



**KTH Computer Science  
and Communication**

# **A Market for the Smart Environment**

PAVEL PODKOPAJEV

Supervisor: Marc-Oliver Pahl  
Examiner: Vladimir Vlassov

TRITA xxx yyyy-nn



## Abstract

This thesis identifies tools and mechanisms of existing application markets by presenting features they offer and how they are realized. Design of a market for the smart environment is proposed, developed and evaluated. The design targets current issues of smart environments - no convenient ways to extend functionality of the environment, dealing with heterogeneous base and unattended application execution. Market called Agora is implemented according to a design. System is evaluated with custom made simulation, simulation procedures and components are developed.

The system designed in this thesis provides application provisioning, automatic application update mechanism, compatibility check with conflict situation suggestions, feedback mechanisms including explicit and implicit ratings, automatic malicious application removal from a market and the smart environments.

Evaluation of the system in various scenarios shows that proposed system design provides solution to a given problems of the smart environments. System is able to provide compatible applications to an end-user and automatically support developers by sending automatic crash reports. Implicit rating of applications and removing malicious applications from the system also provides helpful feedback to developers and market.

## Abstract

En typisk hem idag innehåller ett stort antal datorer, var och en med en dedikerad funktion. Dessa datorer är inbäddade inte bara i högteknologisk utrustning som smarta telefoner, iPods utan också de traditionella hushållsapparater, t.ex. spisar, tvättmaskiner och kylskåp. Även dessa apparater och anordningar är användbara som enskilda objekt kan deras användbarhet kan förbättras avsevärt genom att ansluta dem via ett nätverk för att möjliggöra kommunikation mellan enheter och apparater. Ett system där kylskåpet har förmåga att styra de viktigaste hemdator att beställa mjölk online från en butik, eftersom det slut påmjölk är mycket mer användbar än någon av apparaterna ensam.

Denna avhandling identifierar verktyg och mekanismer för befintliga programmarknader genom att presentera funktioner som de erbjuder och hur de förverkligas. Design av en marknad för smarta miljön föreslås utvecklas och utvärderas. Designen riktar aktuella frågor av smarta miljöer -. Inga praktiska sätt att utöka funktionaliteten av miljön, som handlar om heterogena bas och oönskad tillämpning utförande

Det system som utformats i avhandlingen ger ansökan provisionering, automatisk programuppdatering mekanism, kompatibilitet kontroll med förslag konfliktsituation, återkopplingsmekanismer inklusive explicita och implicita betyg, automatisk skadligt program tas bort från en marknad och de smarta miljöer. Utvärdering av systemet i olika scenarier visar att föreslagna systemet konstruktion ger lösning till ett givet problem med de smarta miljöer. Systemet kan ge kompatibla program till en slutanvändare och automatiskt stöd utvecklare genom att skicka automatiska felrapporter. Implicit rating av ansökningar och ta bort skadliga program från systemet ger också hjälp feedback till utvecklare och marknad.

**Keyword:** application, market, smart environment, appstore

# Contents

<b>Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>viii</b>
0.1 Acknowledgements . . . . .	1
<b>1 Introduction</b>	<b>3</b>
1.1 Background . . . . .	3
1.2 Objective . . . . .	4
1.3 Outline . . . . .	4
<b>I Analysis</b>	<b>5</b>
<b>2 Analysis</b>	<b>7</b>
2.1 Smart environment . . . . .	7
2.2 Application market . . . . .	9
2.2.1 General features provided by application markets . . . . .	14
2.3 How application markets revolutionized software deployment for smart phones . . . . .	20
2.4 Different Platform Strategies . . . . .	22
2.5 Users of a market . . . . .	26
2.5.1 Developers . . . . .	26
2.5.2 Administrators . . . . .	27
2.5.3 End-Users . . . . .	27
2.6 Information provided by market . . . . .	27
2.7 Technical features of market . . . . .	28
2.8 Distributed Smart Space Operating System . . . . .	29
2.9 Questions to be answered . . . . .	30
<b>3 Related work</b>	<b>33</b>
3.1 Explicit and Implicit Ratings of application . . . . .	33
3.2 Content recommendation . . . . .	34

3.3	Provisioning . . . . .	36
3.4	Update management . . . . .	38
3.5	Privacy . . . . .	40
3.6	Security . . . . .	41
3.7	Transparency . . . . .	42
<b>II Design</b>		<b>45</b>
<b>4</b>	<b>System Design</b>	<b>47</b>
4.1	Application and information storage . . . . .	48
4.2	Version control . . . . .	49
4.3	Signature/Integrity check . . . . .	49
4.4	Malicious applications removal . . . . .	49
4.5	Application recommendation . . . . .	50
4.6	Dependencies . . . . .	50
4.7	User authentication . . . . .	51
4.8	Provisioning . . . . .	51
4.9	Application control . . . . .	52
4.10	Top charts . . . . .	52
4.11	Rating . . . . .	53
	4.11.1 Explicit rating . . . . .	53
	4.11.2 Implicit rating . . . . .	53
4.12	Feedback . . . . .	54
	4.12.1 Crash report . . . . .	54
4.13	Transparency . . . . .	55
4.14	Private information filtering . . . . .	56
4.15	Compatibility . . . . .	56
<b>5</b>	<b>System Implementation</b>	<b>57</b>
5.1	System topology . . . . .	57
5.2	Global market . . . . .	58
5.3	Local manager . . . . .	58
5.4	Implicit rating . . . . .	58
5.5	Recommendation . . . . .	59
5.6	Malicious application removal . . . . .	59
5.7	Test environment . . . . .	59
<b>6</b>	<b>Test/Evaluation</b>	<b>61</b>
6.1	Evaluation . . . . .	61
	6.1.1 Rating . . . . .	61
	6.1.2 Crash reports . . . . .	61
	6.1.3 Malicious application removal . . . . .	62
	6.1.4 Update . . . . .	63

6.2 Summary . . . . .	64
<b>III Conclusion</b>	<b>65</b>
<b>7 Conclusion &amp; Future Work</b>	<b>67</b>
7.1 Conclusion . . . . .	67
7.2 Future Work . . . . .	67
<b>Bibliography</b>	<b>69</b>
<b>Appendices</b>	<b>75</b>
<b>Appendices</b>	<b>77</b>

