model.

Fifth, our study provides methodological refinements to previous framework studies, including refinements to data collection and measurement. For example, we expand on the previous conceptualization and operationalization of the constructs of flexibility and portability. We also develop a formative measurement for the framework technical quality construct and refine it through the pre-test and main study.

From a practitioner's perspective, this study should help us better understand factors affecting a user's perception of the usefulness of the framework and his or her continued framework usage intention, which is a proxy for the success of a framework. Conversely, it should help framework creators understand what factors influence their users to continue using their product. In turn, our findings can be used to improve the quality of frameworks and to increase their success. Additionally, IT managers can use this research to improve productivity with frameworks by better understanding their employees' requirements.

To accomplish our goals, this study utilizes the TAM theoretical model in a post-adoption scenario. The research consists of two phases: (a) a pre-test to fine tune the instrument and (b) the main survey. The sample frame included open-source framework users from the SourceForge database (SourceForge, 2011). Access to the data was secured through an agreement with the SourceForge Research Database (Van Antwerp & Madey, 2008). Data was collected via an online survey created through the Qualtrics software application. The statistical analysis technique was structural equation modeling (SEM) and data analysis was performed with the SPSS and Amos software packages. Our findings suggest that the framework's understandability and flexibility have a significant impact on perceived ease of use, while perceived usefulness is mainly determined by the framework's flexibility and efficiency. Continued framework usage intention is influenced by perceived usefulness and social pressure. While confidence is influenced by both the framework's suitability and understandability, it doesn't influence perceived usefulness or continued usage intention.

In the next section of this study, the literature review, we present how the framework literature evolved from software reuse and provide an in-depth analysis of frameworks and existing research. Following the literature review, we analyze existing theoretical models, present our selected model, and develop our hypotheses. The fourth chapter contains the methodology of the pre-test and main study. In the fifth chapter we present the results of our study, while in the last chapter we discuss our findings and provide an overall summary of our research.

Chapter 2: Literature Review

Overview of Frameworks

The concept of software reuse, or the reuse of software artifacts, was first introduced by McIlroy (1968). He proposed that the software industry could benefit from the reuse of components and that it should be based on such components. Following this research, Parnas (1976) suggested that program families could be used to increase the productivity and quality of software development. According to the author, program families are "sets of programs whose common properties are so extensive that it is advantageous to study the common properties of the programs before analyzing individual members" (Parnas, 1976, p. 1). Following these initial contributions to software reuse research, the field has expanded significantly with research into software reuse libraries, software components, code generators, reuse design principles, etc. (Frakes & Kang, 2005). These contributions are the main pillars on which frameworks have been developed.

Based on the research on software reuse, several authors (e.g., Boehm, 1999; Frakes & Kang, 2005; Sindre et al., 1995) proposed guidelines for new tools that would allow software developers to increase their productivity and the quality of their software. Based on their guidelines, a set of tools labeled "frameworks" began to appear (Polančič et al., 2011; Srinivasan, 1999). With the development of the Internet, frameworks have developed even further and a new type, open-source frameworks, has appeared (Weber, 2004).

We previously defined a framework as "a semicomplete application" (Srinivasan, 1999, p. 24). However, a framework is better viewed as a set of building blocks that can be used and reused for creating any number of applications. Since applications in the same domain share a number of features or requirements, utilizing a framework that already meets these requirements decreases the development time of each application (Srinivasan, 1999). Apart from their code reuse advantage, frameworks facilitate standardization of development procedures across applications and also allow framework users to manage and upgrade their applications more easily (Boehm, 1999; van Gurp & Bosch, 2001). For example, application developers who wish to add new functionality to their applications need only upgrade the component of the framework that provides the required functionality. Since the same framework can be used in several applications, the user doesn't need to program the new functionality for each application individually (Srinivasan, 1999).

Open Source

The open-source movement appeared as a response to the issue of increased "closed source" proprietary software in the 1970s and 1980s. According to Weber (2004):

> Many of the best programmers were hired away into lucrative positions in spin-off software firms. MIT began to demand that its employees sign nondisclosure agreements. The newest mainframes came with operating systems that did not distribute source code—in fact, researchers had to sign nondisclosure agreements simply to get an executable copy. (p. 46)

In 1984, as a response to this issue, Richard Stallman created the Free Software Foundation and identified the four essential freedoms for "free" software development:

> (1) Freedom to run the program for any purpose; (2) Freedom to study how the program works and to modify it to suit your needs; (3) Freedom to redistribute copies, either gratis or for a monetary fee; (4) Freedom to change and improve the

program and to redistribute modified versions of the program to the public so others can benefit from your improvements. (Weber, 2004, p. 48)

With the development of 386BSD by Bill Jolitz and Linux by Linus Torvalds, the "open" source movement gained momentum and became an alternative to "closed" software development. "Closed" and "open" software development provided two different software development paradigms: (a) the cathedral and (b) the bazaar. While the cathedral focuses on a centralized, tightly organized approach, the bazaar comprises a multitude of "agendas and approaches" out of which a stable system arises (Raymond, 1999, p. 3).

As open-source software development expanded, a multitude of software solutions began to appear on the market. This list included both simple software systems, as well as large, complex systems such as Linux, Apache, Mozilla, Android, etc. (Mockus, Fielding, & Herbsleb, 2002; Weber, 2004).

Open-Source Frameworks

Open-source frameworks are part of the open-source movement and, as such, share some of the main characteristics of open-source software: public source code, distributed development, a flexible development system, etc. (Srinivasan, 1999; van Gurp & Bosch, 2001). As frameworks, they also have similar characteristics and benefits to their commercial counterparts such as code reuse and software development optimization (Polančič et al., 2011; Polančič et al., 2010). Moreover, to add functionality, users are able to modify it, and share their contributions with the framework community, thus becoming framework developers themselves (Polančič et al., 2011; Polančič et al., 2010). These potential improvements in productivity and quality make framework-based development one of the most promising approaches for improving application development (Frakes & Kang, 2005; Polančič et al., 2011).

While the actual number of frameworks, open-source frameworks, and the projects for which they have been used are not available, previous studies (e.g., Manolescu, Noble, & Voelter, 2006; Polančič et al., 2011; Polančič et al., 2010; Polančič et al., 2009) have suggested that they are extensively used. One of the most respected and widely accepted online project and code repositories is SourceForge (2011). In their study of open-source framework usage, Polančič et al. (2011) found that in 2007, SourceForge contained over 5,000 framework projects and over 10,000 active users were contributing to these projects. Open-source frameworks are also widely used outside of SourceForge. For example, Drupal is a web development framework that is used and maintained by over half a million users (Drupal, 2011). Analyzing such repositories shows that a variety of frameworks are available, most tailored for specific platforms.

Framework Taxonomy

Frameworks come in a variety of types, offering different functionality based on their intended purpose and platform (van Gurp & Bosch, 2001). Van Gurp and Bosch (2001) attributed one of the first framework classifications to Taligent. They used Taligent's initial classification to group frameworks based on their intended use as: (a) application frameworks; (b) domain frameworks; and (c) support frameworks.

Application frameworks "aim to provide a full range of functionality typically needed in an application. This functionality usually involves things like a GUI, documents, databases, etc." (van Gurp & Bosch, 2001, p. 278). Because of increasingly complex application requirements and the fact that application frameworks aim to provide a full range of functionality for the entire application, these types of frameworks have evolved considerably and their numbers have increased substantially (Polančič et al., 2011). Most of the open-source frameworks used for web development, such as Drupal, Wordpress, etc., are application frameworks.

Domain frameworks are aimed at specific domains such as banking or alarm systems (van Gurp & Bosch, 2001), while support frameworks "typically address very specific, computer related domains such as memory management or file systems ... and are typically used in conjunction with domain and/or application frameworks" (van Gurp & Bosch, 2001, p. 278). Because these latter two types of frameworks are not as numerous or widely used as application frameworks, we will focus our research only on application frameworks.

Given the evolution of technology and the requirements of new applications (Polančič et al., 2011), frameworks are now being used to develop applications for different platforms. For example, since mobile devices have different processing power, screen resolution, or resource constraints than PCs, developers must take these factors into consideration when selecting the best framework to use. Although some frameworks support multiple platforms, most support only one. Thus, the classification of application frameworks can be extended to include the platform(s) for which they are used.

Therefore, we have further divided application frameworks, including open-source application frameworks, into: (a) general frameworks, which provide functionality for several platforms; and (2) specialized frameworks, which limit creation of applications to a single platform. These specialized frameworks can then be subdivided into PC-based frameworks, web-based frameworks and mobile-based frameworks. For example, web development frameworks include Ruby on Rails, Django, CakePHP, Zend, Drupal and many others. This classification is presented in Figure 1. We intend to use this classification to determine whether there are significant differences in user requirements between the different types of application frameworks.

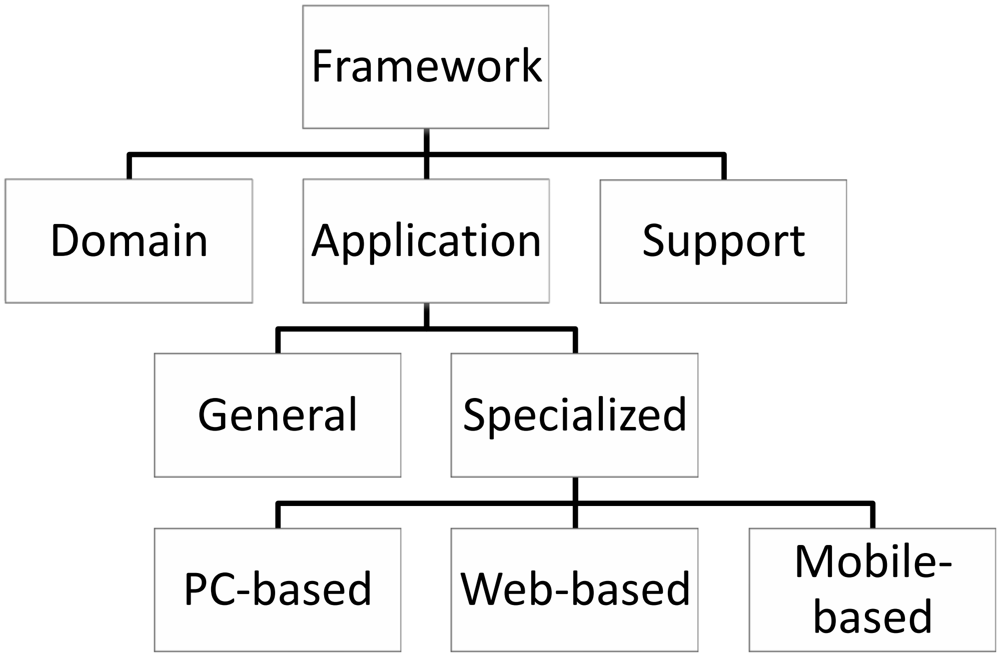

Figure 1. Framework Classification.

Framework Components

At its simplest level, a framework is a collection of classes that can communicate with each other and with other classes through an Application Programming Interface (API). The API can also be used to create new functions or classes or to display information (van Gurp & Bosch, 2001). The elements of a framework are presented in

Table 1. A framework doesn't have to include all of these elements and could consist of only one class, but it would not be very useful in today's environment.

Table 1. Elements of a Framework [adapted from van Gurp and Bosch (2001)].

Order (of Development)	**Name of the Component**	**Description**
1	Design Documents	Contain the design of the framework. May be modelled through UML (class diagrams)
2	Interfaces	Describe how each class can be used or can interact with other classes or objects
3	Abstract Classes	"an incomplete implementation of one or more interfaces" (van Gurp & Bosch, 2001, p. 288)
4	Components	A part of a class or a set of several classes that has an API, a specific function and explicit dependencies
5	(Concrete) Classes	A set of functions that are usually part of a component and are not directly used by the application developer through the API

Based on the elements presented in Table 1, application frameworks can be classified into whitebox and blackbox frameworks (van Gurp & Bosch, 2001). Whitebox frameworks usually consist only of interfaces and abstract classes. Thus, for application developers (i.e., framework users) to actually use these classes, they must extend them and create their own concrete classes (van Gurp & Bosch, 2001). These classes allow for maximum flexibility since the software developer can create concrete classes that are specifically tailored to the requirements of the application. In contrast, blackbox frameworks also contain components and concrete classes. Therefore, the application developer only has to configure the classes based on the requirements of the application and then instantiate them (van Gurp & Bosch, 2001). This reduces the application development time. In either case, the developer can choose to extend the existing classes if the framework doesn't meet all the requirements of the application.

Factors Affecting Framework Usage

Table 2 provides an overview of the factors that have been identified by previous studies as potentially affecting a user's intention to continue using a framework.

Table 2. Previous Framework and Related Studies.

No	Topic	Author	Study Summary	Main Findings
1	Software Development	Sindre, Conradi, and Karlsson (1995)	Presents Reuse Based on Object-Oriented Techniques (REBOOT).	REBOOT can be used for improving software reuse and productivity. The four most important characteristics of software development with reuse are flexibility, portability, understandability and confidence.
2	Software Development	Frakes and Kang (2005)	Analyzes software reuse research.	While assessing the advantages of software reuse, several issues, such as increased complexity, scalability and design issues should be analyzed Emphasis is put on the flexibility and efficiency of the framework. Better documentation and design practices are required to reduce complexity and increase usefulness.

Table 2. Previous Framework and Related Studies.

No	Topic	Author	Study Summary	Main Findings
3	Framework Development	Srinivasan (1999)	Analyzes the development process of a framework for speech recognition.	Frameworks offer advantages in terms of code reuse, flexibility and evolution through reuse. Documentation and design are paramount in developing a successful framework. Role separation of the modules, flexibility and efficiency of the framework are also key factors.
4	Framework Development	Batory, Cardone, and Smaragdakis (2000)	Analyzes the impact of reusable and instance specific code in a framework.	By decomposing frameworks and framework instances into small components, code replication problems are alleviated. This allows for increased productivity and usefulness of frameworks.
5	Framework Development	Bosch, Molin, Mattsson, and Bengtsson (2000)	Analyzes a number of framework development, usage, composition and maintenance problems.	Framework development problems usually arise from the lack of business models and framework testing. Framework usage problems usually arise when the user of the framework does not understand the applicability of the framework and underestimated development time of the application. Framework composition problems usually arise from different architectures used in each framework and possible control flow collisions. Framework maintenance problems usually arise as frameworks evolve over time.

Table 2. Previous Framework and Related Studies.

No	Topic	Author	Study Summary	Main Findings
6	Framework Development	Van Gurp and Bosch (2001)	Creates a conceptual model and a set of guidelines for developing and using frameworks.	Frameworks offer advantages in terms of flexibility, usability and reusability. There are several potential issues with frameworks that can be addressed by following the guidelines (see Table 3). Increasing flexibility and efficiency will increase usefulness, but increased complexity may affect ease of use.
7	Framework Development	Manolescu, Noble, and Voelter (2006)	Presents a collection of patterns for designing frameworks for software reuse.	Successful frameworks usually have a clear focus, are easy to use, avoid being overly complex, are flexible and efficient.
8	Framework Usage	Polančič, Horvat, and Rozman (2009)	Identifies the main characteristics of frameworks that can influence a user's perceptions of them.	Understandability, adaptability and confidence are the main characteristics that influence ease of use and usefulness.
9	Framework Usage	Polančič, Heričko, and Rozman (2010)	Examines the factors that influence a framework's acceptance and success.	Successful use of a framework depends on the intention to continue using a framework and the perceived usefulness of the framework.
10	Framework Usage	Polančič et al. (2011)	Identifies the main characteristics of frameworks.	Framework characteristics and individual differences have a significant impact on user's perceptions of frameworks.

Table 2. Previous Framework and Related Studies.

No	Topic	Author	Study Summary	Main Findings
			Together with individual differences, framework characteristics affect the user's perceptions of the usefulness and ease of use of the framework.	The study provides a conceptual model for evaluating frameworks and for analyzing the antecedents that affect the user's perceptions of frameworks. While the impact of these antecedents on perceived usefulness and ease of use is analyzed, the impacts of perceived usefulness and ease of use on continued usage intention are hypothesized, but not analyzed.

Based on the software reuse literature (Frakes & Kang, 2005; Sindre et al., 1995) and software development guidelines for increasing software productivity (Boehm, 1999), frameworks incorporate most of the key ideas, and thus should increase productivity, efficiency, and the quality of applications (Manolescu et al., 2006; Polančič et al., 2011; van Gurp & Bosch, 2001). While users still face some challenges utilizing frameworks, several researchers (e.g., Manolescu et al., 2006; van Gurp & Bosch, 2001) have focused on creating guidelines for framework development to help alleviate these challenges. These guidelines are presented in Table 3 and will be discussed later in this section.

One of the major building blocks of our study is the research conducted by Polančič et al. (2011). The authors analyzed the impact of technological characteristics of the framework and individual differences among framework users on the user's intention to continue using a framework. They found that both technological characteristics and individual differences affect the user's perception of the framework's usefulness and ease of use.

Another important building block is the REBOOT model. It identifies four important characteristics for successful software development through reuse, and, by extension, for open-source frameworks: flexibility, portability, understandability and confidence (Sindre et al., 1995). Based on the factors identified by Polančič et al. (2011) and Sindre et al. (1995), as well as those from other studies of software development, reuse, and frameworks, the following factors are likely to affect a user's intention to continue using a framework.

Confidence. According to REBOOT, confidence can be defined as "the (subjective) probability that a module, program or system performs its defined purpose satisfactorily (without failure) over a period of time" (Sindre et al., 1995, p. 207). Confidence in the framework is essential, because the user will use a framework with a variety of applications. A failure of the framework may affect all of the applications that were developed using it.

Failures can range from an error in displaying an interface to a failure in the core components of the framework. These failures can occur because of conflicts between different elements of the framework or because of evolution problems. While in commercial frameworks these failures have to be addressed by the developers of the framework, in open-source frameworks the community will play an active role in fixing any issues that may arise. However, this also makes these failures known to all the users of the framework and, depending on the frequency and severity of the failures, may cause users to discontinue their usage of the framework.

In their revised model, Polančič et al. (2011) found that confidence was directly influenced by task-technology fit and understandability. They suggested that confidence, perceived usefulness and perceived ease of use are the three main factors that influence continued framework usage intention.

Portability and flexibility. Portability refers to the "ease with which the component can be transferred from one computer system or environment to another" (Sindre et al., 1995, p. 207). Therefore, portability should increase the usefulness of the framework, but may also increase its complexity and thereby reduce ease of use.

Flexibility, or adaptability, can be defined as "the ease with which the component can be adapted or modified to different functional needs, or used in different contexts" (Sindre et al., 1995, p. 207). Similar to portability, flexibility should increase the usefulness of the framework, but may also increase complexity and therefore reduce ease of use.

Portability has been operationalized differently in framework studies; while REBOOT defines flexibility and portability as two independent factors, Polančič et al. (2011) combined them into one measure. However, after analyzing their results, they recommended that flexibility and portability should be treated as distinct constructs and measured separately. Thus, we define flexibility as the ability to use a framework across different applications on the same platform, while defining portability as the ability of a framework to be used across different platforms.

Understandability. Understandability, the last of the four constructs identified by REBOOT to affect software quality, is defined by Sindre et al. (1995) as:

> A software product is understandable to the extent that its purpose is clear to the prospective reuser. If the component can be reused as is, understandability is only essential for its interface. If it has to be modified, it is also important that the implementation is understandable. (p. 207)

This means that if a framework is hard for a user to understand, then the user's perception of the ease of use of the framework will be negatively affected. Additionally, most of the core components of the framework can be expected to be reused "as is" (van Gurp & Bosch, 2001), thus making understandability of the interface highly important. Previous studies (Polančič et al., 2011) have found that understandability has a significant impact on the user's perception of the ease of use of the framework and on the confidence of the user in the framework.

Polančič et al. (2011) measured understandability based on whether the framework is self-descriptive or easy to learn. Given the fact that frameworks are complex systems, the framework's understandability can make it easier to use. However, this does not mean that the framework is easy to learn. Moreover, the ability for a user to understand the framework is closely related to the quality of its documentation, which should provide a clear explanation of the framework's functionality and of how it can be extended.

Efficiency. Efficiency represents "the capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions" (Polančič et al., 2011, p. 3). Given the increased complexity of frameworks due to the requirements of flexibility, portability and software reuse, previous studies

(Polančič et al., 2011; Srinivasan, 1999) have suggested that efficiency of the framework should be one of the top priorities of developers. Redundant code, bad design and monolithic components not only affect efficiency, but may also reduce the perceived usefulness of the framework.

