A Framework for the Evolution of Legacy Software
Towards Context-aware and Portable Mobile Computing Applications

A. Ahmad, I. Alseadoon, A. Alkhalil and K. Sultan
College of Computer Science and Engineering, University of Ha’il, Ha’il, Saudi Arabia
[a.abbasi | i.alsedon | a.alkalel | k.sultan]@uoh.edu.sa

Abstract - Mobile computing as a pervasive technology is fast replacing the traditional computing paradigms by supporting portable computations and context-aware communications. To date, there exist a large number of traditional computing software/systems – running on web or workstation platforms – that lack features of mobile computing such as portability, context-sensitivity, and high-interactivity. Software that executes on traditional computing platforms is referred to as legacy software that needs an upgrade to exploit the features of mobile technologies. However, legacy software may contain critical data and logic that cannot be easily replaced. One of the solutions is to evolve legacy software systems by (a) upgrading their functionality while (b) preserving their data and logic. We propose a solution named Legacy-to-Mobile framework that unifies the concepts of software reverse engineering and software change to support legacy evolution. We have used a case study based approach to evolve (a web-based) legacy software towards (context-aware and portable) mobile-based recommender application. Evaluation results suggest computation and energy efficient execution of the evolved application on resource-constrained mobile devices.

Keywords: Software Maintenance and Evolution, Legacy Software, Mobile Computing, Context-aware Systems.

1 Introduction

Mobile computing is fast replacing the traditional computing paradigms by empowering its users to exploit portability, context-sensitivity, and enhanced interactivity that are lacking in the traditional computing systems [1]. An increasing growth of mobile computing and rapid adoption of mobile technologies is due to the availability of embedded sensors (device’s hardware) that exploit freely available apps (device’s software) for communicating with remote servers (device’s connectivity) to perform a multitude of tasks on the go [2, 3]. As per the statistics of the Global System for Mobile Communications Association’s (GSMA’s) real-time tracker, currently there are more than 8.9 billion mobile device connections that represent approximately five times faster growth of mobile connections than human population [4]. However, to support a sustainable growth of mobile computing technologies, the traditional computing paradigms – software systems that run on workstation or web platforms – must also be modernized so they can execute on mobile platforms and utilize the features offered by mobile computing technologies [5, 6]. The software that runs on traditional computing systems is referred to legacy software that may be based on or developed with conventional or outdated technologies that need an upgrade in the context of mobile computing [7, 8]. Legacy software may contain critical data, logic, and processes that cannot be easily replaced due to their significance in supporting the operations of particular systems or domains [8, 13]. For example, an online product selling store contains all the product-specific data and logic with a web-based interface to view, select, and order the products. Such systems lack context-sensitivity (e.g., dynamic location, current distance) and portability (i.e., computation on the go) to offer customized recommendations based on contextualized information supported by mobile technologies. Software change or evolution1 refers to systematic modifications in the structure and behavior of the existing software as per the changes in system’s requirements and operational environments [9]. For example, the research in [8] presents a systematic review of solutions that support structural and behavioral evolution of legacy or existing software systems to cloud-based services that can be executed on cloud platforms. To modernize legacy software as mobile application, we aim to exploit the principle of software evolution to develop a process-driven and incremental evolution of the legacy software towards a portable, context-aware, and interactive mobile application. However, to support such evolution the primary challenge relates to exploiting a systematic reengineering approach (i) that upgrades software functionality but (ii) preserves the critical data, logic, and processes that cannot be altered to ensure correct functionality and quality of the software [9]. For example, when a traditional (workstation based) geographical information system is migrated to mobile platform(s), it contains same geographical data and relies on same algorithms but supports additional features of portability, interactivity.

1The terms evolution, migration and modernization are virtually synonymous and all refer to adaptive change [10]. We use the term evolution that refers to evolving existing software as per new requirements for upgraded functionality or quality of the software.

context-sensing, and location-awareness to capture and annotate precise geographical information at runtime [3]. Evolution of such legacy software to mobile platforms can be a daunting task due to many issues that include but not limited to platform compatibility, security and privacy along with restrictions due to resource-poverty in terms of processor, storage, and energy availability on mobile devices [5, 6, 24]. In recent years, the research and development on software evolution has primarily focused on the migration of existing or legacy systems to service-oriented and cloud-based systems [7, 8]. The state-of-the-art lacks any systematic approach that relies on reusable knowledge, processes, and tool support to enable legacy software evolution as a mobile computing application [6, 24].