## List of Figures

2.1 Users' place in smart environment [CDS04] . . . . .	7
2.2 Occupants and appliances of smart environment [CDS04] . . . . .	9
2.3 Apple App Store . . . . .	10
2.4 Items required for sharing apps with testers [app08a] . . . . .	11
2.5 General view of Google Play Market . . . . .	12
2.6 Google Play Market crash report manager for developers . . . . .	13
2.7 Google Play Market . . . . .	13
2.8 Amazon Appstore . . . . .	14
2.9 Nokia Ovi Store . . . . .	15
2.10 Google Play Market top lists screen . . . . .	16
2.11 Google Play Market application description . . . . .	17
2.12 Google Play Market application statistics . . . . .	18
2.13 Google Play Market application preview . . . . .	19
2.14 Enabler platform [Bal09] . . . . .	23
2.15 System integrator platform [Bal09] . . . . .	24
2.16 Neutral platform [Bal09] . . . . .	25
2.17 Broker platform [Bal09] . . . . .	25
2.18 Users of a Market . . . . .	26
2.19 ds2os layers [PNS <sup>+</sup> 09] . . . . .	29
2.20 Knowledge plane . . . . .	31

3.1	Google Play Market users rating review screen . . . . .	34
3.2	Google Play Market content recommendation . . . . .	36
3.3	Google Play Market remote provisioning guide screen . . . . .	37
3.4	Google Play Market application compatibility view . . . . .	38
3.5	Google Play Market application compatibility view with all devices available to user . . . . .	38
3.6	Update Manager Model [HKSS05] . . . . .	39
3.7	System Overview [KWKI03] . . . . .	40
3.8	Google Play Market permission screen . . . . .	43
4.1	Entities of the system . . . . .	47
4.2	User interaction with a system . . . . .	49
4.3	Provisioning . . . . .	51
4.4	Implicit rating . . . . .	54
4.5	Crash report . . . . .	55
5.1	Agora system topology . . . . .	57
5.2	Simulation topology . . . . .	60
6.1	Rating distribution without implicit rating feature [App08b] . . . . .	62
6.2	Automatic crash report contribution . . . . .	62
6.3	Bouncer strategy effect . . . . .	63
6.4	How fast all environments receive update after a new version is released . . . . .	64
.1	Version compatibility check before installation . . . . .	77
.2	Information presented to a user before installation . . . . .	77
.3	Before uninstalling an application user can mark it as malicious . . . . .	77
.4	Options available to a user after application is installed . . . . .	78
.5	Top application list . . . . .	79
.6	List of applications that are already installed in the environment . . . . .	80
.7	Agora login screen . . . . .	81

## List of Tables

2.1	Features provided by application markets that are relevant to the smart environment . . . . .	21
2.2	Platform strategies: comparison . . . . .	23



List of Tables	ix
4.1 Application control features and description . . . . .	52



## **0.1 Acknowledgements**

I would like to thank my supervisor, Marc-Oliver Pahl and examiner Vladimir Vlassov for the opportunity to work on this thesis, and for his guidance throughout the work.

I am grateful to the Technical University of Munich for providing research space and delicious food, and also to everyone that has provided feedback on my work. Most of all, I would like to thank my family for their massive support and for having faith in me.



# Chapter 1

## Introduction

### 1.1 Background

A typical home today contains a large number of computers, each with a dedicated function. These computers are embedded not only in high-tech equipment such as smart phones, iPods but also in the traditional household appliances such as cookers, washing machines, and fridges. Although these appliances and devices are useful as individual items, their utility can be greatly enhanced by connecting them through a network to allow communication between the devices and appliances. A system in which the fridge has the ability to command the main home computer to order milk online from a shop, because it is running low on milk is far more useful than any one of the appliances alone.

Application markets are one of the most important parts of mobile devices today. Apple App Store, Google Play Market, Amazon Appstore and Nokia Ovi are major application markets right now. Their main feature is to gather applications from developers and bring them to end-users in a convenient way. Applications that are installed from a market can not introduce new sensors or any kind of a new hardware to a device, but it can bring new features to its users by utilizing existent sensors that are already installed.

The collection of interconnected, different sensors in a building can be seen as a smart environment. Currently there are no application markets for such environments on a market (2012, August). Application markets can be useful for these environments, since they can bring new features to these environments by introducing new ways of utilizing sensors that smart environments are built from.

During the past years an abstraction over heterogeneous device zoo at home was developed (Section 2.8). This abstraction makes it possible to access many sensors and actuators in a standardized way which is the base for developing applications for home. From the end-user perspective, each new application adds value to the platform [ea11]. By introducing a market for the smart environment end-users of such environments would gain value by being able to extend their environments features without leaving it.

## 1.2 Objective

This thesis is about developing a market for the smart environment. After studying existing application markets for hand-held devices it was found that these application markets already provide some base functionality that can be used in smart environments. The objective of this thesis is to assess the needs of smart environment in order to propose a solution for an app store for smart environments that combining it with solutions that current application markets already provide:

- provides quality applications for end users in a convenient way
- provides necessary feedback for developers to create and support applications
- preserves privacy and security of its users
- is compatible with different environments by dealing with their heterogeneous base

and the main contribution of the thesis being identification of the needs of smart environments to extend their functionality by installing different control applications in a convenient way. Market called Agora is implemented according to a design. System is evaluated with custom made simulation, simulation procedures and components are developed.

## 1.3 Outline

This thesis is structured as follows:

First, in Chapter 2 smart environment and application market are defined, followed by an overview of tools and mechanisms of existing application markets.

Chapter 3 presents related work; research on possible solutions for problems defined in Section 2.9.

Chapter 4 describes the system design and Chapter 5 - implementation.

Chapter 6 provides an evaluation of the system.

Finally, in Chapter 7 conclusions and future work are given.

**Part I**

**Analysis**



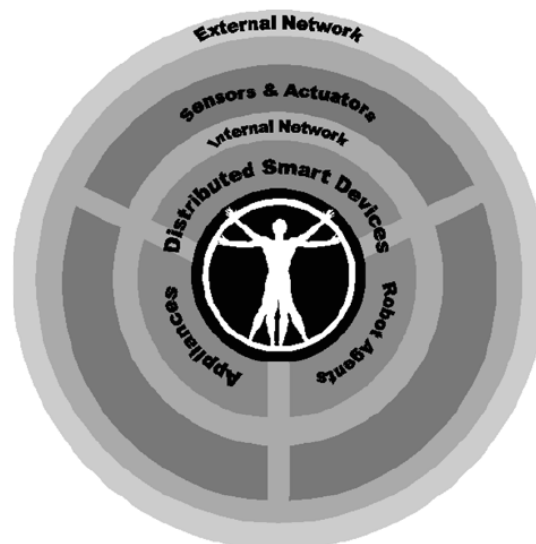


## Chapter 2

# Analysis

### 2.1 Smart environment

Smart environments are living spaces with embedded sensors that sense and effectors that react to the occupants [MRK09]. Such environments are designed to respond, anticipate and adapt to occupants' needs. Figure 2.1 shows users place in a smart environment. Goals of such environments are generally to maximize comfort and safety, optimize energy usage, enhance general well-being, and eliminate strenuous repetitive activities [MRK09]. Adaptation of environments refers to the fact that it learns to recognize and change itself depending on activities that are made by occupant, doing it by minimal intervention from occupant.