While Polančič et al. (2011) found the impact of efficiency on perceived usefulness to be non-significant, this finding may be related to the wording of their instrument. First, they used a reverse-coded item, EF1: "The framework requires too much of system resources" (Polančič et al., 2011, p. 5), which can cause confusion for respondents. Second, their item for EF2 contains a double-barreled question involving response times and processing times. These should be analyzed separately and not in one question.

Framework suitability. Polančič et al. (2011) created a new Task-Technology Fit (TTF) construct for their study, defined as "the matching of technological capabilities with the demands of individuals. TTF posits that IT will be used if, and only if, the functions available to the user support the activities of the user" (Polančič et al., 2011, p. 3). Thus, the focus of TTF is on matching technological capabilities with the demands of the task versus the demands of the individuals.

Polančič et al. (2011) created this new construct based on Task-Technology Fit (TTF) theory (Goodhue & Thompson, 1995). This theory postulates that individual performance will be influenced positively if the capabilities of the system match the task requirements. Along with TAM, TTF has been one of the most important models that has been used in the IS literature for understanding the intention to use and the actual usage of a system (Dishaw & Strong, 1999). Dishaw and Strong (1999) conducted a study to analyze whether the two models could be used together to improve the explanatory power of each of the base models. They found that the new integrated model had better explanatory power and recommended that future studies use TTF constructs to better understand IT system usage (Dishaw & Strong, 1999).

Klopping & McKinney (2004) found that, in e-commerce, TTF has a significant impact on perceived usefulness. Similarly, Polančič et al. (2011) also found that TTF has a positive impact on perceived usefulness and on the user's confidence in the framework. Therefore, we decided to include this construct in our study. However, because of the differences between this new TTF construct and the original one, we have decided to label this construct as framework suitability. This is consistent with Polančič et al. (2011), who also referred to their TTF construct as suitability of the framework.

Framework technical quality. To better understand the use of frameworks, we need to examine some of the framework-specific challenges developers face when using these tools. Van Gurp and Bosch (2001) identified several issues that affect the flexibility and reusability of frameworks and outlined some possible solutions. They grouped the identified issues into two categories, (a) Composition Problems[1] and (b) Evolution Problems.

Evolution Problems arise from the process of upgrading and altering the

[1] Composition Problems arise when two or more frameworks are used in one application. However, open-source frameworks are built on a modular architecture and allow users to change the way they can communicate with other software applications. Therefore, we do not expect composition problems to have a significant impact on open-source frameworks and their users.

framework. As the framework evolves to meet new requirements, these changes may make the new version incompatible with older instances. This requires the software developer to update all the older applications that make use of the framework and, possibly, to reprogram parts of them. Since open-source frameworks are constantly upgraded and modified by users in the community, this can be a significant source of implementation challenges. One possible solution to this problem is to leave existing API calls unchanged and only add new functions or module calls (van Gurp & Bosch, 2001).

Based on the issues identified in their research, van Gurp and Bosch (2001) created guidelines for developing better frameworks, or for extending existing ones. Their findings are presented in Table 3.

Table 3. Framework Usage Guidelines and Implementation Challenges [adapted from van Gurp and Bosch (2001)].

No	Guideline Name	Guideline Description	Impact for Users
1	Interface Component Separation	The interface of a component should be separate from its implementation	If the interface is not separate from the component API, then users are going to face difficulties when trying to change or create new interfaces that use that component. Moreover, changes to the component may create unwanted change in the interfaces. A framework that follows this guideline is expected to have increased usefulness, but reduced ease of use.
2	Interfaces should be role oriented	Often, only a part of the API with a specific role is required. These are usually used in an interface. Unnecessary dependencies are created when an interface has	Interfaces that are not role oriented are likely to have additional dependencies that are not used or required by a particular role. This creates additional overhead and maintenance costs when updating or creating new interfaces.

Table 3. Framework Usage Guidelines and Implementation Challenges [adapted from van Gurp and Bosch (2001)].

No	Guideline Name	Guideline Description	Impact for Users
		more than one role.	A framework that follows this guideline is expected to have increased usefulness, but reduced ease of use.
3	Role inheritance	The use of several roles from a component may be required. Since all other interfaces are excluded when referencing a particular one, this conflicts with the previous guidelines.	In situations where different roles are required, role inheritance is the best solution to pass down access rights or other role information, rather than creating new rules. Therefore, instead of creating entirely new permissions for a moderator role, the role should inherit the permissions from a contributor and editor role. A framework that follows this guideline is expected to have increased usefulness, but reduced ease of use.
4	Prefer loose coupling over delegation	Dependencies between classes and components are one of the major problems in using frameworks. A dependency, or delegation, is created when a component requires specific information or functions from another component.	Since the use of delegation requires specific functions from child components, this can negatively affect users who are trying to develop new components or interfaces. For example, framework users who wish to develop new components will first have to create functionality for missing dependencies. A framework that follows this guideline is expected to have increased usefulness, but reduced ease of use.
5	Use small components	Since components can be considered the building blocks of frameworks, using large components has a detrimental impact	The use of large components makes it difficult for users to understand and extend the functionality of the framework. First, it is not feasible to reuse only a small part of a large

Table 3. Framework Usage Guidelines and Implementation Challenges [adapted from van Gurp and Bosch (2001)].

No	Guideline Name	Guideline Description	Impact for Users
		on flexibility and reusability.	component. Second, it is difficult to break apart a large component into smaller components. Third, inheritance in large components reduces the ease of use of the framework. A framework that follows the guideline is expected to have increased usefulness, but reduced ease of use.
6	Use standard technology	Sometimes developers do not trust components or modules that are developed outside their company or are not aware that these solutions exist. This leads to a "reinvention of the wheel" situation.	The use of non-standard technology and architectures makes it harder for the user to understand and implement the framework and therefore reduces the ease of use and usefulness of the framework.
7	Automated configuration	With increased flexibility and functionality, configuring the framework becomes a more complex process. Automated configuration can help this issue by providing default settings for the application, or by recommending settings based on the type of application developed.	If the user is required to manually configure every part of the framework, instead of accepting generated recommendations for general parts of the framework, this reduces the ease of use and usefulness of the framework.
8	Documentation	Increased flexibility and complexity of the framework requires detailed documentation so that a developer can understand how to use the framework.	Proper documentation of the framework and its API is essential for users to understand how to use the framework. As with automating the configuration of the framework,

Table 3. Framework Usage Guidelines and Implementation Challenges [adapted from van Gurp and Bosch (2001)].

No	Guideline Name	Guideline Description	Impact for Users
			tools can be used to automate the documentation process. For example, a tool that documents all API inputs and outputs would help write the documentation faster and will also be of tremendous help to developers. If the documentation is missing, or if there's no method available for documenting the functions of the API, then this reduces the ease of use and usefulness of the framework.

While these guidelines are supposed to increase the flexibility, portability, efficiency and general usefulness of the framework, they can also increase the complexity of the framework and reduce its ease of use. For example, if you have a complex framework that adheres to all the above guidelines, you will have a multitude of small components that will require a complex inheritance pattern on several levels. Moreover, separating the interfaces from the components requires additional programming, which, in turn, reduces ease of use.

While van Gurp and Bosch (2001) prepared these guidelines for framework developers, these guidelines can also be used as quality characteristics of frameworks. Each guideline deals with a different part of the framework and is supposed to increase overall quality. Moreover, these framework technical characteristics can then be evaluated by framework users to get an overall index of the quality of a framework.

Therefore, framework developers must balance the gains from following these guidelines with the reduced ease of use of the framework. This is especially true, since the challenges users face while utilizing the framework will affect their intention to continue using the framework, which, in turn, will affect the success of the framework.

Debugging. Debugging an application and testing the code for errors usually plays an important role in the development of traditional software. According to Hailpern and Santhanam (2002), around 50% of traditional software development costs represent debugging and testing. However, in frameworks, the tasks of testing and debugging the framework are made easier by the modular nature of the framework and by the fact that frameworks allow users to enable or disable their modules selectively (Bosch et al., 2000; van Gurp & Bosch, 2001). Moreover, most frameworks have some type of debugging tools built inside them (Manolescu et al., 2006). Since the debugging tool is internal to the framework, it technically becomes one of its modules, or components. Therefore, the capabilities of the debugging module are part of the framework's technical quality.

Scalability. The scalability of an application in general usually refers to its ability to be expanded or downsized to fit specific application requirements (Frakes & Kang, 2005). Since frameworks are modular and allow their users to customize their functionality, we expect them to be inherently scalable. However, some frameworks can be more scalable than others. For example, frameworks that have clearly defined and separated roles and interfaces should be easier to expand, or scale up, for use in larger applications, compared to frameworks that use a large, monolithic design. Thus, the scalability of a framework is part of the framework's technical quality.

Social pressure. Apart from the characteristics of the framework and the implementation challenges faced by the users of the frameworks, several studies on methodologies and software development (e.g., Hardgrave, Davis, & Riemenschneider, 2003; Lee, 2010) have identified the importance of social factors on the user's intention to continue using a software application. Coupled with the importance of the community in open-source software (Weber, 2004), social factors could to play a significant role in open-source frameworks.

Social pressure, also referred to as subjective norm, comes from the Theory of Reasoned Action (TRA) and refers to the "perceived social pressure to perform or not to perform the behavior" (Ajzen, 1991, p. 188). Although the original TAM did not include this construct, later studies showed it has a significant impact on the intention to use technology (Schepers & Wetzels, 2007). In their study of the factors that influence software developers to follow methodologies, Hardgrave et al. (2003) defined social pressure as "the extent to which a developer experiences interpersonal influence from important others within his or her social milieu" (Hardgrave et al., 2003, p. 128). The authors found that subjective norm has a significant impact on the developer's intention to follow a methodology. Additionally, Lee (2010) stated that "subjective norm is related to the normative beliefs about the expectation from other people. Many Internet users choose to use e-learning because their friends are the users of e-learning system, and they recommend it to them" (Lee, 2010, p. 508).

However, unlike previous studies on social pressure (e.g., Hardgrave et al., 2003), the open-source community is composed of individuals who the user has not met face-to-face. Nevertheless, individuals can be influenced by other users who are important to them, or who are important in the online community, via the internalization effect. Therefore, the intention to continue using a framework could be influenced by social pressure.

Implementation gap. Polančič et al. (2011) suggested that the implementation gap can influence the perceived usefulness and perceived ease of use of the framework. According to the authors, this refers to the gap between new and old technologies; as this gap becomes wider, framework users require more time to learn and to adapt to the framework. Polančič et al. (2011) based this construct on the conceptualization of Chau (1996, p. 272), who proposed that using a new technology requires "new skills and new knowledge." Therefore, the implementation gap contains two components: the user's ability to learn and understand the framework and the challenges faced when using it.

However, REBOOT (Sindre et al., 1995) already contains the construct of understandability, and the framework suitability construct already includes the matching of technological capabilities with the demands of the task versus the demands of the individuals (Polančič et al., 2011). Moreover, the existing literature on frameworks (Srinivasan, 1999; van Gurp & Bosch, 2001) details a number of implementation challenges software developers face when using a framework. These challenges are included in the operationalization of the framework technical quality construct. Thus, implementation gap has a significant overlap with the previously discussed factors.

Research Opportunities

While frameworks have been presented as one of the most promising solutions for improving productivity through software reuse, application developers face a number of challenges when utilizing these frameworks. These challenges are mostly specific to the use of frameworks and software reuse and, without solutions to overcome them, may negatively impact the user's intention to continue using a particular framework and, by extension, the framework's success.

To date, most research has been conducted on frameworks themselves, with framework users receiving less attention. From the user's perspective, frameworks offer the possibility of increasing both productivity and software quality. However, for any system to be successful, it has to be accepted and utilized. In their 2010 study, Polančič et al. (2010) found that the framework's usefulness and its ease of use have a positive impact on the continuous framework usage intention. In turn, this has a positive effect on the IS success measures of net benefits and satisfaction. The authors then suggest that the antecedents affecting continuous usage intention should be analyzed. While Polančič et al. (2011) provided an initial analysis of these antecedents, their study only measured the impact of their suggested antecedents on perceived usefulness and perceived ease of use, and not on continuous usage intention. Moreover, in their revised model, the authors theorized that the construct of confidence should have a direct impact on continuous framework usage intention, but did not test this hypothesis.

Based on the results of the literature review, we identified five gaps in the literature that we intend to address in our study. The first gap relates to the lack of understanding of how framework and open-source framework technical factors such as flexibility, portability, efficiency, understandability, and framework suitability impact perceived usefulness, ease of use, and continued framework usage intention. Polančič et al. (2010) found that perceived usefulness and continued usage intention determine system use, net benefits, and satisfaction, which are the three main dimensions of IS success as defined in the updated DeLone & McLean IS success model (2003). An analysis of the antecedents of these factors could contribute to both the existing literature and the framework developers' and IT managers’ need to understand factors affecting framework utility and usage intention.

Second, Benbasat & Barki (2007) suggested using relevant constructs from other theories to improve the explanatory model of existing models. As suggested by other studies on methodologies and software development (e.g., Hardgrave et al., 2003; Lee, 2010), social pressure may affect the intention of developers to continue using a framework. While the impact of social pressure has not been analyzed in the framework literature, we expect it to have a significant influence on the user's decision to continue using a framework. We expect this to be particularly important in the case of open-source frameworks, where developers are part of and interact with the open-source community, most of whom they have never met face-to-face.

Third, several studies on frameworks suggest a number of guidelines to improve the quality of frameworks (van Gurp & Bosch, 2001). However, there is a lack of studies that look at these guidelines and analyze the impact of the framework's technical quality on the user’s intention to continue using the framework. Therefore, we will create a framework technical quality construct based on these guidelines, allowing us to analyze the impact of the framework technical quality on both perceived usefulness and perceived ease of use.

Fourth, the TAM model has been used mostly in pre-adoption scenarios (King & He, 2006). By analyzing the user's perceptions about a framework that they are already using, we are utilizing TAM in a post-adoption scenario and therefore will contribute to the literature on post-adoption usage models (e.g., Hong, Thong, & Tam, 2006). This contribution will be discussed in more detail in the following section.

Fifth, our analysis of the previous literature on framework research discovered a number of conceptualization and operationalization issues that we intend to address. For example, the constructs of portability and flexibility were operationalized as a single construct by Polančič et al. (2011). Based on the discussion in the current chapter and results of Polančič's et al. (2011) study, we believe these two constructs to be distinct. Therefore, they should be operationalized and analyzed separately.

The construct of Task Technology Fit (TTF), as used by Polančič et al. (2011), is different from the original TTF construct that was developed by Goodhue and Thompson (1995). The original TTF construct theorizes that individual performance will be influenced positively if the capabilities of the system match the task requirements. Thus, the original TTF construct suggests that performance is a characteristic of the framework and task, not a characteristic of the individual. Polančič et al. (2011) adapted this construct to include the user: "Task-Technology Fit (TTF) implies the matching of technological capabilities with the demands of individuals. TTF posits that IT will be used if, and only if, the functions available to the user support the activities of the user" (Polančič et al., 2011, p. 3). Since previous studies on frameworks (e.g., Srinivasan, 1999; van Gurp & Bosch, 2001) suggest that the user’s individual characteristics play a significant role on performance, we intend to follow the conceptualization of Polančič et al. (2011). To avoid ambiguity, we will name this construct as framework suitability. Polančič et al. (2011) also referred to their TTF construct as the suitability of the framework.

In summary, because the developer's intention to continue using the candidate framework is crucial for the success of the framework and because there are significant gaps in the literature identifying the factors that influence this decision, we intend to better understand these factors. This research is important because it is the first step in creating an objective measure for the quality and success of a framework. Moreover, our research can help framework creators understand what factors influence their users to continue using their product and, in turn, improve the quality of frameworks and increase their success. Moreover, IT managers can use this research to improve productivity with frameworks by better understanding their employees' requirements.

Chapter 3: Theoretical Model and Hypotheses Development

Theoretical Model

This chapter presents the theoretical models that have been used in previous studies of software use, as well as the proposed research model. In particular, we will focus on the research conducted by Hong, Thong, and Tam (2006) on three post-adoption models that included continued usage intention as a dependent variable: technology acceptance model (TAM), expectation-confirmation model in IT domain (ECM-IT) and extended ECM-IT.

TAM. The technology acceptance model theorizes that perceived usefulness and perceived ease of use determine an individual's intention to use a technology (Davis, 1986). TAM extends the theory of reasoned action (TRA), which was proposed by Fishbein and Ajzen (1975). TRA hypothesizes that if a person intends to engage in a certain behavior, then that person is likely to actually do so. The intentions of the person are influenced by two factors: the person’s attitude toward the behavior and the subjective norm. Three main dimensions can be attributed to the TRA: behavioral intention (BI), attitude (A), and subjective norm (SN).

TAM was heavily influenced by TRA theory and extends several of its dimensions. BI became behavioral intention to use, A became attitude toward use and two new constructs, perceived ease of use (PEU) and perceived usefulness (PU) were added. PEU and PU are hypothesized to be the primary predictors of BI.

Since its initial proposal, TAM has become widely used and many models have attempted to extend it (King & He, 2006). One of these models, developed by Venkatesh and Davis (2000), is commonly referred to as TAM2. This model proposes a number of external variables that are hypothesized to affect perceived usefulness and intention to use. One of the most important of these is the social component from TRA, labeled as subjective norm (Schepers & Wetzels, 2007). This extension of the model allows us to better understand how a person's perception about a technology can change based on the social context.

ECM-IT. Bhattacherjee (2001) created the expectation-confirmation model in IT domain (ECM-IT) to analyze a user’s intention to continue using an information system. This model builds on expectation-disconfirmation theory and helps explain the differences between acceptance and continuance behaviors (Bhattacherjee, 2001). The model hypothesizes that a user develops a set of expectations when they encounter a new IT system. After using the system, if the positive expectations of the user are confirmed, the perceived usefulness and the satisfaction of the user with the system increase. In turn, this leads to continued usage intention by the user.

Extended ECM-IT. The extended expectation-confirmation model in the IT domain was developed by Hong et al. (2006). This model combines the TAM and ECM-IT models into a "hybrid model with enhanced predictive power by incorporating their different aspects of user perceptions in the original frameworks" (Hong et al., 2006, p. 1823). However, the increased number of constructs and relations among them comes at the cost of model parsimony.

Our research model. When comparing TAM with ECM-IT and EECM-IT, Hong et al. (2006, p. 1819) concluded that "TAM is the most parsimonious and generic model that can be used to study both initial and continued IT adoption." Furthermore, TAM meta-analyses (e.g., Benbasat & Barki, 2007; King & He, 2006) have concluded that TAM is one of the most cited and validated theoretical models for examining acceptance and use of information technologies. Therefore, the TAM model was chosen as the basis of our theoretical model. Moreover, this model was also used by Polančič et al. (2011) and, by using it, we will be able to better compare our results with the results from their study.

The TAM model serves only as our starting point; we are proposing several extensions to the model. In their meta-analysis of TAM, King and He (2006) classified the modifications that have been made to the original TAM into four types (Figure 2). Type 1 modifications represent prior factors, or antecedents, that are hypothesized to affect the original TAM constructs. Type 2 modifications include factors suggested by other theories, while type 3 includes contextual factors, such as gender, culture, etc., that may have moderating effects. Type 4 modifications represent constructs that measure the consequence of the behavioral intention, such as attitudes, perceptual usage and actual usage (King & He, 2006).

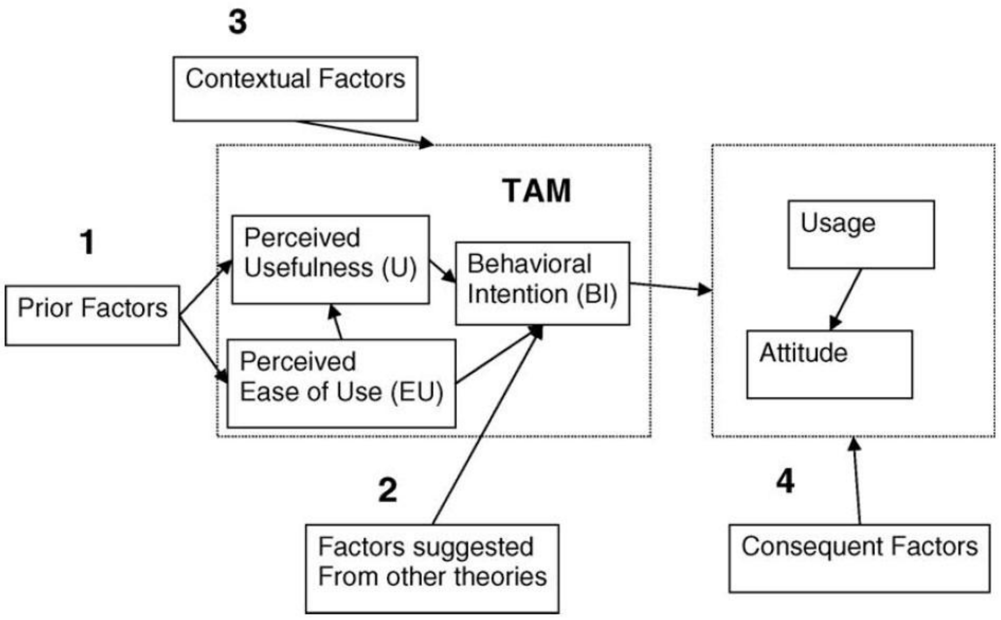

Figure 2. Technology Acceptance Model and its Modifications (King & He, 2006).