Overview and Significance of proposed Solution – We propose to unify the concepts of software reverse engineering and software change to develop a framework that supports a systematic evolution of legacy software to mobile and provides a foundation for automation and tool support. We propose a framework named Legacy-to-Mobile that relies on (a) software reverse engineering to recover the artifacts (e.g., design/architecture) of legacy software and (b) software change management to upgrade the artifacts of legacy software as per the requirements of mobile platform. Figure 1 highlights that the proposed Legacy-to-Mobile solution framework comprises four processes namely (1) Planning the evolution (why to evolve?), (2) Modeling the artifacts of legacy software (what to evolve), (3) Transformation of the legacy artifacts to mobile application artifacts (how to evolve?), and (4) Evaluation of the evolved software to ensure desired functionality and quality in the context of resource-constrained mobile devices. We outline the contributions of the proposed research as below.

- Extending the software re-engineering and evolution models to develop a process-driven framework for legacy evolution to mobile computing that is lacking in the existing research.
- Case study based modernization of the legacy software to portable and context-sensitive mobile application and its performance evaluation on resource-constrained devices.
- Roundtrip process for legacy evolution to mobile that serves as a reference model to enable tool and human decision support for automation and customization of the evolution.

This paper is organized as follows. Related research is in Section 2. Proposed framework for legacy evolution is in Section 3. Case study based demonstration and evaluation of the framework are in Section 4. Section 5 concludes the paper.

2 Related Research

We present the existing research on legacy migration to service and cloud-based systems in Section 2.1 that helps us to position the needs for legacy evolution to mobile computing in Section 2.2.

2.1 Evolution of Legacy Systems to Modernized Platforms

In recent years, there has been a lot of research and development on the evolution of legacy software systems to modern computing platforms that mainly include service-oriented systems, cloud-based platforms, or software product lines [7, 8, 10]. Such evolution can support efficiency and economy of software operations. Most recently the US department of defense (US DoD) has been able to achieve IT specific process efficiency including enhanced software security, performance, and scalability by modernizing their legacy enterprise mail as cloud-based email [11]. Another significant example is the migration of oracle client/server software to cloud-driven data infrastructure [12].

- Legacy Evolution to SOAs. In the context of legacy software evolution, [7] presents a Systematic Literature Review (SLR) of the existing research on the methods and techniques used in legacy evolution to Service Oriented Architecture (SOA). The results of the SLR (i) establish an inventory of current research approaches, methods, and techniques used in legacy evolution and (ii) identify several dimensions for future research. In [17], the horseshoe model for an incremental evolution of legacy software to SOA is presented that provides a generic framework with processes, patterns, and tools for legacy evolution to SOAs.

- Legacy Migration to Cloud Computing Platforms. In [8], the authors have presented the Cloud Reference Migration Model (Cloud-RMM) that comprises three processes (i) Migration Planning, (ii) Migration Execution and (iii) Migration Evaluation along with a set of cross-cutting concerns for legacy migration to clouds. Some recent projects such as [11, 16, 18] have also demonstrated a successful migration of legacy systems to cloud.

2.2 Evolving Legacy Software to Mobile Computing Platforms

In contrast to the research and development on legacy software evolution to service-based and cloud-driven platforms [7, 8], there is much less research on legacy evolution to mobile. One of the pioneering works on legacy software evolution towards mobile computing platforms is presented in [19] to enable the migration of software applications from a resource rich work-station based computing environment to more self-reliant mobile computers. In the last decade and specifically beyond the year 2010, a rapid emergence and adoption of the mobile devices, affordable networking, and free software apps have made mobile computing an ideal platform adopted by legacy
software as part of software modernization. The primary motivations for legacy modernization as per mobile platforms are to achieve context-sensitivity and enhanced usability of the existing systems. For example, the research in [20] supports the migration of Windows based software running on a desktop machine towards a Palm OS based app for Personal Digital Assistant devices. A similar study [21] presents a solution that enables the modernization of user interfaces (UIs) of legacy software by evolving them to interactive and touch-based interfaces to enhance usability of legacy software. In terms of the partial tool support for legacy evolution, the only solution is presented in [22] to enable the migration of Java desktop applications to resource constrained mobile devices.