**Figure 2.1.** Users' place in smart environment [CDS04]

Miniaturization of devices and sensors, advances in network technologies have made

possible the smart environment. A typical home today contains a large number of computers, each with a dedicated function. These computers are embedded not only in high-tech equipment such as smart phones, iPods but also in the traditional household appliances such as cookers, washing machines, and fridges [MRK09]. Although these appliances and devices are useful as individual items, their utility can be greatly enhanced by connecting them through a network to allow communication between devices and appliances. A system in which the fridge has ability to command main home computer to order milk online from a shop, because it is running low on milk, is far more useful than any of appliances alone. Smart environments are typically contain embedded cameras and microphones. There is variety of sensors used in smart environments (Figure 2.2, to monitor ambient characteristics such as humidity or temperature in a room, to detect presents of human through motion or pressure sensors or bio-sensing to monitor health status.

However, if now users of such environments want to extend functionality of their environments, there is no convenient way to do it. In other words if user wants to introduce new cooperation between his/her alarm clock and coffee machine, he/she will have to wait until whole process of development will finish. So there will be no hot coffee in the morning just after he/she will be ready to have it.

So currently, users of smart environments cannot introduce new software, or update existing one in their environments or expand their environments and introduce new experience in a convenient way. Convenient way implies to introduce new functionality within a press of a button as this is happening right now in mobile device (smart phones) environments.

Mobile devices can also be seen as a smart environment. As it is described in further section 2.3, in the past, mobile devices had similar issues as smart environments currently face. Software provisioning in a convenient way was introduced to mobile phone environments, that created a whole ecosystem that is introduced with more details in section 2.4.

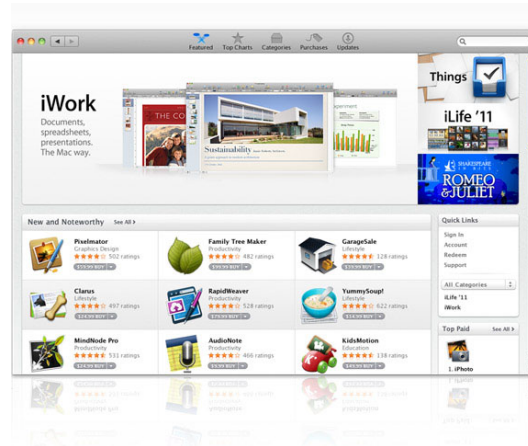
Mobile devices and smart environments are both collection of different sensors that can be controlled in a variety of ways to provide different outcomes. In mobile phones, there are sensors like GPS, Wi-Fi, cameras, microphones, gyroscopes, accelerometer and many others, which differ from manufacturers of specific devices. Same sensors can be found in smart environments.

Difference of smart environments and mobile devices is that appliances presented in houses are tending to be used for a longer time period than mobile device. Sensor management software is changed with a new phone model or with a new version of operating system that manages those devices. Sensors or devices deployed at home, do not have this opportunity to be updated, or requires expert user or administrator to perform it locally.

Since mobile phones environment already have solutions for similar issues to the ones that this thesis is focused on, analysis of such solutions is introduced. Further Application markets and their features are defined.



Apple devices, which makes this market the only place where users can get applications for their devices [cI08]. One of the main differences of Apple's



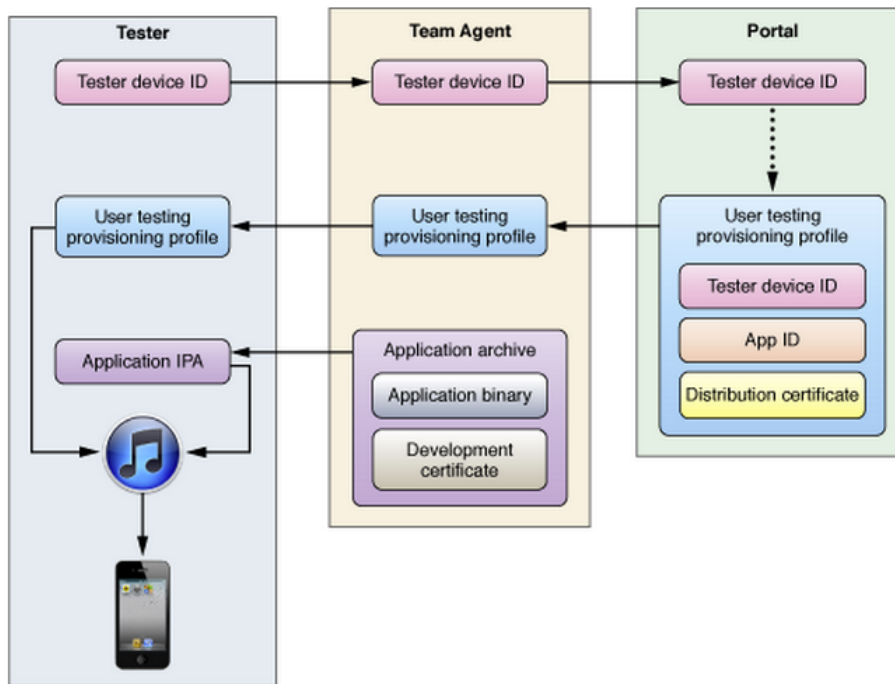
**Figure 2.3.** Apple App Store

App Store with its competitors is that Apple will approve only those items to the market, that will pass application screening phase. Appropriate team members inside the company will perform this phase. This increases quality of applications, since low quality applications are not going to be approved. But these tests increase time period before developed product reach the market by several weeks in comparison with other application markets [MLLC10]. Also it is impossible to see for a user, which sensors application is going to use during its usage, but developers of applications have to define this kind of information during development process. Only then they are allowed to use sensors or features of other applications, otherwise, application is not going to pass the screening phase [MLLC10].

To test developed application outside development team, developers of Apple App Store have to perform specific steps shown in Figure 2.4. After all required steps are passed, testers are able to use provided application. If application crashes during its usage, all crash reports are saved on a mobile device. When mobile device is connected to a computer, these crash reports will be saved on a computer. Later, testers are able to send this crash logs to developers by email [app08a].