Based on this taxonomy, our contribution will consist of type 1 and 2 modifications to TAM. By making these changes to the original TAM, our research not only contributes to the field of framework research, but also to the TAM research stream.

Our inclusion of the antecedents of perceived usefulness and perceived ease of use represents a type 1 modification to TAM. As recommended by Benbasat and Barki (2007, p. 215), "we need to identify the antecedents of the beliefs contained in adoption models in order to benefit practice."

A type 2 modification to TAM (King & He, 2006) is the inclusion of social pressure. An important aspect of framework usage and development is the interaction and communication among different open-source framework users, their framework's community and the general open-source community. Application frameworks evolve and grow organically as users bring extensions to the core classes of the framework. As suggested by the internalization effect, the interaction and opinions of other important users can be interpreted by the individual as evidence about reality (Schepers & Wetzels, 2007). Since the framework user can also be its developer and since the community can be formed of people with whom the user has not interacted face-to-face, our social component is somewhat different from that used in previous studies.

Another type 2 modification to TAM is the inclusion of the confidence construct. As discovered by Polančič et al. (2011), the user's confidence in the framework fits better in the model as a determinant of the intention to continue using a framework rather than an antecedent of perceived usefulness and ease of use. As with social pressure, this not only helps us better understand the factors that influence the developer to continue using a framework, but it also brings a significant theoretical contribution to the TAM literature.

Based on the findings of the literature review (Srinivasan, 1999; van Gurp & Bosch, 2001), the results of the previous studies on frameworks (Polančič et al., 2011; Polančič et al., 2010; Polančič et al., 2009) and the discussion above, our research model is presented in Figure 3. We will discuss the hypothesized relationships shown in Figure 3 in the next section.

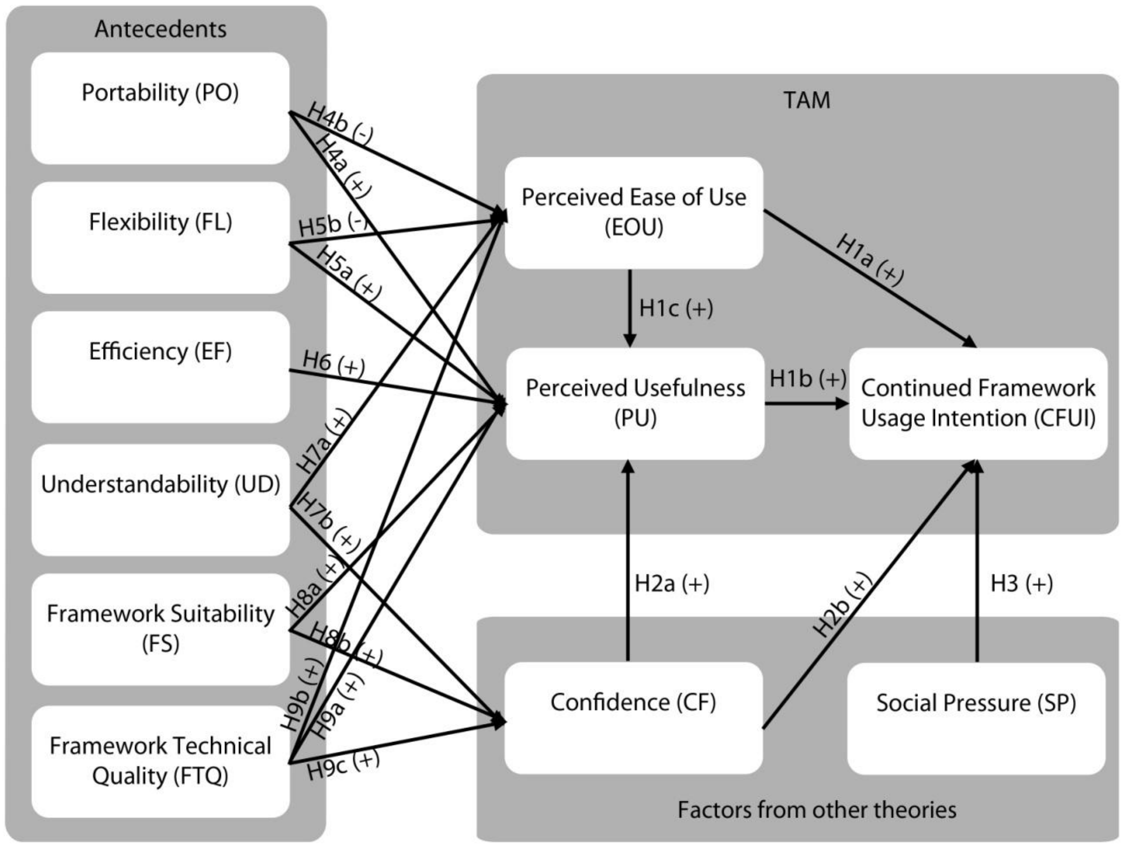

Figure 3. Research Model.

Original TAM hypotheses

The TAM model contains three constructs and three paths. According to this model (Davis, 1989) and previous framework usage studies (Polančič et al., 2011; Polančič et al., 2010; Polančič et al., 2009), the user's perception of the usefulness of the framework and the user's perception of the ease of use of the framework will influence the user's intention to continue using the framework. Moreover, perceived usefulness and the intention to continue using the framework are considered measures of the framework's success. All original TAM hypotheses, revised for frameworks and continued usage intention, are part of the H1 hypotheses. Therefore:

H1a: A user’s perception of the ease of use of the framework (EOU) will have a positive effect on the user’s continued framework usage intention (CFUI).

H1b: A user’s perception of the usefulness of the framework (PU) will have a positive effect on the user’s continued framework usage intention (CFUI).

The original TAM model also hypothesizes that as the user's perception of the ease of use of a framework increases, the user's perception of the usefulness of the framework will also increase. Therefore:

H1c: A user’s perception of the ease of use of the framework (EOU) will have a positive effect on the user’s perception of the usefulness of the framework (PU).

Extended TAM hypotheses

Confidence. As future research, Polančič et al. (2011) proposed that confidence has a significant impact on continued framework usage intention, along with perceived usefulness and perceived ease of use. However, no results have been published to confirm this. Therefore, we intend to test the impact of confidence on both the user's perception of the usefulness of the framework and continued framework usage intention.

H2a: A user’s perception of confidence in the framework (CF) will have a positive effect on the user’s perception of the usefulness of the framework (PU).

H2b: A user’s perception of confidence in the framework (CF) will have a positive effect on the user’s continued framework usage intention (CFUI).

Social pressure. Based on our previous discussion of the social component and on the user also being an application developer, we expect individuals to be influenced by other users who are important to them or who are important in the online community via the internalization effect. Therefore, the intention to continue using a framework could be influenced not only by the technological factors of the system, but also by social factors. While social pressure has not been analyzed in the open-source frameworks literature, studies on software development and methodologies (Hardgrave et al., 2003) have found that social factors have a direct influence on usage intentions. Therefore:

H3: A user's perception of social pressure (SP) will have a positive effect on the user's continued framework usage intention (CFUI).

TAM antecedents hypotheses

Based on the findings of Hong et al. (2006), Lee (2010) and Polančič et al. (2011), the user's perception of the framework should be affected by the characteristics of the framework. Framework characteristics refer to the defining traits of frameworks, which identify and differentiate them from other types of software. Based on the existing literature, we have identified characteristics which have been hypothesized to influence reusability and the user's intention to continue using the framework.

Portability. Portability will be defined as the ability of a framework to be used across different platforms or environments. Therefore, increasing the portability of the framework improves the perceived usefulness of the framework, but it is also expected to reduce its perceived ease of use (Polančič et al., 2011).

H4a: A user's perception of the portability of the framework (PO) will have a positive effect on the user's perception of the usefulness of the framework (PU).

H4b: A user’s perception of the portability of the framework (PO) will have a negative effect on the user’s perception of the ease of use of the framework (EOU).

Flexibility. Flexibility is defined as the ability to use a framework across different applications within the same platform and is expected to increase perceived usefulness, but reduce ease of use (Polančič et al., 2011). Therefore:

H5a: A user’s perception of the flexibility of the framework (FL) will have a positive effect on the user’s perception of the usefulness of the framework (PU).

H5b: A user’s perception of the flexibility of the framework (FL) will have a negative effect on the user’s perception of the ease of use of the framework (EOU).

Efficiency. Efficiency is defined as the "capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions" (Polančič et al., 2011, p. 3). Efficiency is expected to increase the perceived usefulness of the framework (Polančič et al., 2011). Therefore:

H6: A user’s perception of the efficiency of the framework (EF) will have a positive effect on the user’s perception of the usefulness of the framework (PU).

Understandability. Understandability is defined as the extent to which the purpose of a framework or component is clear to the user (Sindre et al., 1995). Given the fact that frameworks are complex systems, the user's understanding of the framework can make it easier to use. However, this does not mean that the framework is easy to learn. Moreover, the ability for a user to understand the framework is closely related to its documentation, which should provide a clear explanation of the framework's functionality and of how it can be extended. Polančič et al. (2011) also found that understandability has a significant impact on the confidence of the user in the framework. Therefore:

H7a: A user’s perception of the understandability of the framework (UD) will have a positive effect on the user’s perception of the ease of use of the framework (EOU).

H7b: A user’s perception of the understandability of the framework (UD) will have a positive effect on the user’s confidence in the framework (CF).

Framework suitability. The construct of framework suitability is defined by how well the capabilities of the framework fit the requirements of the application. This construct is derived from the Task-Technology Fit (TTF) construct that was used by Polančič et al. (2011). Previous studies have found a significant impact of TTF on perceived usefulness (Klopping & McKinney, 2004) and on the user's confidence in the framework (Polančič et al., 2011). Therefore, we hypothesize that:

H8a: A user’s perception of the framework suitability (FS) for the task is positively related to the user’s perception of the usefulness of the framework (PU).

H8b: A user’s perception of the framework suitability (FS) for the task is positively related to the user’s confidence in the framework (CF).

Framework technical quality. Based on the guidelines in Table 3, as well as the discussion on scalability and debugging, the framework's technical quality construct measures the user’s perception of various aspects of its technical quality. This construct should positively affect the user's perception of the usefulness of the framework (PU) and the user's confidence (CF) in the framework. On the other hand, increased functionality and technical quality increases the complexity of the framework (see Table 3 discussion), which in turn should negatively affect the user's perception of ease of use (PEOU). Therefore:

H9a: A user’s perception of the framework technical quality (FTQ) will have a positive effect on the user’s perception of the usefulness of the framework (PU).

H9b: A user’s perception of the framework technical quality (FTQ) will have a negative effect on the user’s perception of the ease of use of the framework (EOU).

H9c: A user’s perception of the framework technical quality (FTQ) will have a positive effect on the user’s confidence in the framework (CF).

Chapter 4: Methodology

Overview of Research Design

Based on the research questions, a cross-sectional online survey is an approriate research design. The research was conducted in two phases. In the first phase, a pre-test was used to evaluate and refine the instrument's measures. After obtaining approval from the Human Subjects Research Committee at the University of Lethbridge, the pre-test was conducted online with a small group of framework users. Based on the results of the pre-test, a number of changes were made to the survey. In the second phase, the refined online survey was sent to the randomly selected sample.

The collected data was exported into the statistical analysis package SPSS. To conduct our statistical analysis, we used both SPSS and Amos. Both tools are well suited to complete our statistical analysis and have previously been used in framework studies (Polančič et al., 2011; Polančič et al., 2010).

The following sections contain detailed information on survey construction, operationalization and measurement, the results of the pre-test, as well as information on the population, sample frame, sample selection process and the statistical analysis.

Survey Construction

Constructs overview. An previously discussed, most of the items from our instrument are based on those utilized by Polančič et al. (2011). As such, their validity has already been assessed. However, some of these items have been modified for reasons discussed in the following paragraphs. As recommended by methodology and survey design texts (Dillman, 2006; Hair, Black, Babin, & Anderson, 2009; Trochim & Donnelly, 2008), each construct contains at least three items.

The following subsections present the conceptualization, operationalization, and measurement of each construct in the study. The full questionnaire is available in Appendix C, while Appendix D contains a list of the original and modified items, as well as their sources.

Portability and flexibility. As previously discussed, Polančič et al. (2011) combined the constructs of portability (PO) and flexibility (FL). Based on the recommendations of Sindre et al. (1995) and Polančič et al. (2011), we measured these two constructs independently. According to the REBOOT definition of these constructs, portability refers to the ability to use the framework on different platforms, while flexibility refers to the ability to adapt the framework to the requirements of the application. Based on these definitions, two of the items used by Polančič et al. (2011) to measure adaptability were deemed to measure portability (PO1, PO2) and one of them to measure flexibility. The flexibility item "The framework can be easily adapted or extended to fulfill application requirements" used by Polančič et al. (2011, p. 5) is double-barreled, as adapting and extending a framework are different concepts. Therefore, we split this question into two items (FL1, FL2). One additional item was created to measure flexibility and one to measure portability.

Efficiency. Based on the discussion in the literature review, the measurement of Efficiency (EF) has also been revised slightly. First, one of the items was a reverse-coded question that could pose problems for the respondents. The item, "The framework requires too much of system resources" (Polančič et al., 2011, p. 5), was vague, negatively framed and contained awkward wording. A new item (EF1), "The framework did not require excessive system resources," was selected to replace it. This new item is not reverse-coded and utilizes clearer language. A second item, "The framework provides appropriate response and processing times" (Polančič et al., 2011, p. 5), was double-barreled, as response time and processing time are distinct concepts. While they have a certain overlap, processing time relates to how fast the framework analyzes information, whereas response time relates to how fast the framework responds to user input. Based on the recommendation of methodology and survey building texts (Dillman, 2006; Trochim & Donnelly, 2008), we split this item into two separate questions (EF2, EF3).

Understandability. The items that measure the construct of understandability (UD) have also received a number of changes. First, one of the questions asked if the framework is "easy to learn" (Polančič et al., 2011, p. 5), which is essentially different from being easy to understand. A framework can be hard to learn and still be easily understood once learned. This would be the case of a framework that has a large number of functions, modules and components, but also has good documentation, good structure, and inheritance. Since the structure and inheritance schemas are also part of good documentation, we believe that the best measure for the understandability of the framework is the understandability of its documentation. Therefore, we used the original item that referred to the understandability of the documentation (UD1, UD2) and also added two more related items (UD3, UD4).

Framework technical quality. The operationalization of the framework technical quality (FTQ) construct was developed based on the results of the literature review and the pre-test feedback obtained from framework users and developers. It represents one of our contributions to framework research. This formative construct measures the quality of different aspects of the framework. Unlike a reflective measurement model that would focus on the impact of the general quality of a framework, a formative measurement allows us to understand which specific aspects are most important to framework users. Therefore, we believe that this formative measurement model will be more useful to both researchers and practitioners than a reflective one. We will address reflective versus formative measurement models in the statistical analysis section.

Framework suitability. The framework suitability (FS) measure was developed by Polančič et al. (2011), based on Task-Technology Fit (TTF) research conducted by Goodhue & Thompson (1995). To avoid any confusion between this new TTF construct and the previous one, it is renamed as framework suitability to better represent the definition of the construct. Additionally, Polančič et al. (2011) also referred to TTF as framework suitability. The two items developed by Polančič et al. (2011) were retained (FS1, FS2) and one additional item (FS3) was added to help increase the construct's convergent validity.

Social pressure. The social pressure (SP) construct was used to identify the impact of social factors on the respondent's intention to continue using a framework. We chose to utilize the measurement of Hardgrave et al. (2003), as their study had a number of similarities to our own. Their study measured the intention of software developers to follow a software development methodology. They found that social pressure has a significant impact on the dependent variable and explains a good portion of its variance. Based on these arguments, we chose to utilize their items (SP1-3), after changing the methodology identifier, ADM, to that of the framework identifier. Additionally, one of the items (SP3) referred to coworkers. Since a framework user's counterpart of coworkers also includes people in the local and online IT communities, we created two additional items (SP4, SP5) to reflect this change.

Confidence and original TAM constructs. The items for the constructs of perceived usefulness (PU) and perceived ease of use (EOU) were originally developed by Davis (1989) and further validated by Moore and Benbasat (1991). Along with confidence (CF) and continued framework usage intention (CFUI), Polančič et al. (2011) adapted and validated these constructs and their respective measurements for framework research. The Polančič et al. (2011) items are used, with only a few modifications. First, the items that were negatively framed were changed to positively framed ones. Second, based on the recommendation of Polančič et al. (2011), only items found to have high factor loadings and high reliability indicators were used.

Descriptive and control variables. In addition to the above measures, a number of descriptive and control variables were included. These variables will be used to help us form a clear picture about our selected sample and their involvement with frameworks. As such, we gathered data about the type of application framework according to its platform (Q7 - Appendix C) and whether it is a blackbox or whitebox framework (Q10). To combat any possible comprehension challenges, we defined the terms blackbox and whitebox as part of the item wording.

We also gathered data regarding whether the respondent is using the framework voluntarily (Q9) and, in the case of mandatory use, we asked respondents to answer questions related to the intention to continue using a framework assuming that they would have the choice to stop using the framework. Additionally, we asked respondents whether they have contributed code such as modules, classes, components, interfaces or templates to the community (Q11). This should help us better understand what percentage of respondents are actively contributing to the development of the framework and whether there is a significant difference between contributors and non-contributors.

Operationalization and measurement. When operationalizing and measuring constructs and variables, researchers should consider possible threats to validity and find solutions to address them (Hair et al., 2009; Trochim & Donnelly, 2008). One of the most significant threats to the validity of a study is common method variance, or "variance that is attributable to the measurement method rather than to the constructs the measures represent" (Podsakoff, MacKenzie, Lee, & Podsakoff, 2003, p. 879). This method bias affects the validity of the conclusion of a study, as it introduces measurement error. As recommended by Podsakoff et al. (2003) we addressed this issue by utilizing different scale formats and scale anchors. For example, the framework technical quality items used a seven-point Likert scale, while the other antecedents used a five-point scale. Additionally, we tested for common method bias during our statistical analysis.

Another important concern is non-response bias. As presented by Hair et al. (2009), there are two types of non-response bias: item and unit. Item non-response bias appears when certain questions in a survey are not answered by respondents. As they suggest, we tried to combat item non-response bias by carefully designing and pre-testing the instrument. Unit non-response bias appears when randomly sampled individuals do not respond to the survey and when the answers of the respondents would differ from the possible answers of the individuals that have failed to respond (Hair et al., 2009). We tried to combat non-response bias by sending reminders to users who had not completed the survey. Additionally, we tried to identify non-response bias by comparing late responders, people who responded following the second reminder, to early responders, and performing an independent samples *t*-test. We will discuss the results of the test in the results chapter.

The survey needs to indicate whether the respondent should think about multiple frameworks, or just one particular framework when answering the survey. Moreover, should the respondent think about all applications developed with the framework, or only one application? To ensure the respondent can be reasonably expected to recall their experiences with a framework (Dillman, 2006) and to ensure that the responses to each item all relate to the same underlying framework and project, we chose to ask the respondent to think about the most recently completed application that was developed with an open-source framework (Q5). As suggested by previous studies (Srinivasan, 1999; van Gurp & Bosch, 2001), consistently utilizing a framework leads developers to become familiarized and, in turn, committed to that particular framework. We expect this commitment to increase the developer's intention to continue using that framework, even if the framework isn't particularly flexible or efficient. By focusing on the most recently used framework, we expect this commitment to have a lower impact, especially since Polančič et al. (2011) found that over 90% of framework users have used more than one framework and that 24% of the respondents have used more than 13 frameworks.