Summary of Related Work: We conclude that the existing research lacks any systematic solution(s) to enable legacy evolution to mobile. Our proposed solution is conceptually similar to [17, 18] that follow an incremental and process-driven approach for the evolution of legacy software to mobile computing platform. We go beyond the existing solutions [8, 18] and models [21, 22] to (a) develop a framework that supports legacy evolution to mobile, (b) demonstrate the applicability and usefulness of the framework with a case study, and (c) establish foundations for tool-support and human decision in the evolution process as highlighted in a recent mapping study [24].

3 Overview of the Proposed Framework

An overview of the proposed Legacy-to-Mobile framework is presented in Figure 2 that highlights a number of processes and activities for an incremental evolution. Incremental evolution refers to the decomposition of a coarse-grained evolution framework into a collection of fine-grained processes that can be executed incrementally [9], as below.

Framework Abstraction vs Framework Instantiation

Framework abstraction refers to a conceptual representation and organization of the (i) framework processes (i.e., what needs to be done?) and (ii) framework activities (i.e., how it is done?) [17] as in Figure 2 and Table 1. Table 1 provides information about individual processes, their underlying activities along with the incoming and out-coming data for each process. Framework instantiation refers to implementing abstract processes of the generic framework with case studies and to demonstrate the usage of the framework, detailed in next section (Figure 3).

3.1 Evolution Planning

The first process in legacy to mobile evolution, as presented in Figure 2 relates to the planning for evolution. This process aims to develop an evolution plan based on two primary activities. The first activity is to perform trade-off analysis about the motivations and challenges for the legacy evolution [8, 18]. The second activity identifies the level (i.e., source code, or software architecture or software configurations) at which the change can be implemented. The outcome of the process is an evolution plan to guide the further processes of the framework. The planning process highlights an important question: Why to evolve legacy software towards mobile computing application?

3.2 Legacy Modeling

The next process is legacy modeling (i.e., structural representation) of the legacy system that needs to be evolved. Modeling provides an overall view of the legacy software as a blueprint of the system to be evolved. The model can abstract the implementation specific complexities of the software with design or architectural representation of system for graphical (high-level) view of the software to analyze and transform the system at higher abstraction level [12]. Modeling process poses an important question: What level of software abstraction can be exploited to enable legacy evolution?

3.3 Legacy Transformation

The next process involves transformation of the legacy or source architecture to an evolved or target architecture [18]. The target architecture represents a high-level architectural representation in terms of the architectural components and connectors. Evolution from source to the target model is achieved by means of model transformation that refers to addition, removal, or modification of architectural elements to support architectural transformation highlighting an important question: How to evolve legacy architecture towards a desired architecture?
3.4 Application Evaluation

After legacy transformation, the last process involves evaluation of the evolved software. Based on the trade-off analysis, (i.e., Process I - Evolution Planning in Figure 2), the framework identifies some potential trade-offs for the evolved software. The evaluation process focuses on aspects like computation and energy efficiency of the evolved software on mobile devices to investigate: How to evaluate the efficiency and usability of the evolved software on (resource-constrained) target platform?

4 Case Study for Legacy Evolution

We now demonstrate the applicability of framework and the role of the framework’s processes and activities based on a case study. The case study also helps us to present the results of preliminary evaluation for computational efficiency of evolved software when it is transitioned from traditional platform (web/desktop system) to resource-constrained (handheld/mobile) platform.

- **Evolution of Web-based Product Searching Portal to a Portable and Context-aware Product Recommender Application:** ‘e-Bazaar’ (source system), is a web-based portal that allows user(s) to search the products of their interest (e.g., mobile phones, wrist watches, books) based on users’ preferences and search histories. In an era of portable computation and increased adoption of hand-held devices, the web portal inherits a limitation that relates to the lack of mobility and context-sensitivity of computations based on users’ context. The fundamental requirement of the case study is to evolve web-based product searching portal – while preserving its core logic and data – towards a portable and context-aware recommender system. Mobile-based application ensures portability and dynamic location information to automatically generate recommendations of the most relevant products based on users’ geographical position combined with preferences and search history. Moreover, mobile application can allow users to share the relevant recommended products with active contacts.