## Google Play Market

Google Play Market (Figure 2.7) is an application market that is designed and developed by Google. All devices running Android operating system can use it, but it is not the only market that Android users can get their applications from. Amazon Appstore [Inc08a], GetJar [Inc08f], for example, are other opportunities to get applications for Android devices. After publishing



**Figure 2.4.** Items required for sharing apps with testers [app08a]

application to the market it can take several hours, before application will be accessible through the market [Inc08i], which is faster than in Apple App store case. This is happening since Google Play Market screening phase is done automatically, with no human interaction [Kam08]. The strongest part of Google Play Market compared to other marketplaces is that it has all the features that other application markets have to offer, but it also has features like video preview of applications if developers of that application have one, possibility to install application on your mobile device remotely, by simply going to the web page of a market. Also, Google Play Market provides elaborated information to its users. Sensors or other applications that are going to be involved by application usage are reported. This information is provided to users during install procedure [Inc08g].

If developers of Google Play Market want to test their application with third party users, they can simply send applications archive file for a test. Users that are going to install these archives on their device, have to approve installation of applications' from unknown sources on their devices. Otherwise application is not going to install. All crash reports will be gathered on a mobile phone where application is tested. Later this crash reports can be send to developers directly from a mobile phone. Developers can review reports in a convenient way presented in a Figure 2.6



Figure 2.5. General view of Google Play Market

## Amazon Appstore

Amazon Appstore for Android (Figure 2.8) lets their users instantly download games and mobile applications on their Android device. Currently, Amazon Appstore is available only to customers from United States. This application market is a third party application, which users can acquire from Amazon web page only, since this application is not available on Google Play Market. It can provide same features as Google's Play Market except for Auto-Update and Remote Install features. It is possible to buy applications through a web-browser and later download acquired applications from a market application, installed on a device. To install such market on a device, user will have to follow a step-by-step guide provided by Amazon, where he/she will have to let device to install applications from unknown sources that can harm security of a device [Inc08b].

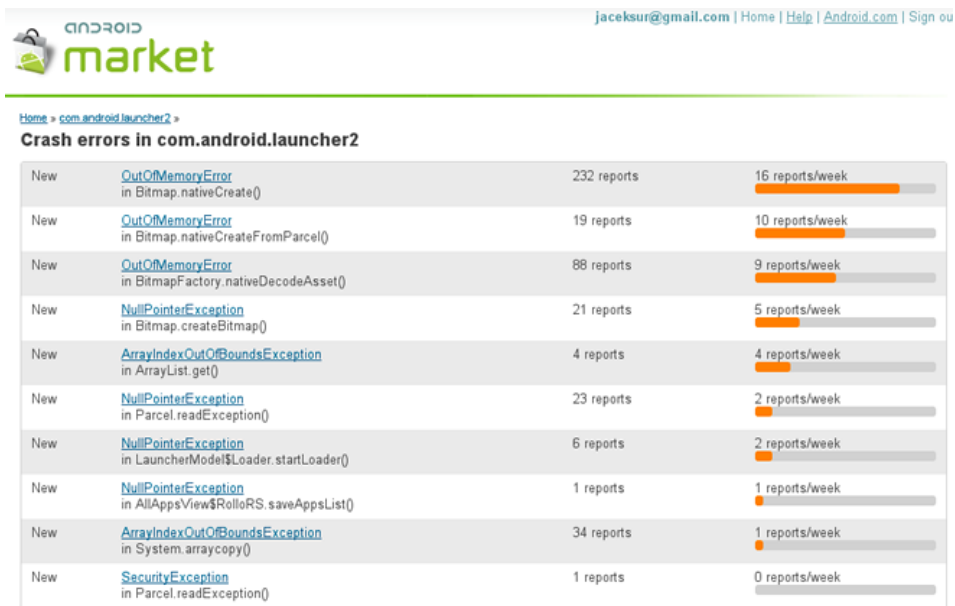


Figure 2.6. Google Play Market crash report manager for developers

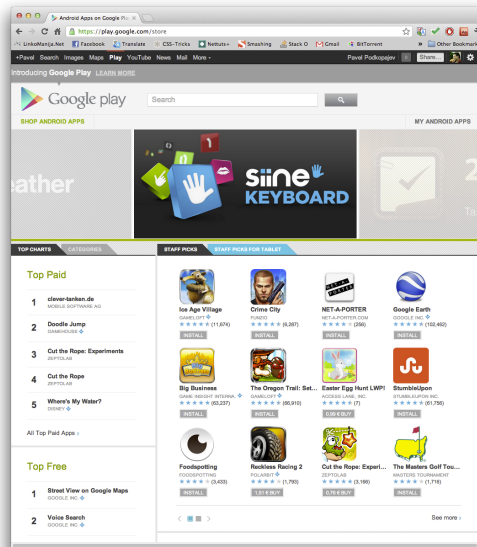


Figure 2.7. Google Play Market

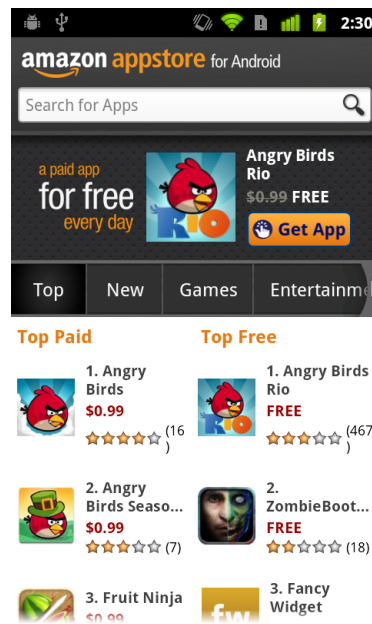


Figure 2.8. Amazon Appstore

## Nokia Ovi

Ovi store (Figure 2.9) is a marketplace, where users of Nokia Symbian or Java ME operating systems can get applications, audio and video files, ringtones for their mobile devices. This application market provides least features from all of shops that are compared (Table 2.1).

### 2.2.1 General features provided by application markets

General features of application markets that are relevant to this thesis topic are described further. Some of the features that are described are unique to specific markets that are highlighted in the description of the specific application markets. Some features of application markets are restricted to specific countries only; further list is based on Swedish, Lithuanian and German markets specifications. We describe only those features of application markets that are relevant to smart environments.

**Search** - allows users to search application market for specific applications by providing name of application they are looking for, or keywords that could describe that application. Result of this feature brings list of applications that match query. In some of application markets search query can be entered by text or voice. Search feature is one of the most important features of appli-



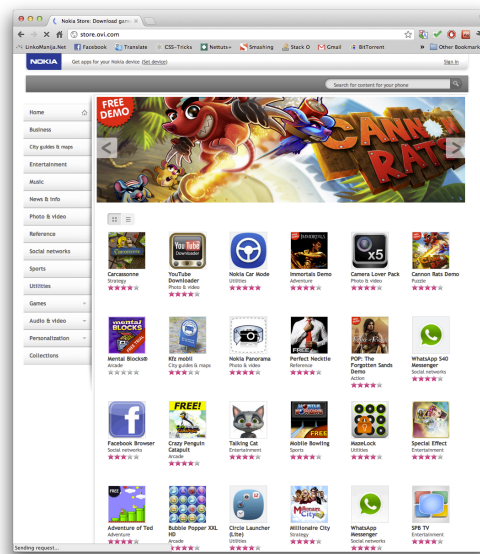


Figure 2.9. Nokia Ovi Store

cation markets, since otherwise it would be almost impossible to find desired application in market with more than 500 000 applications.