To increase generalizability, we believe it is important to include respondents who have used only one framework, or who have developed only one application, as well respondents who have used several frameworks, or who have developed several applications. Thus, we decided not to have a cutoff value for the number of frameworks used.

Pre-test. After obtaining approval from the Human Subjects Research Committee at the University of Lethbridge, the pre-test was conducted online with a small group of framework users. As suggested by survey building and research methodology texts (Dillman, 2006; Trochim & Donnelly, 2008), this is an appropriate approach when using an instrument that contains new items or items that have been modified from their original source. In our case, we used this opportunity to get feedback particularly on the framework technical quality, portability, and flexibility items.

Pre-test procedure. The pre-test began by selecting a group of application developers and framework users who were deemed qualified to provide an expert opinion. The selection criteria required developers who had experience working with at least one application framework and had completed one or more projects with their selected framework. Based on these criteria, we created a pool of suitable candidates known to us and who are active in the framework community.

We contacted participants individually to ask them to complete the survey. After each question in the survey, respondents were asked if they understood the question and were given the opportunity to provide feedback on the question. The invitation e-mail was sent to 20 individuals. After four days, an e-mail reminder was sent again to the group. After another week, the results were gathered from Qualtrics and the pre-test was deemed complete. From the twenty invitations that were sent, ten people started the survey, but only seven of them completed it. This can be largely attributed to the time commitment required to both answer the questions and provide feedback. As evidence, two individuals contacted the researcher and indicated that they didn't have sufficient time to complete the survey.

Pre-test results. The pre-test respondents provided a number of recommendations on how the survey could be improved. Apart from minor wording changes (e.g., replace "adapted" with "adapted/modified"), and adding a "not applicable" option, there were two significant changes to the survey.

The first change was not related to the actual items in the survey, but to the technical functionality of Qualtrics and its use of JavaScript (JS). JS is a client-side library which is used to make changes on the client side of a particular website. Moreover, it can also be used to transmit data from the client. In practice, JS provides clients with interactive elements on websites, such as sliders, tooltips, password strength validators, etc. Additionally, JS can also be used to track the client's activity on a website (e.g., Google Analytics tracking codes). Because of this aspect, some browsers and browser modules allow users to selectively or totally block JS on websites. These browsers and modules can be used when debugging applications, or to ensure user privacy and provide protection from several online threats. Given that framework users are application developers, and thus often debugging applications, and are also well-aware of potential security issues with JS, a number of the pre-test users were setting their browsers or modules to block or contain JS.

However, since Qualtrics requires JS to handle its operations and validations, respondents who partially block JS used by Qualtrics cannot see dynamic elements such as sliders. Moreover, respondents who fully block it cannot complete the survey. To remedy this situation, we discontinued using all interactive elements such as sliders and instead replaced them with normal text boxes. A warning was also added in the User Consent letter that JS is required by Qualtrics to complete the survey, and a general warning appeared on the first page of the survey if the respondent had JS disabled.

The second change was linked to the use of negatively framed items in the survey, especially in the framework technical quality construct. As recommended by Dillman (2006), the initial survey contained both positively and negatively framed items to improve the validity of the study and to reduce common method variance. However, several respondents indicated that the negatively framed items were hard to follow and suggested changing them to positively framed questions. Based on this feedback, we decided to change all negatively framed items to positively framed ones. The other methods to alleviate common method variance, such as different scale formats and scale anchors (Podsakoff et al., 2003), remained unchanged.

A summary of the changes made to the survey following the pre-test can be found in Table 4.

Table 4. Survey Changes Following the Pre-test.

No	Description	Original	Revised
1	JavaScript required by Qualtrics	Dynamic elements (e.g., sliders) were used to enhance functionality.	Removed dynamic elements that required JS; added warning to respondents about JS being mandatory for completing the survey.

Table 4. Survey Changes Following the Pre-test.

No	Description	Original	Revised
2	Negatively framed items	Negatively framed items were used to increase validity and alleviate common method variance.	Changed negatively framed items to positively framed ones.
3	Clarified one of the answer choices for Q8: "For what industry was the application developed?"	One of the original answer choices was "Software Development."	Changed answer choice to "Software Development (for the software industry)."
	Note: Since technically all applications require software development, the rationale is that respondents should only select this option if their application is used in software development by the software industry.		
4	Clarified Q9 to ensure that respondents answer this question based on their opinion when they started the project.	"Given the choice, would you have chosen to use this framework for this application?"	"When you started the project, would you have chosen to use this framework for this application?"
5	Clarified Q10 to specify that instantiation is not an extension of the framework.	"Is it necessary to extend the selected framework with components or interfaces to create an application with it?"	"Is it necessary to extend the selected framework with components or interfaces to create an application with it? Please note that instantiation is not considered an extension of the framework."
6	Clarified what the term "adapted" means for FL1.	"The framework was easily adapted to fulfill my application requirements."	"The framework was easily adapted/modified to fulfill my application requirements."
7	Clarified what the term "adapted" means for FL3.	"The framework was easily adapted to create applications within the same domain."	"The framework was easily adapted/modified to create applications within the same domain."

Table 4. Survey Changes Following the Pre-test.

No	Description	Original	Revised
8	Removed one FTQ item related to standard technology because it is different for each framework. Moreover, the term cannot be easily defined and applied to all frameworks.	"The framework did not use standard technology."	The item was removed.
9	Added the "Not Applicable" answer option to the fourth section, "Individual."	"Don't know"	"Don't know/Not applicable"

Sample Frame

The population of interest for this study includes all application developers who have used an open-source application framework for developing software packages and who have completed at least one application with their chosen framework. Our sample frame consists of framework users who have used an application framework to complete at least one application and are part of the SourceForge database. SourceForge is a code and project repository that contains over 300,000 open-source projects (Wikipedia, 2011). According to Polančič et al. (2011), who also used SourceForge as a sample frame for their study, SourceForge contained 5,216 framework projects as of December 2007. Combined with the fact that SourceForge is one of the largest open-source project repositories on the Internet (SourceForge, 2011), we believe that it is an appropriate sample frame for our study. Moreover, using the same sample frame as Polančič et al. (2011), but with a different random sample at a different time, allows our results to be easily compared to theirs.

Access to our sample frame was secured from the SourceForge Research Data Archive (SRDA) (Van Antwerp & Madey, 2008). SourceForge provides access to data about its users and its projects for academic and scholarly researchers via an agreement with the University of Notre Dame. Access was secured via an application and an agreement was signed with Greg Madey, the researcher in charge of this project. The database contains information about each project, including the users who have contributed to and/or used the project. In essence, each of these framework users is a member of our population.

Once access was obtained to the data, a random sample was drawn from our sample frame. To determine the minimum sample size for this study, we relied on the requirements of confirmatory factor analysis (CFA) and structural equation modeling (SEM). According to Hair, Black, Babin, and Anderson (2009), the sample size should be based on the desired power, number of constructs and items, and the population size. Since the population size is unknown to us, we will be using the general guidelines for CFA and SEM. Based on Bryant and Yarnold (1995) and MacCallum, Widaman, Zhang, and Hong (1999), a subject to variable ratio (STV) of 5 and a sample size around 200 are recommended.

In their study Polančič et al. (2011) randomly selected a sample of 4,000 framework users and obtained 447 completed surveys. Out of that number, 391 surveys (9.7% net response rate) were used for the statistical analysis. Based on their results, and to allow for a safety margin, we selected a sample of 5,000 framework users.

Our analysis was conducted on the latest "snapshot" of the database available at the time (June 2012). Based on the schema of the SRDA database (Van Antwerp & Madey, 2008), all projects and groups are classified into categories called "troves." An initial query of the database identified that framework projects have a dedicated trove category. A query of the framework trove for projects found 7,231 unique active projects that contained 13,778 users. This formed our sample frame. The sample frame was imported into Qualtrics (2012), which was then used to generate a random sample of 5,000 users. Qualtrics is a tool that allows survey building and data collection via the Internet.

Procedure

Once access to the sample frame was obtained and the random sample was selected, the online survey was finalized based on the results of the pre-test and approval of the revised survey was obtained from the Human Subject Research Committee at the University of Lethbridge.

As suggested by several methodology and survey design texts (Dillman, 2006; Trochim & Donnelly, 2008), gaining the trust of your respondents is essential for obtaining high response rates. To do this, we developed a website for this research, www.frameworkstudy.com. The website served as a portal where information about our research was accessible to both participants and the general public. This website served as a resource for participants who wanted more information about the importance of our research and also as a channel to communicate to the general public. The website also contained a link to the survey so that participants could use it to access it more easily. To determine their invitation source, one of the questions asked participants about how they accessed the survey. By creating this website with a web-development framework, we also established ourselves as framework users and tried to gain credibility with the participants of our study.

To further increase response rate, approval for organizing a draw for the survey participants was obtained from the Human Subject Research Committee at the University of Lethbridge. All individuals approached to participate in either the pre-test or the main study were eligible to be entered in a draw to receive a $500 Visa gift card. At the end of the survey, individuals were asked if they wanted to participate in the draw by providing their e-mail address. All email addresses were placed in a secure location, separate from the survey data. Once data collection was completed, the researcher and the supervisor used a random number generator to select the winning e-mail address. The winner was then contacted via email and arrangements were made for the participant to obtain the gift card.

Once the website was created and approval for the survey and draw was obtained from the Human Subject Research Committee, e-mails were sent out to all the randomly selected participants inviting them to complete our survey. Along with the invitation, participants received the researchers' contact information, the link to the research website, and the link to the survey. As per Dillman's (2006) recommendations, the invitation e-mail was kept as short as possible so that interested respondents would read it entirely. Additionally, it contained the logo of the University of Lethbridge, so that respondents could differentiate it from spam. To encourage a higher response rate, a reminder was sent to all users who did not respond within 72 hours, and another one after the first week.

Statistical Analysis

Overview. Once the data was collected, it was exported into the statistical analysis package SPSS. To conduct our statistical analysis, we used both SPSS and Amos. Both tools are well suited to complete our statistical analysis and have previously been used in framework studies (Polančič et al., 2011; Polančič et al., 2010). Once imported, the data was checked for validity, consistency, and out of range and missing variables. After that, descriptives were generated to provide a better understanding of the data. Next, the data was checked for normality, distribution, and heteroscedasticity. These initial steps provided a visual representation of the data and allowed us to check the assumptions required for our chosen statistical procedure (Hair et al., 2009; Trochim & Donnelly, 2008).

Formative versus reflective measurement models. When trying to measure a construct, there are two types of measurement models: (a) Reflective and (b) Formative (Freeze & Raschke, 2007; Hair et al., 2009). According to Freeze and Raschke (2007, p. 1482), "Reflective measures are caused by the latent construct, whereas, formative measures cause the latent construct." Therefore, formative constructs can be viewed as indices, where each variable represents a part of the construct. These two types of measurement models are presented in Figure 4.

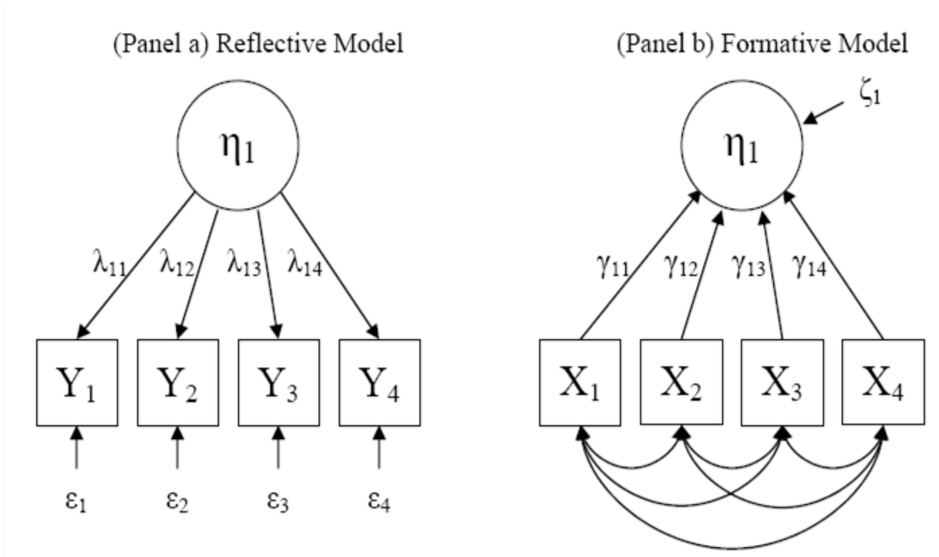

Figure 4. Reflective and Formative Measurement Models (Freeze & Raschke, 2007).

Another important aspect is that in a reflective measurement model the error consists of an inability of the latent construct to explain the measured variables, while in a formative measurement model the error represents the inability of the measured variables to explain the construct (Hair et al., 2009). Therefore, a construct that has formative measures when it should have reflective measures leads to a measurement model misspecification and will have a negative effect on the conclusions that can be drawn from the model (Freeze & Raschke, 2007).

The constructs in our research model are largely reflective with a single formative one. All of the reflective constructs have already been specified and validated in previous studies. On the other hand, the framework technical quality formative construct has not been previously specified or validated. Unlike a reflective measurement model that focuses on the general quality of the framework and its impact on perceived usefulness, ease of use, and continued framework usage intention, the formative measurement model allows us to understand what specific elements form the framework's technical quality and are the most important for users of the framework. By identifying the impact of each element, we believe that the formative measurement model will be more useful than a reflective one to both researchers and practitioners.

Based on the recommendation of Freeze and Raschke (2007), we used the multiple indicators and multiple causes model (MIMIC). This model requires two paths to be formed from the formative construct to two reflective indicators. The authors recommend this model because the formative construct is not dependent on the structural model. Thus, "future researchers are not bound by any constraints on how that construct is used in their theoretical model" (Freeze & Raschke, 2007, p. 1485).

Structural equation modeling. The main statistical technique used to test our hypotheses is structural equation modeling (SEM). This technique allows us to test the fit of our hypothesized model through model specification (Hair et al., 2009; Trochim & Donnelly, 2008). Compared with other statistical techniques such as multiple regression or partial least squares (PLS), covariance-based SEM has a number of advantages.

Compared to multiple regression, SEM allows for a construct to act as an endogenous variable in one relationship, while also acting as an exogenous variable in another relationship in the same model. Moreover, since in SEM these relationships are estimated simultaneously, this allows for a better understanding of unexplained covariances and helps accommodate measurement error (Hair et al., 2009).

When comparing PLS to LISREL, or covariance-based SEM, Haenlein and Kaplan (2004) indicated that PLS cannot guarantee the consistency of estimators and that it "tends to underestimate the correlations between the latent variables and overestimate the loadings" (p. 292). Third, while PLS can be used reliably with lower sample sizes and can arguably handle formative constructs easier than SEM (Haenlein & Kaplan, 2004), previous research has shown that, when the sample size is sufficient to conduct SEM, both SEM and PLS provide similar results (Hair et al., 2009).

Additionally, SEM is the standard statistical technique when using TAM (Benbasat & Barki, 2007; Hong et al., 2006; King & He, 2006) and was also used by Polančič et al. (2011). Thus, SEM allows us to better compare our results with those of previous research. Even though TAM is essentially a mediation model where perceived usefulness and ease of use fully mediate the relationship between the antecedents and continued usage intention, studies in the TAM literature do not focus on mediation hypotheses, but analyze the direct impact of the antecedents on perceived usefulness and ease of use.

The model specification of SEM consists of two parts: the measurement model and the structural model. The measurement model allows us to analyze the relationships between latent variables and their indicators while the structural model allows us to analyze the relationships between endogenous and exogenous variables, which are the counter-parts of dependent and independent variables (Hair et al., 2009; Trochim & Donnelly, 2008). Similar to a path analysis, the structural model allows us to test the impact of our selected technological characteristics on confidence, perceived usefulness, perceived ease of use, and continued usage intention. Both models provide a series of model fit indexes that help us assess the overall fit of the data to our hypothesized model (Trochim & Donnelly, 2008).

Once the measurement model was specified and estimated, model validity was assessed through the analysis of the goodness of fit (GOF) indicators and construct validity, which requires establishing convergent and discriminant validity, as well as nomological and face validity. Convergent validity was established by analyzing factor loadings on their respective constructs and ensuring that values surpass the 0.7 rule of thumb. Discriminant validity was established by comparing the average variance extracted to the square of the correlation estimate (Hair et al., 2009; Trochim & Donnelly, 2008).

Based on the results of the initial measurement model, if the initial model fit indexes are below recommended values, researchers go through a number of iterations until acceptable model fit is reached. During this process, based on statistical and theoretical considerations, certain items that have low factor loadings may be dropped. After the revised measurement model is complete, the analysis of the structural model and the path coefficients can proceed (Trochim & Donnelly, 2008).

Since in TAM the impact of the antecedents on the dependent variable, in our case the continued usage intention, is fully mediated (Davis, 1986), we assessed the direct relationship between each antecedent and the factor it was hypothesized to influence. We assessed each hypothesis based on the significance of the relationship, at a 95% confidence level ($p < 0.05$). Moreover, we used the standardized regression weights and the explained variance in the dependent variables (R^2) to understand the impact of each antecedent on the dependent variable.

The results of the study are presented in the following chapter.

Chapter 5: Results

After finalizing the survey, obtaining our sample frame and sample, and developing the research website, e-mail invitations to participate in the survey were sent to the 5,000 randomly selected framework users. The survey was opened 522 times and 204 complete responses were recorded, for a response rate of 4.1%. After removing surveys where respondents indicated at the end of the survey that they wanted to delete their answers and surveys that were deemed unusable (e.g., had "don't know" selected for all answers or had no variance), 189 surveys remained (3.8% net response rate).

Initial Steps

The data was exported from Qualtrics into SPSS, and prepared for analysis. First, the data was checked for consistency, out of range variables, and outliers. Since these initial checks did not identify any issues, the missing value analysis and descriptive statistics were generated. The data was then checked for normality, distribution and heteroscedasticity, as well as for independent sub-groups via independent samples t-tests. These initial steps allowed us to provide a visual representation of the data, as well as check the initial assumptions required for performing our statistical analysis (Hair et al., 2009; Trochim & Donnelly, 2008).

Missing value analysis. The purpose of the missing value analysis was to determine the type and extent of missing data, as well as to apply specific missing data remedies. While from a purely practical perspective, missing data reduces the effective sample size that can be used in the analysis, from a substantive perspective, non-random missing data can bias statistical results obtained from the analysis (Hair et al., 2009;

Trochim & Donnelly, 2008). The results of the missing value analysis are presented in Appendix E – Statistical Analysis – Initial Steps, Panel E.1.

As recommended by Hair et al. (2009), the first step in this analysis was to determine whether the missing data is ignorable, which means whether it is expected and part of the research design. An example of ignorable missing data is when the data collection procedure allows participants to skip a section of the survey, if they are not selected or qualified to answer it (e.g., programmers skipping the project management section). Our survey had no ignorable missing data, as all respondents were shown all items.

The second step of the missing value analysis consisted of determining the extent of missing data. Based on the general guidelines of conducting statistical analysis using the SEM, it is recommended that missing data should not exceed 10% per variable (Hair et al., 2009). In our case, out of the 45 variables that form our constructs, only 3 had missing values exceeding 10%. These variables are presented in Table 5. Missing data rates for all variables can be found in Appendix E – Statistical Analysis – Initial Steps, Panel E.1.

Table 5. Variables with Significant Missing Data.

Variable	**N**	**Missing**	
		Count	**Percent**
FTQ_6_E	143	46	24.3
FTQ_7_E	136	53	28.0
FTQ_8_E	157	32	16.9

All three variables that had missing data over 10% were part of the framework technical quality (FTQ) construct. The questions asked whether: (a) “The framework interfaces were role oriented”; (b) “The framework used role inheritance to pass down information”; and (c) “The framework used delegation to require specific functions from child components.” We then proceeded to analyze the individual cases that had missing values on these variables by looking at the answers for the other variables, as well as analyzing the comments and the positive and negative events. The respondents generally selected the “Not Important / Not Applicable” option in answering these questions. While most of the respondents did not provide any comments related to these questions, some of them indicated that their framework wasn't object oriented and that some of these items do not apply. After these three items were excluded, the hypothesized FTQ construct was measured by eight items.

The next two steps in the analysis consisted of diagnosing the randomness of the missing data and then selecting the imputation method. Determining the randomness of the missing data is essential because different missing data remedies have to be applied if the data is Missing at Random (MAR) versus Missing Completely at Random (MCAR). MAR data occurs when the missing values in variable Y are dependent on variable X, but not Y. On the other hand, with MCAR data missing values in variable Y are not dependent on variable X, which makes the observed Y variables a true random sample of Y (Hair et al., 2009).

