Chapter 19
Fragmentation of Mobile Applications

Damith C. Rajapakse
National University of Singapore, Singapore

ABSTRACT

Fragmentation is a side effect of the high diversity of mobile devices. Some of such diversity is accidental (e.g., diversity caused by platform implementation bugs) and often can be eliminated with the use of better standardization. However, most of such diversity is essential (e.g., diversity of screen size) and the resultant fragmentation needs to be dealt with. Currently, there are many tools and techniques for de-fragmenting mobile applications (i.e., to reverse the effects of fragmentation and make the application work as expected on all target devices). The chapter gives an in-depth analysis of the fragmentation problem and the current state-of-the-practice in dealing with the problem. In particular, it illustrates how the current tools and techniques fit into nine basic approaches: MANUAL-MULTI, SELECTIVE, EMBED, INJECT, GENERATE, AIM-LOW, ABSTRACTION-LAYER, SELF-ADAPT, and DEVICE-ADAPT. The authors use a running example of a simple mobile application and a free de-fragmenting tool to demonstrate each approach. The reader of this chapter will gain an insight into the theoretical and practical issues related to fragmentation of mobile applications, the current state-of-the-practice in de-fragmenting, how the various de-fragmenting approaches relate to each other, and how they fit into a bigger picture.

INTRODUCTION

A major challenge in mobile application development is the inability to 'write once and run anywhere'. Developers often have to customize (or 'port') a mobile application to suit a multitude of diverse mobile devices. This increases the effort required in all aspects of application development, narrows the target market, and raises barriers-to-entry to the market. Practitioners call this the 'fragmentation problem'. *Fragmentation*, in the context of mobile applications, is the inability to "write once and run anywhere." Fragmentation is a widespread problem among mobile applications. Note that by "mobile applications" we mean *installed* applications on the mobile device and not the server-side applications such as SMS-based applications (server-side applications accessed

DOI: 10.4018/978-1-61520-655-1.ch019

using SMS messages) or mobile web applications (applications accessed over the Internet, using a web browser on a mobile device). When an application is fragmented, it shows unintended and undesirable behavior on some of the mobile devices. In other words, it shows "fragmented" behavior. For example, Figure 1 shows the same Calculator application running in two different phones. It shows fragmented behavior on the Device B as the full Calculator UI is not visible to the user.

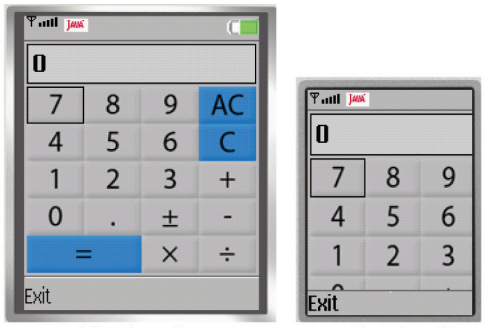
More formally, we define fragmentation as the "inability to develop an application against a reference operating context and to achieve the intended behavior in all operating contexts suitable for the application." Further, we define the *operating context* (OC) for an application as the "external environment that influences its operation." Therefore, an OC is defined by the hardware/software environment in the device, the user, and the environmental constraints introduced by various stakeholders such as the network operator.

CAUSES OF FRAGMENTATION

By definition, fragmentation is caused by the diversity of operating contexts (OCs). One operating context may differ from another for the following reasons:

- **Hardware diversity** of the device, such as differences in screen parameters (size, color depth, orientation, aspect ratio), memory size, processing power, input modes (keyboard, touch screen, etc.), additional hardware (camera, voice recorder etc.), and connectivity options (Bluetooth, IR, GPRS, etc.).
- **Software diversity**, which may be a result of platform diversity or implementation diversity:
  - **Platform diversity** is caused by factors such as differences in platforms/OS (Symbian, Nokia OS, RIM OS, Android, BREW, iOS, etc.), API standards (MIDP 1.0, MIDP 2.0, etc.), optional/proprietary APIs, variations in accessing hardware (e.g., full screen support), maximum binary size allowed, etc.
  - **Implementation diversity** is caused by factors such as quirks/bugs in implementing standards.
- **Feature variations**, such as light version versus full version.
- **User-preference diversity**, in aspects such as the language, style, etc., or accessibility requirements.
- **Environmental diversity**, such as diversity in the deployment infrastructure (e.g., branding by carrier, compatibility requirements of the carrier's back-end APIs, etc.), locale, local standards.

Figure 1. A simple example of fragmented behavior



Device A Device B

As we can see from the above, one OC can differ from another due to many factors. Let us call these factors *fragmentors*. i.e., a fragmentor is a factor, diversity of which causes fragmentation. The fragmentation of mobile applications is often referred to as *device* fragmentation, because most of the fragmentors can be traced to a particular device model. This is a misnomer however, as factors outside the device (e.g., branding by carrier) too can cause fragmentation. Figure 2 gives