**Search with text** - allows users to enter query of search by text. User enters desired application name character by character.

**Search with voice** - allows users to type in a query for a search by voice. Whenever a user press search button of application market, he or she is presented with an option to say a query rather than just type it in. User is asked to say a query that is converted into a text that is used to search application database.

**Search results prediction** - as soon as user starts to type in a query for search, drop down list appears. It shows user different names of applications that exist in market, that coincide with user's query. This feature helps users to find applications even though they mistype name of it. Also it brings name of application before user finish to type it in.

**Search history** - whenever user comes to search for a new applications, he will be able to see his previous searches, if he or she will leave the search field empty, otherwise, application markets that have prediction of the search query feature will start to show predictions for the current query. This feature helps users to come back to the applications that they were searching later in time (in case users want to save their mobile traffic, and download the applications

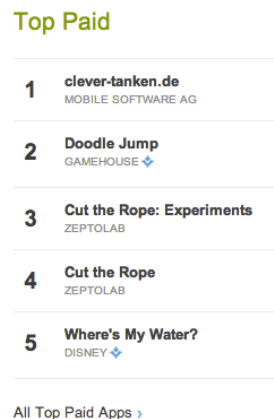
through WiFi network), without having to type them in again, or in case they forget the name, it will remind it to them.

**Featured apps** - every time application market is launched, the first thing that users see is this list. It provides users with a list of a high quality applications or information about applications that are currently on sale. Managers of application market choose these applications. It helps users to find new high quality applications, or will let them know about available discounts or special offers on the market.

**Top Applications** - provides users with different type of top applications: top free, top paid, top new paid, top new free. These lists are places where users will search for their new applications. Whenever user will get his new device, he or she will not search for an application using search feature, but they will start browsing the lists of top applications.

**Top Paid apps** - list of most popular paid applications (Figure 2.10) on the market. Users can see what other users are paying for and decide on their purchase easier, since the fact that so many users paid for product shows that the product really worth it, otherwise users would not buy it.

**Top Free apps** - list of most popular free application on market. Helps new users get free applications that most of users of market are using.



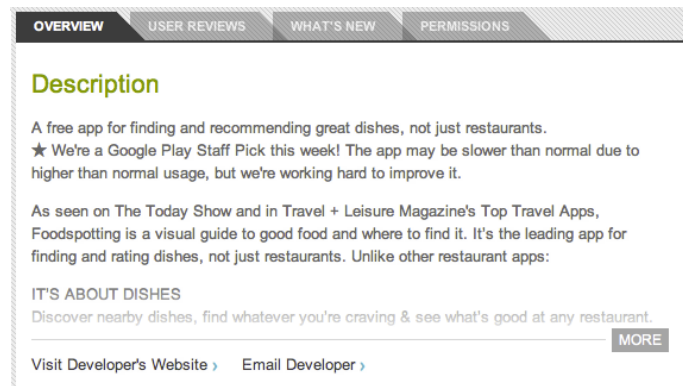
**Figure 2.10.** Google Play Market top lists screen

**Top New apps** - this list provides information about new paid/free applications on market, that immediately got attention by other users of market. It helps new high quality applications to get to their users faster. This is a new feature of application markets. Previously there were only "Top Paid Applications", where users could see old applications that got user support, that would updated slowly, since it would be hard for a new applications to compete with

the applications that already have lots of users. This feature, fixes this problem, by giving new applications their own market, where they can compete with equal competitors.

**New apps** - list of applications that provide users with the items that were posted to a market recently. It helps new applications to get to their users; otherwise chance that they would be lost (no one would find them) would be higher. Having this feature in application market will motivate developers, since with this feature they have more chances to reach customers.

**Description** - provide users with general idea of features that application is suppose to do (Figure 2.11). It also gives users information about current version of an application; when it was last updated, what is the size of it and the maturity level.



**Figure 2.11.** Google Play Market application description

**Feedback** - after installing of application, users can share their experience with this application. Also this experience can be read by other users before installing application to learn more about item he or she is about to install. This feature makes developers to support their applications, and react on problems of users as soon as possible.

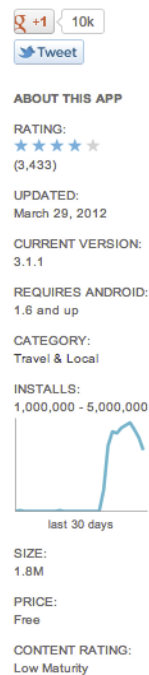
**Rating of the feedback** - users can rate feedback note of application as positive or negative. Feedback with higher rating will be shown higher than the one with lower rating; this feature helps users to read most appropriate feedback notes first skipping the ones with low rate.

**Rating of the application** - allows users to rate applications. Item can be given one to five stars rating, that later will be summed up and presented to users as an average. Developers can react accordingly, if they see that users are not satisfied with their current solutions, and users can decide on which application to download easier.

**Share markets item** - users can share application they are about to install or are already using with their friends, through email client or any other application that supports sharing (Facebook, Evernote, Dropbox etc.). It will give users URL that point to the item that was shared with the description and the icon of the application.

**Information on how to reach developers** - provides users with information on how to reach developers of current application. Webpage, email address or phone number could be used as a reference. This feature is helpful if someone would like to get support for the application, or even if someone would like to order an application for him or her self.

**Download statistics** - provides information for users of a market of different kind of data (Figure 2.12). How many people shared application with others. How many users vote on a specific application. When it was updated last time. Also, users can see number of users that installed this application in last 30 days.



**Figure 2.12.** Google Play Market application statistics

**Preview** - allows users to see some screenshots of application (Figure 2.13). At least two of them are required, if user wants to publish applications to a market. Preview feature provides more feeling to customers about an application.

### App Screenshots



Figure 2.13. Google Play Market application preview

**Video preview** - allows users to see application in action. Developers can record promotional video of the application, specifically, how it is operated, or how described functionality is achieved. This provides even more feeling of application than flat images.

**Update** - allows users to update already existent applications on their devices. By executing update function of a market, application is downloaded, and previous version is exchanged with a current one. Usually update functionality saves users data and moves to a new version. If this is not the case, developers inform their users about this issue a priori.