As recommended by Hair et al. (2009), we determined the level of missing data randomness by creating a sub-group for all cases that had missing values and proceeded by performing an independent samples t-test between the group that had missing values and the one that did not have any missing values. The independent samples t-test is conducted on a continuous dependent variable and utilizes a grouping variable to determine the two groups. The results of the test are interpreted by first observing the significance of Levene's test for the equality of variance. Levene's test helps us determine whether the variability of the scores in the two groups is similar or not. A non-significant result in this test shows that the amount of variability between the two scores is similar. Based on the results of Levene's test, we then proceed to observe the two-tailed significance of the independent samples t-test. A non-significant value indicates that the means of the two groups are not statistically different based on the grouping variable. Therefore, this result would indicate that there is no underlying process for the missing data, and the data can therefore be classified as MCAR.

For example, a sub-group was created for all respondents that had missing data on the PO1 item (see Appendix E, Panel E.1). Based on the two groups, an independent samples t-test was performed on each of the dependent variables. The results were analyzed and the process was repeated on each variable that had missing data. After performing the independent samples t-test, the results showed that there are no significant differences between any of the groups. Based on these results and the fact that the amount of missing data was less than 10% per variable or case, we concluded that the missing data can be classified as MCAR.

The last step in the missing value analysis is determining how to impute missing data values. Based on the recommendations of Hair et al. (2009), as well as the type and extent of missing data, mean substitution was determined to be an appropriate approach for imputing missing values. This imputation was performed in SPSS.

Descriptive statistics. As presented above, our dataset consisted of 189 completed surveys. A short overview of the descriptive statistics are presented in Tables 6 and 7. The complete descriptive statistics are presented in Appendix E – Statistical Analysis – Initial Steps, Panel E.2.

All respondents indicated that they heard about the survey based on the invitation e-mail that was sent to the SourceForge sample. Almost 80% of our respondents had some form of post-secondary IT education, and over 95% had more than five years of software development experience. Moreover, around 85% of our respondents have used three or more frameworks and around 75% of them have seven or more months experience with their chosen framework. Our sample exhibits similar characteristics to that of Polančič et al. (2011).

Our respondents indicated experience with 129 different frameworks. Combined with the fact that no framework represented more than 8% of the responses, our sample isn't biased towards a particular framework.

In terms of blackbox versus whitebox frameworks, 40% indicated that their chosen framework was a blackbox framework, thus providing us with a good mix between the two types. In terms of voluntary versus mandatory usage, the vast majority (92%) indicated that they had selected the framework voluntarily for the application. This answer is to be expected, especially in the case of open-source and expert users in general.

In regards to their chosen application's platform, 63% of our respondents indicated that the application was for the Web, approximately 43% indicated that they developed a PC application, while around 15% indicated that the application was developed for a mobile platform. Around 10% of our respondents indicated that the application was used for a different platform, most of them indicating this platform as Unix specific or proprietary hardware. The total exceeds 100% because 25% of our respondents indicated that the application was developed for more than one platform.

Overall, these responses provide a good mix between the different frameworks and application platforms, and will allow us to determine if there are any differences between the sub-groups in our sample.

Table 6. Descriptive Statistics.

Variable	Values	Frequency	Percent
Level of IT Education	No formal IT Education	29	15%
	Certificate / Diploma	27	14%
	Bachelor	48	26%
	Masters / Doctorate	75	40%
	Other	10	5%
Software Development Experience (years)	1-5	12	6%
	6-10	38	20%
	11-15	65	34%
	16-20	43	23%
	21-25	14	8%
	26+	17	9%
Number of Frameworks Used	1	9	5%
	2	19	10%
	3-4	45	24%
	5-6	39	21%
	7-10	36	19%
	11+	41	21%
User Experience with Chosen Framework (person months)	1-3	25	13%
	4-6	22	12%
	7-12	25	13%
	13-24	34	18%
	25-48	39	21%
	49+	44	23%

Table 6. Descriptive Statistics.

Variable	Values	Frequency	Percent
Framework Usage	Voluntary	175	93%
	Mandatory	14	7%
Framework Type	Whitebox	113	60%
	Blackbox	76	40%
User Framework Contribution Level	None	90	48%
	Occasional contributions	57	30%
	Significant contributions	8	4%
	Core developer	34	18%
How long ago was the application developed (months)	6	143	76%
	12	25	13%
	24	5	3%
	More	16	8%
Application Platform Note: Respondents were allowed to select more than one platform.	Web	120	63%
	PC	82	43%
	Mobile	29	15%
	Other	18	10%
Number of Platforms	Single-platform	141	75%
	Multi-platform	48	25%
Number of Unique Frameworks		129	
Top 10 Frameworks Used	Spring	16	8%
	Qt	11	6%
	Zend	7	4%
	Django	6	3%
	Eclipse	6	3%
	CodeIgniter	3	2%
	Ruby on Rails	3	2%
	Drupal	2	1%
	Equinox OSGi	2	1%
	Grails	2	1%

Table 7. Descriptive Statistics for Variables.

Variable	N	Min	Max	Mean	Standard Deviation
PO_1	189	1	7	4.99	1.94
PO_2	189	1	7	4.71	1.94
PO_3	189	1	7	4.95	1.96
FL_1	189	1	7	5.92	1.33
FL_2	189	2	7	5.99	1.25
FL_3	189	1	7	5.92	1.28
EF_1	189	2	7	5.87	1.24
EF_2	189	3	7	6.26	0.96
EF_3	189	2	7	6.16	1.01
UD_1	189	1	7	5.55	1.42
UD_2	189	1	7	5.02	1.64
UD_3	189	1	7	4.88	1.67
UD_4	189	1	7	5.69	1.40
FS_1	189	3	7	6.21	0.92
FS_2	189	1	7	6.06	1.09
FS_3	189	2	7	6.18	1.01
FTQ_1	189	2	5	4.46	0.78
FTQ_2	189	1	5	4.05	1.06
FTQ_3	189	1	5	3.86	1.08
FTQ_4	189	2	5	4.35	0.79
FTQ_5	189	1	5	4.17	1.15
FTQ_9	189	1	5	4.25	0.91
FTQ_10	189	1	5	3.79	1.25
FTQ_11	189	1	5	3.68	1.30
SP_1	189	1	7	5.15	1.54
SP_2	189	1	7	5.24	1.38
SP_3	189	1	7	5.23	1.51
SP_4	189	1	7	4.96	1.48
SP_5	189	1	7	5.14	1.34
CF_1	189	1	7	6.04	1.24
CF_2	189	1	7	5.86	1.21
CF_3	189	1	7	5.50	1.29
PU_1	189	1	7	6.29	1.05
PU_2	189	2	7	6.33	1.03
PU_3	189	3	7	6.51	0.83
EOU_1	189	1	7	5.69	1.30
EOU_2	189	1	7	5.89	1.23
EOU_3	189	2	7	5.97	1.12
CFUI_1	189	1	7	6.27	1.30
CFUI_2	189	1	7	6.20	1.28
CFUI_3	189	1	7	6.16	1.24
CFUI_4	189	1	7	6.12	1.31

Independent groups. The purpose of the independent samples t-test was to detect if there are any distinct or unrelated sub-groups in our sample. Performing this test is essential to our research because it allows us to determine if the means of the dependent variables vary based on specific grouping criteria. If the results of the test show that there isn't a significant difference between the two sub-samples, then the respondents can be pooled together for the analysis. If there is a significant difference between the groups, then the researcher should perform a more in-depth analysis to ascertain its cause (Hair et al., 2009; Trochim & Donnelly, 2008).

For our analysis, we conducted independent sample t-tests on (a) early versus late respondents, (b) voluntary versus mandatory framework users; (c) blackbox versus whitebox frameworks; (d) PC versus non-PC applications; (e) web versus non-web applications; (f) mobile versus non-mobile applications; (g) low versus significant contribution levels; (h) certificate/bachelor education levels versus graduate education; and (i) less than five frameworks used versus more than five. The results of the tests indicated no significant differences between these groups. Therefore, the respondents were pooled for the rest of the analysis. While the tests were conducted on each dependent variable, to conserve space, only the results of the independent samples t-tests on CFUI1 are presented in Table 8.

Table 8. Independent Samples t-test.

Group	Mean	Levene's test		t-test		
		F	Sig.	t	df	Sig.
Early versus late respondents	6.33 6.03	.104	.747	1.270	187	.206
Voluntary versus mandatory users	6.30 5.93	.633	.427	1.022	187	.308
Blackbox versus whitebox frameworks	6.30 6.25	.500	.480	-0.284	187	.777
PC versus non-PC applications	6.26 6.28	.302	.584	-0.127	187	.899
Web versus non-web applications	6.30 6.22	.411	.522	0.420	187	.675
Mobile versus non-mobile applications	6.31 6.26	.043	.835	0.182	187	.856
Contribution level: low (1,2) versus significant (3,4)	6.13 6.45	.794	.374	1.451	187	.148
Education: certificate/ bachelor versus graduate education	6.35 6.18	1.531	.217	-0.893	187	.373
Software development experience: <10 years versus ≥10	6.41 6.24	1.814	.180	-0.707	187	.481
Numbers of frameworks used: <5 versus ≥5	6.34 6.22	3.209	.075	-0.609	187	.543

Data normality. The purpose of the checks for data normality, distribution and homoscedasticity is to provide statistical evidence that the assumptions of SEM have been met. Moreover, as indicated by Hair et al. (2009), data that deviates from multivariate normality requires larger sample sizes to adequately conduct SEM.

However, the authors also indicate that as sample sizes reach a threshold of 200 respondents, the impact of non-normal data tends to diminish (Hair et al., 2009). Therefore, our sample size of 189 respondents should not be significantly influenced by non-normal data.

To analyze data normality, we used SPSS to generate histograms and normal probability plots, as well as scatter plots to check for homoscedasticity. The skewness and kurtosis values of each variable were also checked and are presented in Appendix E – Statistical Analysis – Initial Steps, Panel E.3.

While the skewness values were between the acceptable range of ± 2, the kurtosis values of the dependent variables exceeded the +3 threshold (Hair et al., 2009). However, since the dependent variables measure continued framework usage intention, a ceiling effect is expected. Framework users should rate these frameworks highly on these variables. Combined with the fact that kurtosis only affects the dependent variables and the previous discussion on the diminished impact of non-normal data with higher sample size, we concluded that our sample has met the assumptions for conducting SEM.

Sample size. While the minimum requirement to perform SEM is to have one more observation than the number of covariances, increased sample size produces more information, enhances stability of the solutions, and reduces variability (Hair et al., 2009). While some authors (Bryant & Yarnold, 1995; MacCallum et al., 1999) have indicated that a subject to variable ratio (STV) of 5 and a sample size around 200 is sufficient, others (Hair et al., 2009) have stated that the recommended sample size for conducting SEM varies based on a variety of factors such as missing data, data normality, estimation technique, model complexity, the number of measured items per construct, and communalities between items.

Since missing data and data normality have already been addressed, this section will focus on the estimation technique, model complexity, and the number of measured items per construct. The term communality refers to the amount of variance in a variable explained by its latent factor (construct). In other words, variables that have low communalities are an indicator of poor reliability and require larger sample sizes for model stability. While communalities will be addressed in the measurement model, it is important to note that our constructs do not suffer from low communalities and exceed the 0.6 threshold (Hair et al., 2009). The relevant statistics are presented in Appendix F - Statistical Analysis - Measurement Model, Panel F.1.

For the estimation procedure, we used maximum likelihood estimation (MLE). At a high level, MLE is an iterative procedure that tries at each step to improve parameter estimates to improve model fit with the data. One of the most commonly used SEM estimation procedures, it is robust and provides valid and stable results with sample sizes as small as 50 (Hair et al., 2009). As a general guideline, the authors suggest using MLE with sample sizes between 100 and 400, a guideline met by our sample size of 189.

In regards to model complexity and the number of measured items per construct, it is generally recommended to not have underidentified constructs (less than three variables per construct), as these models require increased sample sizes (Hair et al., 2009). We do not have any underidentified constructs, with some constructs having 4 or 5 observed variables. However, the research model contains 11 constructs, which does increase model complexity.

Overall, we believe that our sample size is adequate to analyze the relationships hypothesized in our model and to reach a stable solution that may be generalizable to the population of interest and replicable in other studies.

Measurement Model

As recommended by Hair et al. (2009) and as presented in the methodology section, we used a two-stage approach to conduct SEM. In the first stage, we specified and analyzed the measurement model, while in the second stage we will focus on the structural model. The measurement model is used for specifying the indicators for each construct and analyzing the relationships between latent variables and their indicators. As such, establishing measurement model validity depends on assessing the overall goodness of fit (GOF) of the measurement model, as well as providing evidence of construct validity (Hair et al., 2009).

After specifying the hypothesized measurement model in Amos and conducting the initial analysis in SPSS, we then analyzed the GOF of the model. Since there is no single GOF measure that can provide evidence for acceptable fit, all three types of GOF measures should be examined: (a) absolute fit indices; (b) incremental fit indices; and (c) parsimony fit indices (Hair et al., 2009).

Absolute fit indices provide a basic assessment of how the specified model reproduces observed data, independent of any alternative models. These indices include the chi-square (χ^2) statistic, root mean square error of approximation (RMSEA), root mean square residual (RMR), standardized root mean square residual (SRMR), goodness of fit index (GFI), and the normed chi-square (χ^2/df) (Hair et al., 2009).

Incremental fit indices assess how the specified model fits data compared to alternative models, such as the null model. The null model assumes that all variables are uncorrelated and that no model specification can improve the fit of the model (Hair et al., 2009). These indices include the Tucker Lewis index (TLI), comparative fit index (CFI), normed fit index (NFI), and the relative noncentrality index (RNI).

Parsimony fit indices, assess how the specified model fits data compared to a set of competing models, on the criterion of comparing model fit with parsimony (Hair et al., 2009). These indices include the adjusted goodness of fit index (AGFI) and the parsimony normed fit index (PNFI).

Indicators within each group tend to have similar values and therefore it isn't necessary to report on each of them. It is generally recommended to report on a combination of GOF indices from each of the three groups. As such, we have chosen to report on the indicators recommended by Hair et al. (2009), as well as those by used by Polančič et al. (2011)

After obtaining the results from the hypothesized measurement model, some of the indices were not suggesting a good model fit. The results for the hypothesized measurement model are provided in Table 9, under the Initial Model column.

Table 9. Measurement Model Fit Indices.

Index	Recommended Value	Initial Model	Revised Model
χ^2 (df, p)	Significant values expected	880.82 (482, $p < 0.001$)	453.668 (305, $p < 0.001$)
χ^2/df	≤ 3.00	1.83	1.49
RMSEA	≤ 0.08	0.07	0.05
CFI	≥ 0.92	0.91	0.96
TLI	≥ 0.92	0.90	0.95
RMR	≤ 0.09	0.11	0.08
AGFI	≥ 0.80	0.74	0.81

To analyze the issue of relatively poor model fit, we began with the results of the confirmatory analysis. The analysis indicated that several variables had standardized loadings below the 0.7 threshold (Hair et al., 2009). Since the other variables loaded highly on the constructs and no construct would remain with less than two indicators, we removed one understandability item, one framework suitability item, three social pressure items, and one confidence item. The standardized loadings of the initial measurement model are presented in Appendix F - Statistical Analysis - Measurement Model, Panel F.1. After removing these variables, we ran the analysis again with the revised model. This produced significantly better model fit, which can be observed in Table 9, under the Revised Model column.

Establishing construct validity requires evidence of convergent validity and discriminant validity, as well as nomological and face validity. Since nomological and face validity issues have been addressed in the design of the survey and during the pre-test process, we will focus convergent and discriminant validity, as well as reliability. Tables 10 and 11 contain all indicators relevant for establishing construct validity of the revised measurement model.

Table 10. Reliability and Convergent and Discriminant Validity.

Variable	CR	AVE	MSV	ASV
1. CFUI	0.962	0.864	0.643	0.268
2. PO	0.928	0.811	0.088	0.051
3. FL	0.859	0.670	0.501	0.299
4. EF	0.807	0.583	0.475	0.274
5. UD	0.873	0.696	0.425	0.141
6. FS	0.735	0.582	0.391	0.211
7. SP	0.814	0.687	0.077	0.040
8. CF	0.725	0.570	0.475	0.255
9. PU	0.905	0.761	0.643	0.321
10. EOU	0.878	0.706	0.437	0.279

Table 11. Correlations.

Variable	1	2	3	4	5	6	7	8	9	10
1. CFUI	**0.93**									
2. PO	0.24	**0.90**								
3. FL	0.69	0.28	**0.82**							
4. EF	0.51	0.21	0.60	**0.76**						
5. UD	0.27	0.24	0.30	0.44	**0.83**					
6. FS	0.50	0.20	0.63	0.58	0.28	**0.76**				
7. SP	0.26	0.11	0.22	0.09	0.23	0.15	**0.83**			
8. CF	0.54	0.30	0.62	0.69	0.43	0.46	0.18	**0.76**		
9. PU	0.80	0.18	0.71	0.67	0.34	0.60	0.19	0.57	**0.87**	
10. EOU	0.55	0.23	0.60	0.58	0.65	0.46	0.28	0.55	0.66	**0.84**

Note: Diagonal represents the square root of AVE.

Convergent validity means that variables that are indicators of a common latent construct share a large amount of variance in common. Convergent validity is established by analyzing factor loadings, average variance extracted, and reliability (Hair et al., 2009). Factor loadings are essential for establishing convergent validity as items that load highly on a single factor indicate that they converge on a common point. Moreover, the square of a standardized factor loading represents the amount of variance explained by the factor, or the variance extracted from the item. As recommended by Hair et al. (2009), even if significant, items with factor loadings below 0.7 are not recommended, as that would mean the factor explains less than half of the variance in the item. In our revised measurement model, all items have factor loadings above 0.7. The standardized factor loadings for the initial and revised measurement model are presented in Appendix F - Statistical Analysis - Measurement Model, Panels F.1 and F.2.

The average variance extracted (AVE) represents the arithmetic mean of the variance extracted from the items loading on a construct. Similar to the discussion on factor loadings, AVE values are recommended to be above the 0.5 threshold, as this would indicate that, on average, at least half of the variance in the item is explained by the latent factor rather than by error variance (Hair et al., 2009). For the revised measurement model, the AVE for all factors is above the recommended 0.5 threshold (Table 10).

Reliability refers to the overall consistency of a measure and is also an indicator of convergent validity. While there are multiple available reliability estimates, two of the most widely used ones are Cronbach's alpha and composite reliability (CR). Based on the observation of Hair et al. (2009, p. 687) that "different reliability coefficients do not produce dramatically different reliability estimates" and, because the CR indicator is commonly used with SEM models, we decided to use this indicator for establishing reliability and convergent validity. Hair et al. (2009) recommend that all CR values should be above the threshold of 0.7 and that the CR value for each construct be higher than the construct’s AVE value. Our results meet both these criteria (Table 10).

Having established convergent validity, our next step was to find evidence of discriminant validity, or whether a construct differs from others. Hair et al. (2009)

recommend that a rigorous test to determine discriminant validity is to compare the AVE values with the square of the correlation estimate. Theoretically, the construct's explained variance should be higher than the variance shared with other constructs (Hair et al., 2009). We calculated the maximum shared square variance (MSV) indicator and the average shared squared variance (ASV) for each construct and compared them with the AVE values. After analyzing the resulting data, we found that the AVE values were higher than the MSV or ASV values for each of the constructs (Table 10). Together with the fact that the square root of the AVE indicator for each construct was higher than its correlation values (Table 11), sufficient evidence of discriminant validity was provided.

Having established convergent, discriminant, nomological, and face validity, as well as reliability, construct validity is established. Before establishing measurement validity, Podsakoff et al. (2003) recommended that testing for common method bias should also be performed. As discussed in the methodology section, data collected using one collection method may introduce response bias, which may inflate or deflate responses. To provide statistical evidence against common method bias, Hair et al. (2009) recommended the use of Harman's single factor test.

Harman's single factor test was performed in SPSS via an exploratory factor analysis (EFA). Based on the recommendations of Hair et al. (2009), both independent and dependent variables were included in the EFA analysis. Common method bias is an issue if the results of the un-rotated components matrix show the first factor accounting for more than 50% of the total variance. The analysis indicated that the single factor solution explained 38.7% of the variance. Thus, Harman's single factor test indicated that common method variance was not an issue.

Based on the recommendations of Podsakoff et al. (2003), a more robust test for common method bias can be performed by comparing the model fit of the revised measurement model to that of a single-factor model. After conducting this test in Amos, the results showed the revised measurement model had better GOF indices, which indicated that common method bias was not an issue in our study. The results are presented in Table 12.