4.1 Process I – Evolution Planning

Planning as the first process of the framework helped us to formulate a plan and to streamline the activities or analysis before change implementation in the web portal (legacy software) to support its evolution towards context-aware mobile recommender [12]. Planning involves two main activities namely (i) performing trade-off analysis and (ii) identifying the level of software to execute the evolution as illustrated in Figure 3 and Table 1.

- **Process Activities:** The first activity of the planning process is to perform the trade-off analysis in terms of a systematic identification of the motivations and corresponding challenges for legacy evolution to the mobile [24]. For example, the primary motivations to evolve web-based portal to a mobile application are to achieve context-sensitivity of recommendations based on portable computations and enhanced user interactions. However, to realize the above mentioned benefits the primary challenges relate to resource poverty of mobile devices, concerns for data security and privacy along with limited display size of the hand-held devices. For example, evolution of the e-Bazaar web application as a mobile application helps users to exploit their location and portability to turn a passive product search to an interactive location-aware recommender system. In addition to the limited display size and computation resources (compared to desktop-based systems), location privacy is an open challenge for the evolved application. The second activity is related to identification of the appropriate abstraction level (i.e., design, architecture, code, configurations) to evolve the software. We have chosen architecture level to evolve the software due to the fact that architecture-centric evolution has been successfully utilized to support desired changes in existing software at higher abstraction levels [12, 18].

- **Process Income and Outcome.** There is no income to this process. The outcome is an evolution plan with benefits, challenges, limitations and the level of software for evolution.

- **Process Automation and Supervision.** Evolution planning is a manual process that requires human decision to guide further processes of the framework based on the derived plan.

|---------------------|--------------------|----------------|----------------|-------------------------------|-------------------------------------|
| Process I - Evolution Planning | - Trade-off Analysis  
- Identify Level of Evolution | None | Evolution Plan  
- Motivations and Challenges  
- Level of Evolution | No | Yes |
| Process II - Legacy Modeling | - Code Analysis  
- Architectural Representation | Level of Evolution | Legacy Architecture | Yes | Yes |
| Process III - Legacy Evolution | - Architectural Transformation  
- Component Preservation  
- Code Representation | Legacy Architecture | Evolved Architecture  
- Refactored Code | Yes | Yes |
| Process IV - Application Evaluation | - Qualitative Evaluation  
- Context Management | Evolved Architecture | Evaluation Results  
- Application Efficiency and Usability | No | Yes |
4.2 Process II – Legacy Modeling

Modeling the legacy software refers to visualization or high-level representation of the software to identify software parts that must be altered/preserved during evolution.

- Process Activities: In our case study, we have used the (PHP based) source code of the e-Bazzar web portal to represent its architecture. To represent the software architecture (i.e., UML component diagram) from web portal’s source code (PHP), we exploited an open source tool named PHP Extension and Application Repository (PEAR) [13]. PEAR helped us to represent the source code of the legacy software in terms of architectural components and connectors as in Figure 3 (see Process II – Legacy Modeling). By representing the legacy source code in terms of software architecture, we can abstract the implementation specific complexities and represent computational elements of the web portal as architectural components and connectors. For example, the architecture represents the ProductList component that coordinates with the UserPreferences component to generate the list of products based on user preferences. A simplistic mapping between the source code (in PHP) and its architecture (UML component diagram) is in Figure 3. The existing solutions on architecture-centric software evolution can be utilized to enable architectural evolution of legacy software [12, 15, 18].

- Process Income and Outcome. The income to the process is legacy system in terms of legacy source code. The outcome is legacy architecture that needs to be evolved as in Table 1.

4.3 Process III - Legacy Evolution

After modeling of the legacy software in terms of its architecture, the next process enables transformation of the legacy architecture towards desired architecture of mobile computing application.