**Auto-Update** - application market can automatically update application that are installed on a device or notify user that there is an update pending for some applications. It keeps user up to date and saves his or her time on searching for updates manually.

**Related apps viewed** - list of applications that are relevant to the one that user is looking at right now. It provides user with applications that were viewed by other users when they were searching for same application. Usually this list is presented at the bottom, after presenting current application. List is aligned horizontally and will show to user only one row of information with possibility to see whole list in a new window. This list will help users to find new solutions to their problems and help developer to promote their applications.

**Related apps installed** - list of applications that are relevant to the one that user is looking at right now. It provides to the user, applications that were installed by other users, when they were searching for the same application type. Applications are made by other developers but can perform the same tasks. It helps user to compare the application he or she is about to install with other applications on a market and find best one for his needs. Usually this list is presented at the bottom, after presentation of current application. The list is aligned horizontally and will show to user only one row of information with possibility to see the whole list in a new window.

**Remote Install** - allows users to install applications from a remote location. Users can use their web browsers, to deploy applications on their mobile devices. Mobile device is recognized by a username, given on a first start of a device. Later, using the same username, users can deploy applications remotely.

**Mark as inappropriate** - gives user an option to report application as inappropriate one. If after reading the descriptions and installing the application user discovers that it has offensive content that was not described in the description field, or it will not work on a device that is actually on the list of the supported applications, he can report this application to a market team, to take required measures.

**Backup** - allows to a user to save his or her application on remote server, so when user will change his mobile device or reinstall the current one, it will save him a lot of time, since all the application will be restored automatically. Also, it can save user a space on the mobile device that application is using, since it can be removed from the device and restored back, with no need of paying for it again.

Different features of four application markets are presented here. Further, these features are used to design a market for the smart environment.

## 2.3 How application markets revolutionized software deployment for smart phones

Before application markets hit market, users were searching for applications by using general search engines like Google, Yahoo, Bing etc. or getting them directly from developers' web pages. After applications were acquired, users were moving executable or archived files to their device manually. When transfer was complete, user was allowed to install transferred files by locating it in a directory where it was saved [Inc08j].

Packet managers in operating systems like Linux, Mac or Windows have functionality of application markets. For example, if Linux user wants to install a packet (application) that has some prerequisites (dependencies from other packets (applications)), he/she can perform it by using APT manager. User can simply request needed packet by sending "apt-get install name-of-package" command to a packet manager. If requested packet requires other packets that are not present in the system, packet manager will automatically install required dependencies and updates [Lin08]. These features of packet managers are relevant to the question arrived earlier - How to bring and manage software in smart environments in a convenient for its users way.

Currently, Apple App Store, Google Play Market, Amazon Appstore and Nokia Ovi

Features	Apple App Store	Google Play Market	Amazon Appstore	Nokia Ovi
Search	√	√	√	√
Search with voice		√		
Search results prediction		√		
Search history	√	√		
Related Apps	√	√	√	
Featured Apps	√	√	√	√
Top Paid apps	√	√	√	
Top Free apps	√	√	√	
Top New Paid apps		√		
Top New Free apps		√		
Top Grossing	√	√		
New apps	√	√	√	
Related apps viewed		√	√	
Related apps installed	√	√	√	
Description	√	√	√	√
Feedback	√	√	√	√
Rating of the feedback		√	√	
Rating of the application	√	√	√	√
Share markets item	√	√		
How to reach developers	√	√	√	
Download statistics	√	√	√	
Preview	√	√	√	√
Video Preview		√	√	
Auto-Update	√	√		
Remote Install	√	√		
Backup	√	√	√	
Mark as inappropriate		√		

**Table 2.1.** Features provided by application markets that are relevant to the smart environment

are major application markets on a market for mobile devices. Main task of such markets, is to bring quality applications in one place to their users in a convenient way.

By using facilities of these application markets, user is allowed to search, install and use application from one single place. In general, these application markets are built into operating systems they are coming with, for example Apple App Store could be found only on the Apple iOS devices, Google Play comes with Android operating system only, and Nokia Ovi provides applications for their own platform only. Even though all of these markets come from different vendors, basic functionality is same for all of them: user is provided with a market, that helps to find necessary applications, information about those applications and a convenient way to make these applications available on a device. After following instructions of application market, user is able to use applications after an installation process.

In the following section different platform strategies that can be used to build a market are defined. By looking at these platform strategies different user roles and topologies of a market are obtained and used in a design section.

## 2.4 Different Platform Strategies

In the mobile service domain, different (i.e. respectively hardware based, operating system-based, web-application-based, and portal based) initiatives such as Apple's App Store, Nokia's OVI, Google's Google Play Market and Windows Phone exist. All these initiatives can be referred to as platforms, since they usually mediate between different sides of a market (e.g. developers, advertisers and customers). Four different types of platforms, oriented around their different control over the customers relationship and tangible and intangible assets that make up the value proposition, were suggested [VG10].

By examining these platform types, users of application markets are defined and further used in a design section.

The first type of platform can be labeled an "Enabler Platform". In this platform type owner controls many or most of the assets involved in service provisioning, but the customer relationship is done by the third-party developers.

The second type of platform can be labeled "System Integrator Platform". This platform represents the case where most of the assets that are related to the value proposition, customer ownership, are hold in the hands of the platform owner. This actor - owner will also facilitate entry of the "third parties" to constitute a multi-side market. In other words, it will allow other service providers to use its platform to increase the value of the platform (e.g. Apple App Store, Nokia Ovi).

The third type of platform can be labeled a "Neutral Platform". This platform represents the case in which the platform owner does not control most of the assets that are used for the value proposition and does not have customer ownership, since it does not establish a billing relationship with an end-user (e.g. PayPal).