Table 12. Common Method Bias.

Index	Recommended Value	Single-Factor Model	Revised Model
χ^2 (df, p)	Significant values expected	2099.369 (350, $p < 0.001$)	453.668 (305, $p < 0.001$)
χ^2/df	≤ 3.00	6.00	1.49
RMSEA	≤ 0.08	0.16	0.05
CFI	≥ 0.92	0.55	0.96
TLI	≥ 0.92	0.51	0.95
RMR	≤ 0.09	0.31	0.08
AGFI	≥ 0.80	0.42	0.81

Combined with the previous analysis of the GOF indicators, measurement model validity is also addressed. Thus, we proceeded to specify and assess the validity of the structural model.

Hypothesized Structural Model

The specification and analysis of the structural model represents the second phase of our SEM analysis. The structural model is used to evaluate the hypothesized relationships of variables between constructs. Based on the results of the measurement model and the hypothesized relationships in our research model, we specified the dependence relationships in our structural model by adding directional arrows to represent each hypothesis (Hair et al., 2009; Trochim & Donnelly, 2008).

Before analyzing the structural model validity and path coefficients, it is important to assess the validity of the framework technical quality construct. As discussed in the methodology section, assessing the validity of a MIMIC formative measurement model requires at least two paths from the formative construct to two reflective constructs (Freeze & Raschke, 2007; Hair et al., 2009; Roberts & Thatcher, 2009). Based on our research model (Figure 3) the framework technical quality construct was hypothesized to have a positive impact on confidence, usefulness, and ease of use. As recommended (Freeze & Raschke, 2007; Roberts & Thatcher, 2009), we proceeded to analyze the regression weights and the standardized regression weights and remove variables that were found to be non-significant. After each variable was removed, we ran the model again and repeated the procedure. Because of non-significant values, six of the eight variables were removed in this process. The remaining items, FTQ2 and FTQ3 were not theoretically sufficient on their own to perform the analysis and therefore it was decided to remove the framework technical quality construct from the model. We will expand more on the framework technical quality analysis in the following section.

Similar to the measurement model, structural model validity was assessed by analyzing the GOF indicators of the structural model and by comparing them to both recommended thresholds and the results of the revised measurement model. The GOF indicators suggest that the model has an acceptable fit to the data. However, these values could be improved by respecification. The hypothesized structural model fit indices are presented in Table 13.

Table 13. Hypothesized Structural Model Fit Indices.

Index	Recommended Value	Revised Measurement Model	Hypothesized Structural Model
χ^2 (df, p)	Significant values expected	453.668 (305, $p < 0.001$)	626.973 (371, $p < 0.000$)
χ^2/df	≤ 3.00	1.49	1.69
RMSEA	≤ 0.08	0.05	0.06
CFI	≥ 0.92	0.96	0.94
TLI	≥ 0.92	0.95	0.93
RMR	≤ 0.09	0.08	0.17
AGFI	≥ 0.80	0.81	0.79

The path coefficients and their standardized regression weights are presented in Table 14.

Table 14. Hypothesized Structural Model Standardized Regression Weights.

Hypothesis	Relationship			Standardized Estimate	Supported
H1a	EOU	→	CFUI	-0.033 NS	No
H1b	PU	→	CFUI	0.755 ***	Yes
H1c	EOU	→	PU	0.263 **	Yes
H2a	CF	→	PU	-0.020 NS	No
H2b	CF	→	CFUI	0.101 NS	No
H3	SP	→	CFUI	0.114 *	Yes
H4a	PO	→	PU	-0.041 NS	No
H4b	PO	→	EOU (-)	-0.032 NS	No
H5a	FL	→	PU	0.329 **	Yes
H5b	FL	→	EOU (-)	0.467 ***	No
H6	EF	→	PU	0.169 *	Yes
H7a	UD	→	EOU	0.523 ***	Yes
H7b	UD	→	CF	0.276 **	Yes
H8a	FS	→	PU	0.220 NS	No
H8b	FS	→	CF	0.580 ***	Yes
H9a	FTQ	→	PU	Removed	-
H9b	FTQ	→	EOU	Removed	-
H9c	FTQ	→	CF	Removed	-

* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$

Since each path represents a hypothesized relationship in our research model, this also allows us to determine which of our hypotheses were supported. The results indicate that several of our hypotheses were not supported. This section will focus on the results of the hypothesized model, while the discussion section will focus on the overall findings and implications of the study in regards to existing literature.

From the base model hypotheses, neither of the two confidence hypotheses were supported. In other words, we could not find sufficient statistical evidence that confidence has a positive impact on usefulness (H2a) or on continued framework usage intention (H2b). While the implications of these findings will be presented in the discussion section, it is important to note they differ from the suggestions of Polančič et al. (2011), who theorized a positive impact of confidence on continued usage intention.

While two of the original TAM hypotheses, H1b and H1c, were supported, our study could not find sufficient statistical evidence to support hypothesis H1a, which theorizes that ease of use should have a positive impact on continued framework usage intention. Although one of the base hypotheses, in the case of a complex system such as open-source frameworks, it is likely that, because of their skills and experience, users will be more interested in the system's usefulness rather than its ease of use. This explanation is supported by the findings of King & He (2006), who also reported that professional users are less likely to be influenced by a system’s ease of use.

Social pressure was hypothesized to have a positive impact on the user's continued framework usage intention (H3). This hypothesis was supported in our analysis and its implications will be addressed in the discussion section.

In regards to the antecedents, flexibility, efficiency and understandability were found to have a significant impact on perceived usefulness and perceived ease of use.

However, portability did not have a significant impact on either of these factors. This can be explained by the fact that most respondents indicated that they used their chosen application for a single platform. As such, portability to other platforms was not a factor in deciding whether to continue using the framework. The number of respondents who used the framework to develop the application for more than one platform (48 or 25%) was insufficient for a separate analysis.

As discussed in the literature review section, the construct of framework suitability was developed by Polančič et al. (2011) based on the original TTF construct to measure the fit between the capabilities of the framework and the requirements of the application. Our results indicate framework suitability has a positive impact on the user's confidence in the framework (H8b), but does not impact perceived usefulness (H8a). We expect this to be partially attributed to the user's experience with the framework and the fact that open-source frameworks are highly customizable by the user. As such, application specific functions don't have to be pre-built in the framework, as expert users can simply extend the framework to provide these required functions.

The last set of hypotheses theorized the impact of the framework technical quality construct on confidence (H9c), perceived usefulness (H9a) and perceived ease of use (H9b). With framework technical quality removed, we were unable to test these hypotheses. Based on results of the model fit indicators and the path analysis, we determined that our hypothesized model is not acceptable and therefore proceeded to create and analyze a revised structural model.

Revised Structural Model

The revised structural model was created by incrementally removing non-significant paths in the hypothesized structural model and then assessing model fit, regression weights, residual covariances, and modification indices at each step. In addition to utilizing the principle of model parsimony, we ensured that each relationship in the revised model had theoretical support (Hair et al., 2009). The fit indices for the revised model are presented in Table 15.

Table 15. Revised Structural Model Fit Indices.

Index	Recommended Value	Hypothesized Structural Model	Revised Structural Model
χ^2 (df, p)	Significant values expected	626.973 (371, p < 0.000)	415.374 (256, p < 0.000)
χ^2/df	≤ 3.00	1.69	1.62
RMSEA	≤ 0.08	0.06	0.06
CFI	≥ 0.92	0.94	0.95
TLI	≥ 0.92	0.93	0.95
RMR	≤ 0.09	0.17	0.09
AGFI	≥ 0.80	0.79	0.82

While the revised model has better GOF indicators than the hypothesized model, Hair et al. (Hair et al., 2009) recommended that competitive fit should also be tested against the single factor model. In addition to the authors' recommendations, the revised model's fit was also tested against a two-factor and a three-factor model. The two-factor model had all the exogenous variables grouped on one factor and all the endogenous variables grouped on the other. The three factor model had all the exogenous variables grouped on one factor, while confidence, perceived usefulness, and perceived ease of use were grouped on the second factor. The third factor contained the continued framework usage intention variables. The results indicated that the revised structural model had the best GOF indicators across all the competing models. These results are presented in Table 16 and the revised structural model is presented in Figure 5.

Table 16. Competing Models Fit Indices.

Index	Recommended Value	Single-Factor Model	Two-Factor Model	Three-Factor Model	Revised Structural Model
χ^2 (df, p)	Significant values expected	1581.456 (275, p < 0.000)	1413.986 (274, p < 0.000)	1137.447 (273, p < 0.000)	415.374 (256, p < 0.000)
χ^2/df	≤ 3.00	5.75	5.16	4.17	1.62
RMSEA	≤ 0.08	0.16	0.15	0.13	0.06
CFI	≥ 0.92	0.62	0.67	0.75	0.95
TLI	≥ 0.92	0.58	0.63	0.72	0.95
RMR	≤ 0.09	0.21	0.21	0.18	0.09
AGFI	≥ 0.80	0.44	0.48	0.57	0.82

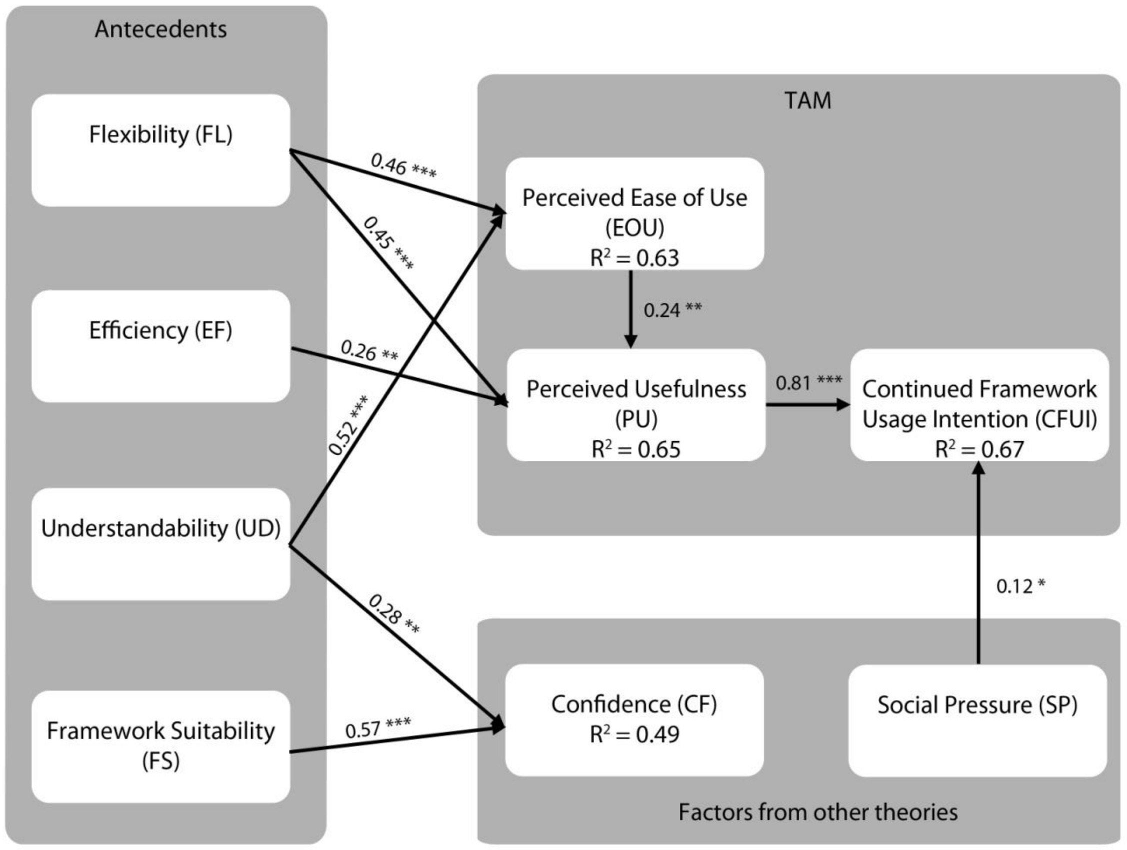
Antecedents
Flexibility (FL)
Efficiency (EF)
Understandability (UD)
Framework Suitability (FS)
TAM
Perceived Ease of Use (EOU) R² = 0.63
Perceived Usefulness (PU) R² = 0.65
Continued Framework Usage Intention (CFUI) R² = 0.67
Confidence (CF) R² = 0.49
Social Pressure (SP)
Factors from other theories
0.46 ***
0.45 ***
0.26 **
0.52 ***
0.28 **
0.57 ***
0.24 **
0.81 ***
0.12 *

Figure 5. Revised Structural Model.

The revised structural model displays better GOF indicators and has greater parsimony than the hypothesized model. Moreover, all its relationships were hypothesized in the original model. In comparison to the original TAM model, the revised model does not have a direct path linking ease of use to continued framework usage intention. As discussed in the hypothesized structural model section, we attribute this finding to the fact that open-source framework users are also software developers and can therefore be labeled as expert users. Moreover, since frameworks are used to develop other software applications, they have an inherent complexity that is expected by the user. As such, for expert users utilizing an inherently complex system, it can be expected that ease of use is not a determining factor for continued framework usage intention. This explanation is also supported by the existing literature on TAM (King & He, 2006) and will be addressed in more detail in the discussion section.

Based on these arguments, we consider the revised structural model to be preferred over the hypothesized one. We will continue discussing our findings in the following chapter.

Chapter 6: Discussion

Based on the results of the statistical analysis, the purpose of this section is to discuss the main findings of the study, as well as their theoretical and practical implications. Additionally, we will also discuss some of the limitations of this study, as well as opportunities for further research.

Theoretical Implications

From a theoretical perspective, our study follows the recommendations of Benbasat & Barki (2007) that contributions to the TAM literature should be made by identifying the antecedents of perceived usefulness, ease of use, and behavioral intentions. Based on the classification of King & He (2006), the analysis of prior factors such as framework suitability, understandability, flexibility, and efficiency are type 1 modifications to TAM, while the inclusion of factors from other theories, such as social pressure, are type 2 modifications.

Our study also adds to the TAM literature on post-adoption and provides a number of methodological refinements over previous studies (Polančič et al., 2011), such as the independent measurement of portability and flexibility. We will discuss each of these theoretical contributions in the following pages.

Original TAM hypotheses. As discussed in the literature review, the original TAM model hypothesizes that the usage intention is influenced by the user's perceptions that the technology is useful and easy to use. Moreover, ease of use is also hypothesized to have a positive impact on perceived usefulness (Davis, 1986).

However, the direct effect of perceived ease of use on continued usage intention was not supported in our context. A possible explanation is that open-source frameworks are inherently complex systems used for software development and that framework users are expert users of those systems. As such, utilizing a framework requires the user to have a certain level of skill and experience, which, in turn, reduces the importance of ease of use. The high level of skill and experience can be observed in our sample where 80% of our respondents had some form of post-secondary IT education, and 94% had more than five years of software development experience (Table 6.). Additionally, the mean of each ease of use variable was above 5.6 (out of a maximum of 7 – see Table 7), indicating that respondents found their chosen frameworks easy to use.

These arguments are also supported by the meta-analysis of King & He (2006) in which, after analyzing 88 TAM studies, the authors state that, while the impact of perceived usefulness on the dependent variable is significant, the direct effect of ease of use is not always significant. In regards to the effects of ease of use on usefulness and of usefulness on continued framework usage intention, both hypotheses were found to be significant at the $p > 0.001$ and $p > 0.002$ levels, respectively.

Confidence and social pressure. Including the constructs of confidence and social pressure in our analysis represents a type 2 modification to TAM (King & He, 2006). While not analyzing the impact of confidence on continued framework usage intention, Polančič et al. (2011) theorized that confidence, perceived ease of use, and perceived usefulness are the three main determinants of continued framework usage intention. They found that their task-technology fit construct, which was also referred to as framework suitability, and understandability are the main determinants of confidence.

Our analysis corroborated their findings in terms of confidence's antecedents and its explained variance ($R^2 = 0.49$). Similar to their results, our SEM analysis did not support the impact of confidence on perceived usefulness. However, while Polančič et al. (2011) theorized a positive impact of confidence on continued usage intention, our study could not find statistical evidence to support this. To better understand this result, we analyzed the comments, as well as the positive and negative events that our respondents identified when using the framework. When describing negative events, several respondents indicated different types of framework failure that affected their usage. For example, one user stated that "some unexpected bugs delayed / confused me. I had to find workarounds for the bug to get ahead" while another indicated component problems such as "Web Sockets were unstable." However, each of these respondents still indicated that they would still use the framework if they were to develop the same application. Thus, we can theorize that while respondents experienced framework failures, they were able to get around them and decided to continue using the framework. This suggests that, as long as the user can overcome framework failures, confidence in the framework does not have a significant impact on continued usage intention. We believe this to be a matter worth investigating and recommend that future framework studies should analyze the relationship between confidence and other TAM constructs.

Based on the findings of previous studies on software development and methodologies (Hardgrave et al., 2003), we hypothesized that social pressure should have a positive impact on the user's continued framework usage intention. As our statistical analysis provided evidence to support this hypothesis, we can conclude that social pressure through the online community does play a role in influencing the user's decision to continue using a framework. This is consistent with the key characteristics of open-source frameworks, which allow users to make contributions to the framework and share them with the framework's community, thus expanding and improving the framework. As such, it can be argued that a framework's success can be influenced by its characteristics as well as the framework’s community.

TAM and confidence antecedents. A key contribution of our study is our identification and analysis of the antecedents of confidence, perceived usefulness, ease of use, and continued usage intention. Our findings suggest that framework suitability and understandability have a significant impact on the user's confidence in the framework while perceived usefulness is mainly determined by flexibility and efficiency, as well as ease of use.

Ease of use is mainly determined by understandability and flexibility. While we expected flexibility to have a negative impact on ease of use, as increased flexibility may increase the framework's complexity, our findings suggest that flexibility has a positive impact. This result can be explained by the fact that framework users are expert users of an already complex system. As such, for this expert group, the increase in complexity is negligible to the increase in functionality. Since this increased functionality may lead to shorter development times and resource costs, then framework users may view this added flexibility as an increase in ease of use rather than a decrease.

Another important discussion point relates to the construct of portability. As discussed in the literature review section, portability and flexibility are two of the four main constructs identified by the REBOOT model that affect the quality of frameworks

(Sindre et al., 1995). While Polančič et al. (2011) measured these two constructs together, they recommended that further studies should try measuring them separately, as they may represent distinct concepts. Our results support the view that portability and flexibility are distinct constructs (Tables 10 and 11).

While the impact of flexibility on both perceived usefulness and ease of use was found to be significant, our analysis did not find sufficient evidence to support the hypothesis that portability impacts perceived usefulness and ease of use. We believe this result is at least partially because only 25% (48) of our respondents developed their application for more than one platform. Since portability measures the ability of the framework to be used across different platforms and environments, if the framework user only utilizes the framework for one platform, then the importance of portability is greatly diminished. Therefore, we believe that this is the main reason why portability was not found to have a significant impact on perceived usefulness and ease of use. Apart from obtaining a larger sample, we recommend that future studies explore the importance of portability by asking framework users about their experience with developing cross-platform applications, or by focusing on open-source frameworks that are specifically designed for cross-platform application development.

Framework technical quality. We developed the framework technical quality construct based on several guidelines found in the literature. Thus, the construct used a formative measurement model to determine the impact of each variable on the overall construct and on perceived usefulness, ease of use, and confidence. While our analysis did not find this impact to be significant, this may be caused by our operationalization of the construct. In the instance of highly customizable frameworks with distinct architectures used for different platforms, it is feasible that some of the features such as role inheritance, delegation, automatic documentation and configuration, etc. vary significantly in their implementation and meaning and are therefore less important for open-source framework users. This theory is also supported by the positive and negative events indicated by users, where some of the respondents identified that their frameworks used different architectures.

Therefore, we would recommend that future studies should first focus on a reflective measurement model to ascertain whether the framework's technical quality has a significant impact on perceived usefulness and ease of use. Assuming the impact is found to be significant, further studies should try to use a formative measurement model for the technical quality construct on only one type of framework architecture, or on architectures that are substantially similar.

Alternatively, future studies could use a qualitative methodology to determine what qualities users look for in an open-source framework. For example, researchers could start with a number of focus groups and/or interviews to determine the components of the framework technical quality construct. This research could then be followed by a quantitative analysis of the importance of each discovered quality component on continued framework usage intention.