- Process Activities: Architecture-centric software evolution is achieved by means of transforming the source architecture components towards the target architecture components as illustrated in Figure 3 (see Process III – Legacy Evolution). As illustrated in Figure 3, two new components named CurrentLocation and ContactList with their corresponding connectors are added in the architecture, while the connector between the components ProductList and UserPreferences has been removed. It is worth mentioning that during the architectural transformation, the existing components in legacy architecture, i.e., ProductList and UserPreferences are preserved in the target architecture as they represent the core logic to generate the list of products based on user preferences. The newly added components, i.e., CurrentLocation and ContactList allow mobile application to (i) automatically recommend the available products
and offers based on geographical proximity of the user, and
(ii) enable users to share the relevant products and offer to a
list of their contacts. The evolved architecture introduces
location and geographical context to automatically
recommend the elements of interests to the users based on
their location proximity and suggest user’s contacts who may
also be interested in similar products. The architecture is
mapped with the target source code that will be executed on
android devices. We have used UML Designer tool to
automatically generate the Java code from UML component
diagrams [14].

The tool only generated basic skeleton of the Java classes
while the fine-grained executable code has been written
manually for complete implementation of the architecture.
From a technical perspective, we have used our previous work
on double push out based graph transformation that support
preservation of desired components and transformation of
required components during architecture evolution [15].

- Process Income and Outcome. The income to the process is
legacy architecture that needs to be evolved. The outcome of
the process is evolved architecture and its source code that can
be executed on (Android-based) mobile devices as in Table 1.
- Process Automation and Supervision. Transformation of the
legacy architecture is a semi-automated process where
architecture is transformed (using graph transformation) [15]
and its code skeleton is generated using model transformation
[14]. Detailed code writing and its mapping is a manual
process with human intervention as in Table 1.

4.4 Process IV – Application Evaluation

The final process involves a qualitative evaluation of the
evolved software in terms of the usability and efficiency of the
mobile application on resource-constrained devices.

- Process Activities: From the efficiency point of view, we are
mainly concerned about efficient utilization of mobile
device’s (i) processor (computational efficiency), (ii) battery
(energy efficiency) and (iii) memory (storage efficiency) by
the application. With the availability of mobile cloud
computing, the data from mobile device can be offloaded to
cloud-based server that provides virtually unlimited storage
services via cloud storage [16]. Therefore, we discard the
evaluation\(^2\) of storage efficiency and primarily focus on the
computational and energy efficiency of the evolved software
on mobile platforms. To assess and evaluate the efficiency, we
monitored its memory and processor usage using CPU
Monitor [22] in Figure 4. a. An overview of the application’s
computation and energy efficiency monitoring is illustrated in
Figure 4. We observed that the proposed application took
approximately three seconds to fetch and display a list of
recommended products to the end user. During the trials,
mobile device’s processor consumption was between 1%
(min. consumption) to 4% (max. consumption) and the
memory use was 157 MB approx. Furthermore, to measure
battery consumption of the mobile application, we used Accu
Battery [23], as in Figure 4. b., the battery or power
consumption was between 0.2% (min. consumption) to 0.4%
(max. consumption). We conclude that, based on
experimental evaluations, the application is computational and
energy efficient on mobile devices as in Figure 4.

- Process Income and Outcome. The income to the process is
evolved application that needs a qualitative evaluation. The
outcome of the process is evaluation metrics to assess the
efficiency of the evolved application as in Table 1.

\(^2\) All evaluations were performed using Huawei P8 Lite and Samsung Galaxy
S6 phone both running Android – 5.0 Lollipop
- **Process Automation and Supervision.** Application evaluation is an automated process based on the tools to measure the computation and energy efficiency. Human decision is also required to benchmark and interpret the results for objective evaluation as in Table 1.

## 5 Conclusions and Future Research

Legacy software running on web or workstation based platforms can be modernized to execute on mobile computing platforms – benefiting from portable computations and context-aware communications – to perform tasks on the go. We have proposed Legacy-to-Mobile framework and demonstrated its applicability to evolve a web-based product searching portal to a context-aware and portable mobile recommender application. Preliminary evaluations highlight that the evolved application is computation/energy efficient on resource-poor mobile devices. The framework is a pioneering effort for modernizing legacy software as mobile application with following contributions.

- Enabling a process-driven and incremental evolution of legacy software to mobile computing application.
- Extending the principle and practices of software reengineering and software modernization in the context of mobile computing with tool-support for evolution.

**Dimensions of Future Research:** The first aspect relates to a comprehensive mechanism to support automation and appropriate user intervention to guide, customize, and execute legacy evolution. We plan to incorporate more case studies to objectivity evaluate the usability of the evolved software.

## 6 References