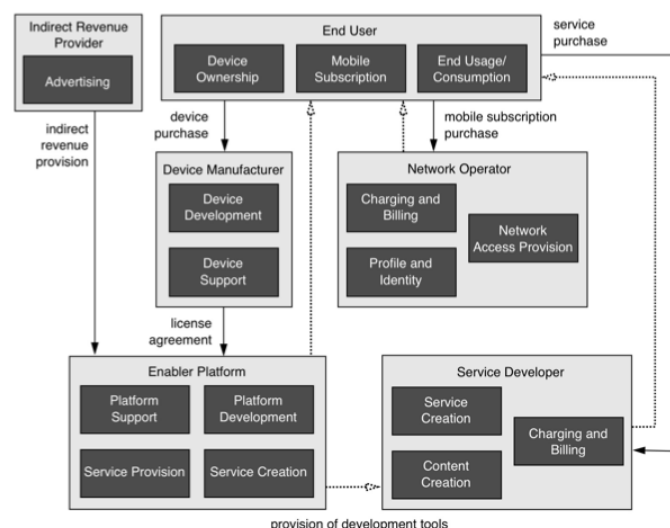


	No Control over Customers	Control over Customers
Control over Assets	<b>Enabler Platform</b>	<b>System Integrator Platform</b>
No Control over Assets	<b>Neutral Platform</b>	<b>Broker Platform</b>

**Table 2.2.** Platform strategies: comparison

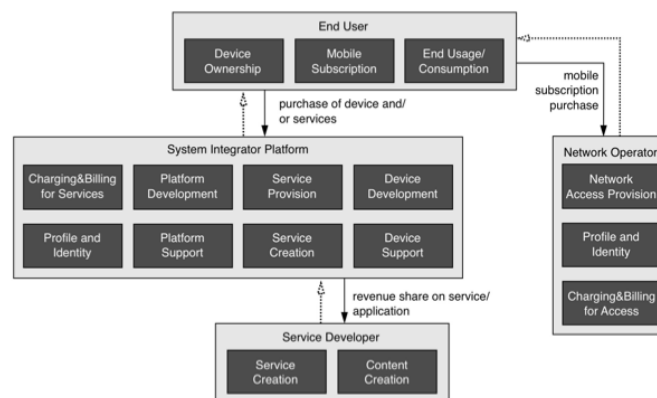
The fourth type of the platform can be labeled a "Broker Platform". Here, platform relies on other actors that control most of the assets to establish the value proposition, but it integrates customer ownership (e.g. Ebay, GetJar).

**Enabler Platform** - it controls the most important assets that brings value. Mobile operating systems are typical examples of enabler platform. Owner of the platform has the knowledge and experience to develop attractive and successful platform for both, developers and end-users. However, there is no direct exchange of money between the end-user and the platform owner that means there is no customer relationship with the end-user. For the success of such platform, owner will have to be constantly aware of what customers need, provide rich API and SDK for the third-party developers to facilitate them to create services that customers need. The drawback of such platform is that since there is no relationship between platform operator and end-user, it is hard for the platform operators to keep developers motivated [VG10]. The overview of this platform is presented in Figure 2.14



**Figure 2.14.** Enabler platform [Bal09]

**System Integrator Platform** - it controls value-adding assets in creating of software and hardware, and also controls customers, by having a direct charging and billing. Platform owner will provide device configurations and software API for developers, will bill all end-users and share this money to cover platform expenses and pay developers. Success factors are similar to the ones from enabler platform, where platform owner should attract developers to adapt his platform by having a market of users and developer community, transition from development to commercialization. The overview of this platform is presented in Figure 2.15



**Figure 2.15.** System integrator platform [Bal09]

**Neutral Platform** - provides the basic set of tools but do not assists in creating the final service for the end-user, thus not controlling the most important value-adding assets. There is no financial customer relationship. This type of platform is used to create collaboration environment between large companies. Participant should know exactly how joining a consortium would benefit them. Not having a strong founding group or weak support for a neutral platform can quickly lead to failure, but extremely useful for working together on a common standard [VG10]. The overview of this platform is presented in Figure 2.16

**Broker Platform** - does not control assets that make up the value proposition, but have a strong customer relationship since the end-users pay for service directly through the broker platform. In most cases a revenue sharing between broker and service developers is set up. Since broker platform can provide services to different kind of OSs and operators, it has to provide support for diverging devices and OSs and develop technological backend that will make service provisioning easier for customers (e.g. filtering content based on the handset model) [VG10]. The overview of this platform is presented in Figure 2.17

By looking at these (Table 2.4) four different platform strategies, three different types of users can be distinguished (Figure 2.18): Developers who develop applica-

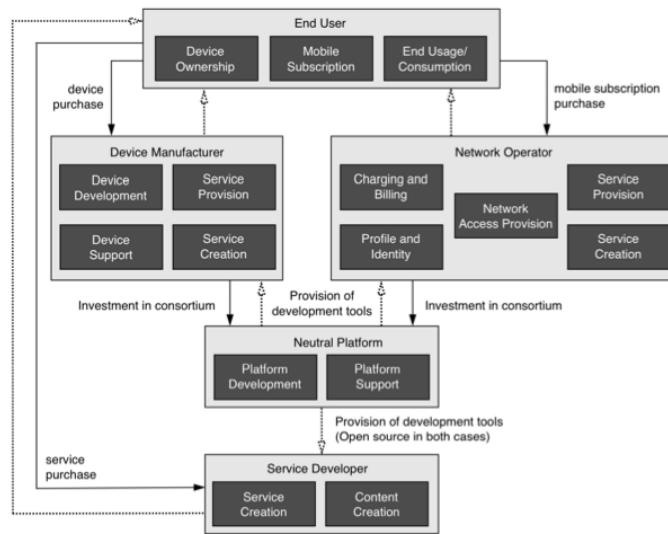


Figure 2.16. Neutral platform [Bal09]

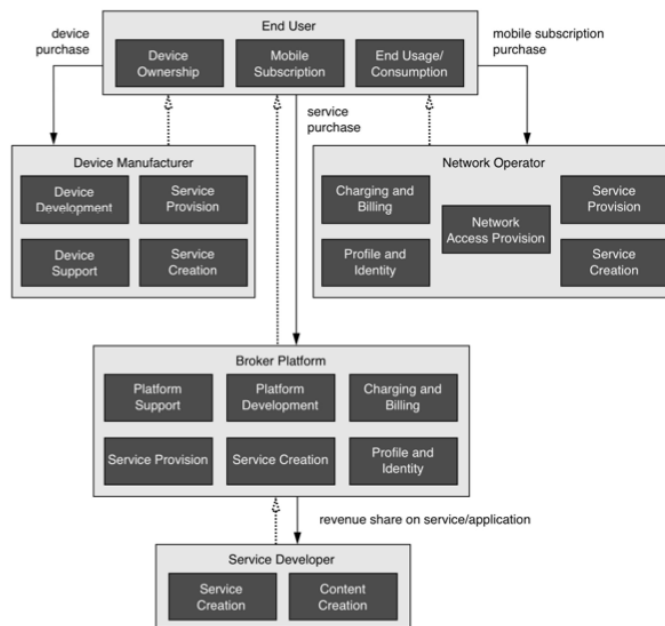


Figure 2.17. Broker platform [Bal09]

tions, Administrators who deploy those applications and End-Users who use them.