It is important to note that the practical significance of a formative measurement model would be much higher than for a reflective one, as the formative model would help identify the impact of each quality characteristic on perceived usefulness, ease of use, and continued framework usage intention.

Another possible approach would be the use of a longitudinal study to analyze any changes in the user's perception of the framework's characteristics and the behavioral intention to use an open-source framework during the creation of an application. Furthermore, this longitudinal approach could also be used to analyze the relationships among the behavioral intention to use a system, actual use, net benefits, and satisfaction. By conducting this analysis, a researcher would be able to validate these IS success dimensions in the context of open-source frameworks and obtain objective measures of net benefits and actual use, which are not typically used in IS research (Benbasat & Barki, 2007).

Practical Implications

Having discussed the main findings of the study, it is important to also look at their practical implications. From a practitioner's perspective, our study can be used by framework developers and IT managers to evaluate or improve existing frameworks and to create better ones. Previous studies in the literature (Polančič et al., 2011; Polančič et al., 2010; Polančič et al., 2009) have found that users' perceptions of the usefulness of the framework and their intention to continue using the framework are two measures of a framework's success. By combining these findings with the ones from our study, we can summarize that these two measures of framework success are mainly influenced by the framework's flexibility, efficiency, and understandability, and by social pressure.

Therefore, these four antecedents can form the basis for evaluating an open-source framework by both existing and future users. Moreover, framework developers should focus on improving these characteristics to ensure the continued success of their frameworks.

In regards to social pressure, practitioners should be aware of the specific characteristics of open-source software and on the importance of the community in the continued success of open-source frameworks. Just as important, practitioners should note that our findings do not suggest any direct impact of the user's confidence in the framework on either perceived usefulness or continued framework usage intention. Therefore, it can be theorized that the main factor affecting continued usage intention is the user's perception of the usefulness of the framework. This pragmatic approach suggests that as long as the user finds the framework useful and is positively influenced by the community, he or she will continue using it.

Conclusion

In this study, we expand the taxonomy of open-source frameworks and analyze the impact of the framework's characteristics and social pressure on perceived usefulness and continued framework usage intention. This study builds on existing open-source framework research, while providing a number of theoretical and practical contributions.

First, we provide an in-depth analysis of key antecedents of continued framework usage intention, perceived usefulness, and ease of use. Our findings suggest that understandability, flexibility, efficiency, and social pressure are the main determinants of the original TAM constructs. This analysis provides a contribution to the existing TAM literature and also helps framework developers and IT managers understand factors that affect the user’s intention to continue using a framework.

Second, as suggested by Benbasat & Barki (2007), our study includes social pressure, a factor not yet researched in the open-source framework context, and analyzes its impact on the user's continued usage intention. The statistical analysis suggests that, together with perceived usefulness, social pressure has a significant impact on continued usage intention. We believe this to be a significant contribution to both the TAM and framework literatures, as it follows the recommendations of King & He (2006) for improving the TAM model.

Third, our study is the first to analyze the impact of the open-source framework's technical quality on perceived usefulness, ease of use, and continued framework usage intention. By using a formative measurement model, we intended to identify the impact of each of the elements on the combined construct. However, in our statistical analysis we failed to find sufficient evidence of construct validity and have therefore removed this construct from the revised model.

Fourth, while analyzing the antecedents of continued framework usage intention, our study provides a contribution to the literature on post-adoption usage models.

Fifth, our study provides several methodological refinements to previous framework studies. One example of this contribution is represented by our conceptualization and operationalization of the flexibility and portability constructs (Polančič et al., 2011).

From the practitioner's perspective, we seek to help framework developers understand what factors influence their users to continue using their framework and what factors influence their perception of the usefulness of the framework. In turn, this can be used by IT managers to evaluate frameworks and by developers to create better frameworks. Moreover, based on the results of previous studies (Polančič et al., 2011;

Polančič et al., 2010; Polančič et al., 2009), the user's perception of the usefulness of the framework and their intention to continue using the framework are measures of the success of the framework.

Given the fact that improving business productivity and cost reduction are the top IT management concerns (Luftman & Ben-Zvi, 2010), our study can be used as benchmark for determining whether a company should use a framework and whether their chosen framework is successful. In turn, our findings should help IT managers understand if the framework fits their needs and how it can be used for improving productivity and aligning IT and business.

References

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes, 50*(2), 179-211.

Batory, D., Cardone, R., & Smaragdakis, Y. (2000). *Object-oriented framework and product lines*. Paper presented at the Proceedings of the first conference on Software product lines: Experience and research directions, Denver, Colorado, United States.

Benbasat, I., & Barki, H. (2007). Quo vadis TAM? *Journal of the Association for Information Systems, 8*(4), 211-218.

Bhattacherjee, A. (2001). Understanding Information Systems Continuance: An Expectation-Confirmation Model. *MIS Quarterly, 25*(3), 351-370.

Boehm, B. (1999). Managing Software Productivity and Reuse. *Computer, 32*(9), 111.

Bosch, J., Molin, P., Mattsson, M., & Bengtsson, P. (2000). Object-oriented framework-based software development: problems and experiences. *ACM Computing Surveys, 32*(1es), 3.

Bryant, F. B., & Yarnold, P. R. (1995). *Reading and understanding multivariate statistics*. Washington, DC: American Psychological Association.

Chau, P. Y. K. (1996). An empirical investigation on factors affecting the acceptance of CASE by systems developers. *Information & Management, 30*(6), 269-280. doi: 10.1016/s0378-7206(96)01074-9

Davis, F. (1986). *A technology acceptance model for empirically testing new end-user information systems: Theory and results.* MIT.

Davis, F. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly, 13*(3), 319-340.

DeLone, W. H., & McLean, E. R. (2003). The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems, 19*(4), 9-30.

Dillman, D. A. (2006). *Mail and Internet Surveys: The Tailored Design Method 2007 Update with New Internet, Visual, and Mixed-Mode Guide* (2 ed.): John Wiley & Sons.

Dishaw, M. T., & Strong, D. M. (1999). Extending the technology acceptance model with task–technology fit constructs. *Information & Management, 36*(1), 9-21. doi: 10.1016/s0378-7206(98)00101-3

Drupal. (2011). Drupal Retrieved 19-05, 2011, from http://drupal.org/

Fishbein, M., & Ajzen, I. (1975). *Belief, attitude, intention, and behavior: An introduction to theory and research*: MA: Addison-Wesley.

Frakes, W. B., & Kang, K. (2005). Software Reuse Research: Status and Future. *Software Engineering, IEEE Transactions on, 31*(7), 529-536.

Freeze, R. D., & Raschke, R. L. (2007). An assessment of formative and reflective constructs in IS research. *15th European Conference on Information Systems, University of St. Gallen*, 1481-1492.

Goodhue, D. L., & Thompson, R. L. (1995). Task-Technology Fit and Individual Performance. *MIS Quarterly, 19*(2), 213-236.

Haenlein, M., & Kaplan, A. M. (2004). A beginner's guide to partial least squares analysis. *Understanding statistics, 3*(4), 283-297.

Hailpern, B., & Santhanam, P. (2002). Software debugging, testing, and verification. *IBM Systems Journal, 41*(1), 4-12. doi: 10.1147/sj.411.0004

Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2009). *Multivariate Data Analysis* (7th ed.): Prentice Hall.

Hardgrave, B. C., Davis, F. D., & Riemenschneider, C. K. (2003). Investigating Determinants of Software Developers' Intentions to Follow Methodologies. *Journal of Management Information Systems, 20*(1), 123-151.

Hong, S., Thong, J. Y. L., & Tam, K. Y. (2006). Understanding continued information technology usage behavior: A comparison of three models in the context of mobile internet. *Decision Support Systems, 42*(3), 1819-1834.

King, W. R., & He, J. (2006). A meta-analysis of the technology acceptance model. *Information & Management, 43*(6), 740-755. doi: 10.1016/j.im.2006.05.003

Klopping, I. M., & McKinney, E. (2004). Extending the technology acceptance model and the task-technology fit model to consumer e-commerce. *Information Technology Learning and Performance Journal, 22*(1), 35-48.

Lee, M.-C. (2010). Explaining and predicting users' continuance intention toward e-learning: An extension of the expectation-confirmation model. *Computers & Education, 54*(2), 506-516.

Luftman, J., & Ben-Zvi, T. (2010). Key Issues for IT Executives 2009: Difficult Economy's Impact on IT. *MIS Quarterly Executive, 9*(1), 49-59.

MacCallum, R. C., Widaman, K. F., Zhang, S., & Hong, S. (1999). Sample size in factor analysis. *Psychological Methods, 4*(1), 84-99. doi: 10.1037/1082-989X.4.1.84

Manolescu, D., Noble, J., & Voelter, M. (2006). Patterns for successful object-oriented framework development. *Pattern Languages of Program Design, 5*(1), 401–431.

McIlroy, M. D. (1968). *Mass produced software components*. Paper presented at the Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany.

Mockus, A., Fielding, R. T., & Herbsleb, J. (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309-346.

Moore, G. C., & Benbasat, I. (1991). Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation. *Information Systems Research, 2*(3), 192-222.

Parnas, D. (1976). On the Design and Development of Program Families. *IEEE Transactions on Software Engineering, 2*(1), 1.

Podsakoff, P. M., MacKenzie, S. B., Lee, J. Y., & Podsakoff, N. P. (2003). Common Method Biases in Behavioral Research: A Critical Review of the Literature and Recommended Remedies. *Journal of Applied Psychology, 88*(5), 879.

Polančič, G., Heričko, M., & Pavlič, L. (2011). Developers' perceptions of object-oriented frameworks – An investigation into the impact of technological and individual characteristics. *Computers in Human Behavior, 27*(2), 730-740.

Polančič, G., Heričko, M., & Rozman, I. (2010). An empirical examination of application frameworks success based on technology acceptance model. *Journal of Systems and Software, 83*(4), 574-584.

Polančič, G., Horvat, R. V., & Rozman, I. (2009). Improving Object-Oriented Frameworks by Considering the Characteristics of Constituent Elements. *Journal of Information Science and Engineering, 25*(4), 19.

Qualtrics. (2012). Qualtrics. Provo, UT: Qualtrics Labs Inc.

Raymond, E. (1999). The cathedral and the bazaar. *Knowledge, Technology & Policy, 12*(3), 23-49.

Roberts, N., & Thatcher, J. (2009). Conceptualizing and testing formative constructs: tutorial and annotated example. *SIGMIS Database, 40*(3), 9-39. doi: 10.1145/1592401.1592405

Schepers, J., & Wetzels, M. (2007). A meta-analysis of the technology acceptance model: Investigating subjective norm and moderation effects. *Information & Management, 44*(1), 90-103. doi: 10.1016/j.im.2006.10.007

Sindre, G., Conradi, R., & Karlsson, E.-A. (1995). The REBOOT approach to software reuse. *Journal of Systems and Software, 30*(3), 201-212.

SourceForge. (2011). SourceForge Retrieved 19-05, 2011, from http://sourceforge.net/search/?q=framework

Srinivasan, S. (1999). Design Patterns in Object-Oriented Frameworks. *Computer, 32*(2), 24-32. doi: 10.1109/2.745717

Trochim, W., & Donnelly, J. P. (2008). *Research methods knowledge base, 3rd edition*: Atomic Dog Publishing.

Van Antwerp, M., & Madey, G. (2008). *Advances in the SourceForge Research Data Archive (SRDA)*. Paper presented at the Fourth International Conference on Open Source Systems, IFIP 2.13 (WoPDaSD 2008), Milan, Italy.

van Gurp, J., & Bosch, J. (2001). Design, implementation and evolution of object oriented frameworks: Concepts and guidelines. *Software: Practice and Experience, 31*(3), 277-300.

Venkatesh, V., & Davis, F. D. (2000). A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management Science, 46*(2), 186-204.

Weber, S. (2004). *The Success of Open Source*: Harvard University Press.

Wikipedia. (2011). SourceForge Retrieved 12-12, 2011, from http://en.wikipedia.org/wiki/Sourceforge

Appendix A - Invitation E-mail

University of
Lethbridge

Faculty of Management

4401 University Drive
Lethbridge, Alberta, Canada
T1K 3M4

Phone 403.329.5148
Fax 403.329.2038

www.uleth.ca/man

Dear Participant:

My name is Alexandru Lemnaru and I am a graduate student conducting my thesis research for the University of Lethbridge Masters of Science in Management (Information Systems) program. I am sending you this e-mail to invite you to answer an online questionnaire for my study on Open-Source Frameworks.

Your participation will benefit yourself and the Open Source Community by improving understanding of Open Source Framework Usage. Furthermore, the results of this study can be used to improve framework and software development guidelines and practices. Please note that your participation is voluntary and that you can withdraw at any time. Answering the survey should only take 15-20 minutes of your time. As a thank you, you will have the chance to win a $500 Visa gift card.

To access the survey, please click on the following link:

Take the Survey

Or copy and paste the URL below into your Internet browser:
https://ulethmanagement.us.qualtrics.com/SE/?SID=SV_1Xt9ZlgpZ6P8645

If you would like to read more about our study, please visit the following link:

www.frameworkstudy.com

Thank you very much for taking the time to participate. It is greatly appreciated!

Alexandru Lemnaru
Master of Science Candidate

Appendix B - User Consent

Dear Participant:

Thank you very much for your interest in our study. It is greatly appreciated! The purpose of this research is to study the use of Open-Source Frameworks by Software Developers. This research requires about 15-20 minutes of your time. As a thank-you you will have the chance to win a $500 Visa gift card. The data in this survey are collected for research purposes and there are no anticipated risks related to this research.

Your participation will benefit you and the Open Source Community by improving understanding of Open Source Framework Usage. Furthermore, the results of this study can be used to improve framework and software development guidelines and practices.

As required by the University of Lethbridge Office of Research Services for all research involving human participants, this letter informs you of your rights as a participant in this research. Several measures will be used to protect your privacy and ensure confidentiality. All answers will remain confidential. Furthermore, we will not ask you for your name or organization. Additionally, you may choose not to give us your e-mail address, as we will only use it to send you the results of this study and to inform the winner of the draw.

The information collected in this survey will be reported only in an aggregated form. With the exception of any comments that you may provide in the final section, no individual answers shall be reported. In the event that we decide to use any comments that you make for research purposes, we will strip them of identifying information. The only people that will have access to this information are myself and my research committee. The responses of this questionnaire will be kept on a secure system and will be destroyed after five years. The results of this research will be presented in my Master's thesis. Additionally, they may be presented in academic or professional journals and / or in conferences.

Your participation in this study is completely voluntary. You may choose not to answer any question that makes you uncomfortable. Moreover, you are free to withdraw from the study at any time. To withdraw, just close the web page for the survey or navigate to another page. Significantly incomplete questionnaires will not be used in the research and will be deleted. There are no consequences for you in choosing not to participate in this research, or in withdrawing from this research. If you have any questions, feel free to contact me at alexandru.lemnaru@uleth.ca or my supervisor Dr. Brian Dobing at brian.dobing@uleth.ca. If you have any other questions regarding your rights as a participant in this research, you may contact the Office of Research Services at the University of Lethbridge at 403-329-2747.

The results of the study will also be available on the research's website **www.frameworkstudy.com** in early 2013, after the analysis is complete**.** Thank you very much for taking the time to participate. It is greatly appreciated!

Note: Please be aware that Qualtrics requires that JavaScript be enabled to answer this survey.

Alexandru Lemnaru
Master of Science Candidate

User Consent:
I understand my rights as a participant and I am willing to participate in this survey.

Appendix C - Questionnaire

1. General Information

INSTRUCTIONS**:**

For the purpose of this study we are interested in your experience with open-source frameworks. We define an open-source application framework as an **open-source software package that provides the functionality usually required to develop a software application**. This may include classes, interfaces, modules, or components that can be used in the application (van Gurp & Bosch, Software: Practice and Experience, 2001).

Examples of open-source application frameworks:

Spring Framework, Ruby on Rails, django, CakePHP, Zend, JUCE, etc.

1) How many different frameworks have you used? ______ frameworks

2) How many years of software development experience do you have? ______ years

When answering the following questions, please think about **your most recently completed application that was developed with an open-source framework.** Please think about only **one framework** and **one application.**

3) Please think about *your most recently completed application that was developed with an open-source framework*. **What is the name of the framework?** We are asking this question only for descriptive purposes and to help reinforce this connection throughout the survey.
   ____

4) What is the version of the selected framework? _________ Not Sure □

5) Please provide a general **label for the most recently completed application** with the framework and only refer to that application when responding to this survey. We are asking this question only for descriptive purposes and to help reinforce this connection throughout the survey.
   ____

6) How long ago was the application completed?

   1. 0-6 months □ 2. 7-12 months □ 3. 13-24 months □
   4. More than 24 months □

7) For what platform(s) has the application been completed? (check all that apply)

   1. PC □ 2. Web □ 3. Mobile □ 4. Other _____

8) For what industry was the application developed?

   Aerospace and Defense
   Communication, Entertainment, Media
   Consumer Product Manufacturing
   Educational Institution
   Financial Services
   Government Dept/Agency
   Health Care, Pharmaceutical
   Hospitality, Travel, Tourism
   Industrial Product Manufacturing
   Primary Producer (e.g., Mining, Oil & Gas, Forestry)
   Professional Services (e.g., Legal, Accounting)
   Retail Sales
   Software Development (for the software industry)
   Transportation
   Utilities, Pipelines
   Other / Please specify _________

9) When you started the project, would you have chosen to use this framework for this application?

   1. Yes □ 2. No □ 3. Not sure □

10) Is it necessary to extend the selected framework with components or interfaces to create an application with it? Please note that instantiation is not considered an extension of the framework.

   1. Yes (Whitebox Framework) □ 2. No (Blackbox Framework) □

11) Have you contributed modules, classes, components, interfaces, templates, etc. for the framework to other members of the open-source community?

   1. None □ 2. Occasional contributions □ 3. Significant Contributions □
   4. Core developer □

12) How much experience have you had with the selected framework? _____ person months

2. Framework Characteristics

The following statements are about your personal **beliefs** about the characteristics of **your selected framework** for the **project as a whole**. Please indicate your degree of agreement / disagreement with each statement by clicking the button that best reflects your answer.		**Strongly disagree**	**Disagree**	**Slightly disagree**	**Neither Agree nor Disagree**	**Slightly agree**	**Agree**	**Strongly agree**	**Don't Know**
PO1	The framework itself could have been easily installed on different *platforms* (i.e., PC, web, mobile).	1	2	3	4	5	6	7	
PO2	The framework itself and its extensions could have been easily transferred from one *platform* to another (i.e., PC, web, mobile).	1	2	3	4	5	6	7	
PO3	The framework itself could easily have been used on different *platforms* (i.e., PC, web, mobile).	1	2	3	4	5	6	7	
FL1	The framework was easily adapted/modified to fulfill my application requirements.	1	2	3	4	5	6	7	
FL2	The framework was easily extended to fulfill my application requirements.	1	2	3	4	5	6	7	
FL3	The framework was easily adapted/modified to create applications within the same domain.	1	2	3	4	5	6	7	
EF1	The framework did not require excessive system resources.	1	2	3	4	5	6	7	
EF2	The framework provided appropriate response times.	1	2	3	4	5	6	7	
EF3	The framework provided appropriate processing times.	1	2	3	4	5	6	7	
UD1	The framework documentation was accurate.	1	2	3	4	5	6	7	
UD2	All the functions of the framework were well documented.	1	2	3	4	5	6	7	
UD3	I could easily understand the framework without much reference to the documentation.	1	2	3	4	5	6	7	
UD4	I could easily understand the framework with the help of the documentation (if necessary).	1	2	3	4	5	6	7	
FS1	To the extent particular framework functions existed, they suited my	1	2	3	4	5	6	7	

	application requirements.								
FS2	The framework provided an overall set of functions that met the application requirements.	1	2	3	4	5	6	7	
FS3	The framework provided a suitable set of functions to build more modules.	1	2	3	4	5	6	7	

3. Framework Technical Quality

The following statements are about **your personal beliefs** about **your selected framework's** technical quality for the **project as a whole**. Please indicate your degree of agreement / disagreement with each statement by clicking the button that best reflects your answer. If you believe that the item was not important, please click on **Not Important**.		**Disagree**	**Slightly disagree**	**Neither Agree nor Disagree**	**Slightly agree**	**Agree**	**Not Important / Not Applicable**
FTQ1	Adding new functionality to the framework was **easy.**	1	2	3	4	5	
FTQ2	Updating the framework while retaining compatibility with previous instances of the framework was **easy.**	1	2	3	4	5	
FTQ3	Debugging the framework was **easy**.	1	2	3	4	5	
FTQ4	The framework was **easily** scalable based on application requirements.	1	2	3	4	5	
FTQ5	The framework interfaces **were** separated from its components.	1	2	3	4	5	
FTQ6	The framework interfaces **were** role oriented.	1	2	3	4	5	
FTQ7	The framework **used** role inheritance to pass down information.	1	2	3	4	5	
FTQ8	The framework **used** delegation to require specific functions from child components.	1	2	3	4	5	
FTQ9	The framework **used** small components instead of large ones.	1	2	3	4	5	
FTQ10	The framework **used** automatic configuration to help with its configuration.	1	2	3	4	5	
FTQ11	The framework **provided** automatic documentation for its functions.	1	2	3	4	5	

4. Individual

The following statements are about **your personal beliefs** about the characteristics of **your selected framework** for the **project as a whole**. Please indicate your degree of agreement / disagreement with each statement by clicking the button that best reflects your answer.		**Strongly disagree**	**Disagree**	**Slightly disagree**	**Neither Agree nor Disagree**	**Slightly agree**	**Agree**	**Strongly agree**	**Don't Know / Not Applicable**
SP1	People who influence my behavior thought I should use the framework.	1	2	3	4	5	6	7	
SP2	People who are important to me thought I should use the framework.	1	2	3	4	5	6	7	
SP3	Coworkers thought I should use the framework.	1	2	3	4	5	6	7	
SP4	People in the IT community in my area thought I should use the framework.	1	2	3	4	5	6	7	
SP5	People in the online community thought I should use the framework.	1	2	3	4	5	6	7	
CF1	I believed that the framework was mature.	1	2	3	4	5	6	7	
CF2	The framework rarely failed.	1	2	3	4	5	6	7	
CF3	The framework handled failures well if or when they occurred.	1	2	3	4	5	6	7	
PU1	I believe that using the framework increased my productivity.	1	2	3	4	5	6	7	
PU2	I believe that using the framework increased my effectiveness.	1	2	3	4	5	6	7	
PU3	Overall, I believed the framework was useful.	1	2	3	4	5	6	7	
EOU1	Learning to use the framework was easy for me.	1	2	3	4	5	6	7	
EOU2	I found it was easy for me to become skillful at using the framework.	1	2	3	4	5	6	7	
EOU3	Overall, the framework was easy to use.	1	2	3	4	5	6	7	
CFUI1	Assuming I were to develop the same project, I would still use this	1	2	3	4	5	6	7	

	framework.								
CFUI2	Assuming I were to develop other applications of this type, I would continue using the framework.	1	2	3	4	5	6	7	
CFUI3	Assuming I were to develop other applications in the same domain, I would continue using the framework.	1	2	3	4	5	6	7	
CFUI4	Assuming others were to develop the same project, I would recommend this framework to them.	1	2	3	4	5	6	7	

13) If you had to do it over again, would you use the same framework to develop the same project?

1. Yes □ 2. No □ 3. Not sure □

5. Descriptive Questions

14) How did you learn about this survey?

1. E-mail Invitation □ 2. Survey Website □ 3. Friend / Colleague □ 4. Search Engines □
5. Other / Please Specify ___________

15) What is the highest level of **IT education** you have completed?

1. No formal IT Education □ 2. Certificate / Diploma □ 3. Bachelor □ 4. Masters / Doctorate □ 5. Other ___________

6. Open Ended Questions

Thank you for responding to the other questions in the survey. Our final question to you is whether you can tell us about a **personal negative event** and about a **personal positive event** that happened while using the selected framework to develop the application?

16) Negative event:

____

____

17) Positive event:

____

____

18) If you have any additional **comments, suggestions or questions**, please use the **Comment Box** below. We will try to answer any questions if an email address is provided either here or at the end of the survey.

____

____

As indicated in the opening letter, you have the right to ask us to discard your answers.
Do you want us to discard your answers?
Warning: If you check the box, your answers will be discarded and shall not be used in the analysis.

Discard my answers:
□ I want to discard my answers.

If you want to be entered in the draw for the $500 Visa gift card, as well as to receive a copy with the results of this study, please write your e-mail address below.

The results of the study will also be available on the research's website **www.frameworkstudy.com** in early 2013, after the analysis is complete.

The winner will be randomly selected from all the respondents of this study and will receive his or her prize via mail. You do not have to answer all the questions to be entered into the draw. We require the e-mail address to be able to contact you if you are the winner of the gift card. We will not use your e-mail address for identifying you or for any other purposes without your consent.

Email address:
____

Appendix D - Instruments and Sources

Construct	Item	Source	Original	Modified
Portability	PO1	Sindre et al. (1995) Polančič (2011)	The framework can be installed on different environments	The framework itself could have been easily installed on different *platforms* (i.e., PC, web, mobile).
	PO2	Sindre et al. (1995) Polančič (2011)	The framework can be easily transferred from one environment to another	The framework itself and its extensions could have been easily transferred from one *platform* to another (i.e., PC, web, mobile).
	PO3	Sindre et al. (1995) Polančič (2011)	-	The framework itself could easily have been used on different *platforms* (i.e., PC, web, mobile).
Flexibility	FL1	Sindre et al. (1995) Polančič (2011)	The framework can be easily adapted or extended to fulfill application requirements	The framework was easily adapted to fulfill my application requirements.
	FL2	Sindre et al. (1995) Polančič (2011)	The framework can be easily adapted or extended to fulfill application requirements	The framework was easily extended to fulfill my application requirements.
	FL3	Sindre et al. (1995) Polančič (2011)	-	The framework was easily adapted to create my applications within the same domain.
Efficiency	EF1	Polančič (2011)	The framework requires too much of system resources	The framework did not require excessive system resources.
	EF2	Polančič (2011)	The framework provides appropriate response and processing times	The framework provided appropriate response times.
	EF3	Polančič (2011)	The framework provides appropriate response and processing times	The framework provided appropriate processing times.

Construct	Item	Source	Original	Modified
Understandability	UD1	Sindre et al. (1995) Polančič (2011)	The accessibility, level of detail and quality of framework documentation is good	The framework documentation was accurate.
	UD2	Sindre et al. (1995) Polančič (2011)	The accessibility, level of detail and quality of framework documentation is good	All the functions of the framework were well documented.
	UD3	Sindre et al. (1995) Polančič (2011)	-	I could easily understand the framework without much reference to the documentation.
	UD4	Sindre et al. (1995) Polančič (2011)	-	I could easily understand the framework with the help of the documentation (if necessary).
Framework Suitability	FS1	Polančič (2011)	The framework functions or services suited to application requirements in each individual case of its use	The framework functions suited my application requirements.
	FS2	Polančič (2011)	The framework provides suitable set of functions for my tasks and user objectives in each individual case of its use	The framework provided a suitable set of functions for the requirements of the application.
	FS3	Polančič (2011)	-	The framework provided a suitable set of functions for the need to build more modules.
Social Pressure	SP1	Hardgrave et al. (2003)	People who influence my behavior think I should use ADM	People who influence my behavior thought I should use the framework.

Construct	Item	Source	Original	Modified
	SP2	Hardgrave et al. (2003)	People who are important to me think I should use ADM	People who are important to me thought I should use the framework.
	SP3	Hardgrave et al. (2003)	Coworkers think 1 should use ADM	Coworkers thought I should use the framework.
	SP4	Hardgrave et al. (2003)	-	People in the IT community in my area thought I should use the framework.
	SP5	Hardgrave et al. (2003)	-	People in the online community thought I should use the framework.
Confidence	CF1	Sindre et al. (1995) Polančič (2011)	I believe that the framework is mature	I believed that the framework was mature.
	CF2	Sindre et al. (1995) Polančič (2011)	The framework fails frequently	The framework rarely failed.
	CF3	Sindre et al. (1995) Polančič (2011)	The framework handles failures well if or when they occur	The framework handled failures well if or when they occurred.
Perceived Usefulness	PU1	Moore and Benbasat (1991) Polančič (2011)	I believe that using the framework will further increase my productivity	I believe that using the framework increased my productivity.
	PU2	Moore and Benbasat (1991) Polančič (2011)	I believe that using the framework will further enhance my job effectiveness	I believe that using the framework increased my effectiveness.
	PU3	Moore and Benbasat (1991) Polančič (2011)	Overall, I believe the framework will be further useful in my job	Overall, I believed the framework was useful.

Construct	Item	Source	Original	Modified
Perceived Ease of Use	EOU1	Moore and Benbasat (1991) Polančič (2011)	Learning to operate the framework is easy for me	Learning to use the framework was easy for me.
	EOU2	Moore and Benbasat (1991) Polančič (2011)	I find it takes a lot of effort to become skillful at using the framework	I found it was easy for me to become skillful at using the framework.
	EOU3	Moore and Benbasat (1991) Polančič (2011)	Overall, I believe that the framework is easy to use	Overall, the framework was easy to use.
Continued Framework Usage Intention	CFUI1	Polančič et al. (2010, 2011)	I intend to increase my use of the framework in the future	Assuming I were to develop the same project, I would still use this framework.
	CFUI2	Polančič et al. (2010, 2011)	I intend to continue my use of the framework in the future	Assuming I were to develop other applications of this type, my intentions would be to continue using the framework.
	CFUI3	Polančič et al. (2010, 2011)	I am not going to use the framework in the future	Assuming I were to develop other applications in the same domain, I plan to continue using the framework for future projects.
	CFUI4	Polančič et al. (2010, 2011)	-	Assuming others were to develop the same project, I would recommend this framework to them.

Appendix E – Statistical Analysis – Initial Steps

Panel E.1. Missing Value Analysis.

Variable	N	Missing	
		Count	Percent
PO_1	176	13	6.9
PO_2	176	13	6.9
PO_3	176	13	6.9
FL_1	187	2	1.1
FL_2	186	3	1.6
FL_3	181	8	4.2
EF_1	183	6	3.2
EF_2	184	5	2.6
EF_3	183	6	3.2
UD_1	188	1	0.5
UD_2	188	1	0.5
UD_3	188	1	0.5
UD_4	186	3	1.6
FS_1	187	2	1.1
FS_2	186	3	1.6
FS_3	181	8	4.2
FTQ_1	184	5	2.6
FTQ_2	177	12	6.3
FTQ_3	185	4	2.1
FTQ_4	178	11	5.8
FTQ_5	180	9	4.8
FTQ_9	180	9	4.8
FTQ_10	176	13	6.9
FTQ_11	173	16	8.5
SP_1	177	12	6.3
SP_2	180	9	4.8
SP_3	175	14	7.4
SP_4	175	14	7.4
SP_5	174	15	7.9
CF_1	189	0	0
CF_2	187	2	1.1
CF_3	181	8	4.2
PU_1	186	3	1.6
PU_2	188	1	0.5
PU_3	188	1	0.5
EOU_1	187	2	1.1
EOU_2	189	0	0
EOU_3	189	0	0
CFUI_1	189	0	0
CFUI_2	189	0	0

Panel E.1. Missing Value Analysis.

Variable	N	Missing	
		Count	Percent
CFUI_3	189	0	0
CFUI_4	189	0	0
FTQ_6_E	143	46	24.3
FTQ_7_E	136	53	28.0
FTQ_8_E	157	32	16.9

Panel E.2. Descriptive Statistics for Respondents.

Variable	Values	Frequency	Percent
Respondent Type	Early	152	80%
	Late	37	20%
Level of IT Education	No formal IT Education	29	15%
	Certificate / Diploma	27	14%
	Bachelor	48	26%
	Masters / Doctorate	75	40%
	Other	10	5%
Software Development Experience (years)	1-5	12	6%
	6-10	38	20%
	11-15	65	34%
	16-20	43	23%
	21-25	14	8%
	26+	17	9%
Number of Frameworks Used	1	9	5%
	2	19	10%
	3-4	45	24%
	5-6	39	21%
	7-10	36	19%
	11+	41	21%
User Experience with Framework (person months)	1-3	25	13%
	4-6	22	12%
	7-12	25	13%
	13-24	34	18%
	25-48	39	21%
	49+	44	23%
Framework Type	Whitebox	113	60%
	Blackbox	76	40%

Panel E.2. Descriptive Statistics for Respondents.

Variable	Values	Frequency	Percent
Framework Usage	Voluntary	175	93%
	Mandatory	14	7%
User Framework Contribution Level	None	90	48%
	Occasional contributions	57	30%
	Significant contributions	8	4%
	Core developer	34	18%
How long ago was the application developed (months)	6	143	76%
	12	25	13%
	24	5	3%
	More	16	8%
Application Platform Note: Respondents were allowed to select more than one platform.	Web	120	63%
	PC	82	43%
	Mobile	29	15%
	Other	18	10%
Number of Platforms	Single-platform	141	75%
	Multi-platform	48	25%
Application Domain	Aerospace and Defense	2	1%
	Communication, Entertainment, Media	32	17%
	Consumer Product Manufacturing	4	2%
	Educational Institution	11	6%
	Financial Services	11	6%
	Government Dept/Agency	9	5%
	Health Care, Pharmaceutical	10	5%
	Hospitality, Travel, Tourism	2	1%
	Industrial Product Manufacturing	11	6%
	Primary Producer (e.g., Mining, Oil & Gas, Forestry)	4	2%
	Professional Services (e.g., Legal, Accounting)	7	4%
	Retail Sales	14	7%

Panel E.2. Descriptive Statistics for Respondents.

Variable	Values	Frequency	Percent
	Software Development (for the software industry)	31	16%
	Transportation	4	2%
	Other	37	20%
Number of Unique Frameworks		129	
Top 10 Frameworks Used	Spring	16	8%
	Qt	11	6%
	Zend	7	4%
	Django	6	3%
	Eclipse	6	3%
	CodeIgniter	3	2%
	Ruby on Rails	3	2%
	Drupal	2	1%
	Equinox OSGi	2	1%
	Grails	2	1%

Panel E.3. Skewness and Kurtosis.

Variable	Skewness		Kurtosis	
	Statistic	Standard Error	Statistic	Standard Error
PO_1	-0.687	.177	-0.745	.352
PO_2	-0.481	.177	-0.919	.352
PO_3	-0.685	.177	-0.779	.352
FL_1	-1.555	.177	2.139	.352
FL_2	-1.580	.177	2.004	.352
FL_3	-1.359	.177	1.543	.352
EF_1	-1.193	.177	0.916	.352
EF_2	-1.313	.177	0.959	.352
EF_3	-1.239	.177	1.314	.352
UD_1	-0.996	.177	0.300	.352
UD_2	-0.702	.177	-0.445	.352
UD_3	-0.581	.177	-0.644	.352
UD_4	-1.217	.177	0.940	.352
FS_1	-1.252	.177	1.280	.352
FS_2	-1.477	.177	2.704	.352
FS_3	-1.385	.177	1.792	.352
FTQ_1	-1.351	.177	1.089	.352

Panel E.3. Skewness and Kurtosis.

Variable	Skewness		Kurtosis	
	Statistic	Standard Error	Statistic	Standard Error
FTQ_2	-1.176	.177	0.813	.352
FTQ_3	-0.856	.177	0.041	.352
FTQ_4	-1.049	.177	0.423	.352
FTQ_5	-1.461	.177	1.321	.352
FTQ_9	-1.244	.177	1.244	.352
FTQ_10	-0.962	.177	0.015	.352
FTQ_11	-0.894	.177	-0.282	.352
SP_1	-0.649	.177	-0.174	.352
SP_2	-0.647	.177	0.224	.352
SP_3	-0.776	.177	0.241	.352
SP_4	-0.707	.177	0.478	.352
SP_5	-0.399	.177	-0.273	.352
CF_1	-1.766	.177	3.534	.352
CF_2	-1.542	.177	3.147	.352
CF_3	-0.870	.177	0.358	.352
PU_1	-2.003	.177	4.750	.352
PU_2	-2.202	.177	5.728	.352
PU_3	-2.278	.177	6.036	.352
EOU_1	-1.172	.177	1.079	.352
EOU_2	-1.310	.177	1.647	.352
EOU_3	-1.426	.177	2.197	.352
CFUI_1	-2.324	.177	5.380	.352
CFUI_2	-2.291	.177	5.547	.352
CFUI_3	-2.013	.177	4.348	.352
CFUI_4	-2.007	.177	4.237	.352

Appendix F – Statistical Analysis – Measurement Model

Panel F.1. Initial Measurement Model - Standardized Regression Weights (Factor Loadings).

Variable		Construct	Estimate
PO_3	←	PO	0.905
PO_2	←	PO	0.910
PO_1	←	PO	0.887
FL_3	←	FL	0.815
FL_2	←	FL	0.797
FL_1	←	FL	0.841
EF_3	←	EF	0.764
EF_2	←	EF	0.798
EF_1	←	EF	0.728
UD_3	←	UD	0.504
UD_2	←	UD	0.880
UD_1	←	UD	0.790
UD_4	←	UD	0.807
FS_3	←	FS	0.527
FS_2	←	FS	0.715
FS_1	←	FS	0.793
SP_3	←	SP	0.754
SP_2	←	SP	0.867
SP_1	←	SP	0.830
SP_4	←	SP	0.659
SP_5	←	SP	0.617
CF_1	←	CF	0.613
CF_2	←	CF	0.701
CF_3	←	CF	0.742
PU_1	←	PU	0.868
PU_2	←	PU	0.927
PU_3	←	PU	0.819
EOU_1	←	EOU	0.798
EOU_2	←	EOU	0.864
EOU_3	←	EOU	0.857
CFUI_1	←	CFUI	0.952
CFUI_2	←	CFUI	0.971
CFUI_3	←	CFUI	0.946
CFUI_4	←	CFUI	0.844

Panel F.2. Revised Measurement Model - Standardized Regression Weights (Factor Loadings).

Variable		Construct	Estimate
PO_3	←	PO	0.904
PO_2	←	PO	0.911
PO_1	←	PO	0.887
FL_3	←	FL	0.815
FL_2	←	FL	0.793
FL_1	←	FL	0.846
EF_3	←	EF	0.761
EF_2	←	EF	0.795
EF_1	←	EF	0.733
UD_2	←	UD	0.908
UD_1	←	UD	0.812
UD_4	←	UD	0.778
FS_2	←	FS	0.746
FS_1	←	FS	0.779
SP_5	←	SP	0.881
SP_4	←	SP	0.773
CF_2	←	CF	0.685
CF_3	←	CF	0.819
PU_1	←	PU	0.866
PU_2	←	PU	0.930
PU_3	←	PU	0.818
EOU_1	←	EOU	0.801
EOU_2	←	EOU	0.862
EOU_3	←	EOU	0.857
CFUI_1	←	CFUI	0.952
CFUI_2	←	CFUI	0.971
CFUI_3	←	CFUI	0.946
CFUI_4	←	CFUI	0.844

Appendix G – Statistical Analysis – Structural Model

Panel G.1. Hypothesized Structural Model - Regression Weights.

Relationship			Estimate	Standardized Estimate	P
PO	--->	EOU	-0.019	-0.032	0.605
FL	--->	EOU	0.435	0.467	***
UD	--->	EOU	0.475	0.523	***
UD	--->	CF	0.194	0.276	0.002
FS	--->	CF	0.749	0.580	***
EOU	--->	PU	0.227	0.263	0.001
CF	--->	PU	-0.022	-0.020	0.839
PO	--->	PU	-0.022	-0.041	0.460
FL	--->	PU	0.265	0.329	0.009
EF	--->	PU	0.169	0.169	0.048
FS	--->	PU	0.317	0.220	0.215
EOU	--->	CFUI	-0.039	-0.033	0.648
PU	--->	CFUI	1.028	0.755	***
CF	--->	CFUI	0.154	0.101	0.134
SP	--->	CFUI	0.122	0.114	0.034

*** $p < 0.001$

Panel G.2. Hypothesized Structural Model - Squared Multiple Correlations.

Construct	R^2
EOU	0.622
CF	0.505
PU	0.655
CFUI	0.669

Panel G.3. Revised Structural Model - Regression Weights.

Relationship			Estimate	Standardized Estimate	P
FL	→	EOU	0.432	0.461	***
UD	→	EOU	0.471	0.519	***
EOU	→	PU	0.203	0.235	0.002
FL	→	PU	0.360	0.446	***
EF	→	PU	0.260	0.259	0.002
PU	→	CFUI	1.077	0.789	***
UD	→	CF	0.197	0.278	0.002
FS	→	CF	0.725	0.567	***
SP	→	CFUI	0.126	0.116	0.028

*** $p < 0.001$

Panel G.4. Revised Structural Model - Squared Multiple Correlations.

Construct	**R^2**
EOU	0.626
PU	0.654
CFUI	0.669
CF	0.493