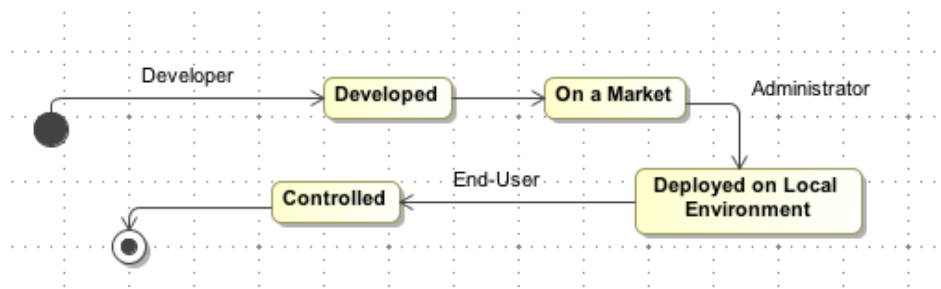


Figure 2.18. Users of a Market

## 2.5 Users of a market

As described in section 2.4, three basic types of users of smart environments can be distinguished: developers, administrators and end-users. Users are important part of a market since they are producers and consumers of what market have to offer. Even though three different types of users are defined, it is possible that same user acts as developer, administrator or end-user, if enough knowledge acquired to perform required actions of specific user type. To support its users, market has to provide basic functionality for each type of users as it was discovered in Section 2.2 and Section 2.4.

### 2.5.1 Developers

Developers of market populate markets with applications and provide support for end-users. To perform these actions they need to test, deploy and receive feedback from end-users.

Since not every developer can have an opportunity to establish such (smart) environment with different sensors at his/her place, there is a way for developers to test their developed products on remote smart environments. Testing scenarios provide preliminary feedback for developers.

After developing and testing products, developers bring them to customers. Developers have a way to put applications to a market, so that users would be able to reach applications through market. Also, since products are updated after some time, developers have a possibility to update files (applications) on a market.

To update software or fix bugs of a current version, developers get feedback from users. It is done to understand better which features of a product do not work as expected and need to be tuned up. Feedback is also needed to motivate developers to make their products better [Ant11], since feedback from users show to developers that their products are being used.

### 2.5.2 Administrators

Administrators of a market are users, who manage smart environments. They do it by keeping applications/services up to date, installing or uninstalling services/applications and configuring them in such a way so that they would perform in a desired way. Administrators can be treated as more elaborate end-users, since they have more responsibilities and actions to perform, that are described further. Administrators of smart environments manage it by starting or stopping applications, installing or uninstalling them from environments, performing configurations of them, if such options are made available by developers.

Also, administrator's feedback to developers is more detailed than end-users, since they refer to misbehaviors that occur in configuration or install/uninstall procedures.

### 2.5.3 End-Users

End-users of smart environments are the one who use already installed and configured applications. Applications are installed and configured by administrators of such environments.

End-users control their environment by using applications acquired from a market. For example by opening window in a room, fan, that was working a priori of this action (opening window), is stopped if such functionality is provided by applications currently installed/deployed in environment. End-users rate applications, write their opinions (comments) about specific applications and ask for support from developers if necessary.

## 2.6 Information provided by market

Market manages information, gathered from users and applications. It is done to support basic functionality and features of a market described in section 2.2.1 that builds up its value.

To evaluate products on a market, help developers to see how end-users respond to applications they have created, and help other users to choose better solutions for their needs, users are able to rate different items of a market. Items that can be rated are applications and their comments. Ability to rate comments of an application, help end-users to get most relevant comments first [SCNSP10] by omitting spam. Also, voting provides support for different features of an application market that are described in section 2.2.1

Crash reports are used as a feedback to developers and contributes in implicit rating of applications. Market constantly reports to developers about crashes in end-users environments, this helps developers to find solution for problems faster [MLLC10]. Resources in smart environment are limited. Deploying and running applications on environments cost resources to its users. Memory, capacity of storage unit (hard drive, solid state drive etc.), bandwidth and CPU usage are resources that we are

concerned about. These resources needs are classified into two parts: dynamic and static. Dynamic resources represent current information of environment where application is going to be installed. Static resources represent information that application needs, to run on environment.

Surveillance applications in private environments such as smart houses require privacy management policy if such systems are to be accepted by the occupants of the environment. This is due to the invasive nature of surveillance, and the private nature of home [MVW08]. Advances in both simple and complex sensor technologies have extended the scope of surveillance and monitoring applications beyond public spaces. However, such applications invade the privacy of those being surveyed [CDS04]. Information exchange with a market is needed for feedback, crash reports and during testing procedures. Administrators that manage smart environment can be third party people. In both cases, issues of privacy is of serious concern as sensors that are used (either individually, or in combination) have the ability to communicate sensitive information to those observing an environment.

Applications are able to use different sensors and functionality of other applications to perform desired actions defined by developers. If users don't want to allow new applications to use some of their sensors, or they want to be sure that application is going to access only specific devices, there could be a way to provide transparency on what sensors application is allowed to access.

## 2.7 Technical features of market

Some smart environments don't have user interfaces to control applications deployed on environment, thus provisioning and management of applications is done both remotely and locally. Administrators have possibility to install, uninstall and configure applications remotely.

Applications are rated by submitting crash reports, by rating them explicitly or implicitly. Comments are done about issues that are met during application usage. This feedback help developers and end-users to evaluate applications. After receiving crash reports, developers release a new version of their software, that fixes previous misbehaviors or introduce new features. New versions of applications are treated separately from its previous releases to differentiate crash reports, feedback of users and to reflect rating more accurately, since previous versions of applications could have bugs, that were fixed in a new versions.

Users of a market are distributed around it. Developers publish applications from one location, while administrators install them from another. To make applications accessible and store gathered information from different users and provide same information and application lists to all users, market provides centralized storage. In this storage all required (applications, ratings, comments etc.) information is stored.

Rating of applications, crash reports, comments of applications are collected by market. Crash reports are reported periodically, users rate applications and leave

their comments without predefined time interval. To make this information available for other users whenever they need it, for example see average rating of an application, or read comments that were left by other users, information collection is done by market not by local environment. Local environments submit their locally gathered information for evaluation to a global market.

Communication is done between smart environments and market. Malicious applications exist, that can expose users' private information, change permission that users' gave to an application or use users' devices to perform denial of service attacks [Gee05]. To overcome these issues market keeps track of applications published by its developers and provide secure communication channel for its users.

## 2.8 Distributed Smart Space Operating System

Distributed Smart Space Operating System (DS2OS) is a platform (shown in Figure 2.19) that brings connectivity and autonomous functionality into the home context [Aut08]. It is developed by Technische Universität München.

Application market designed in this thesis is highly dependent on functionality provided by this operating system. DS2OS consists of four layers, user interface, services, knowledge plane and control plane, presented below.

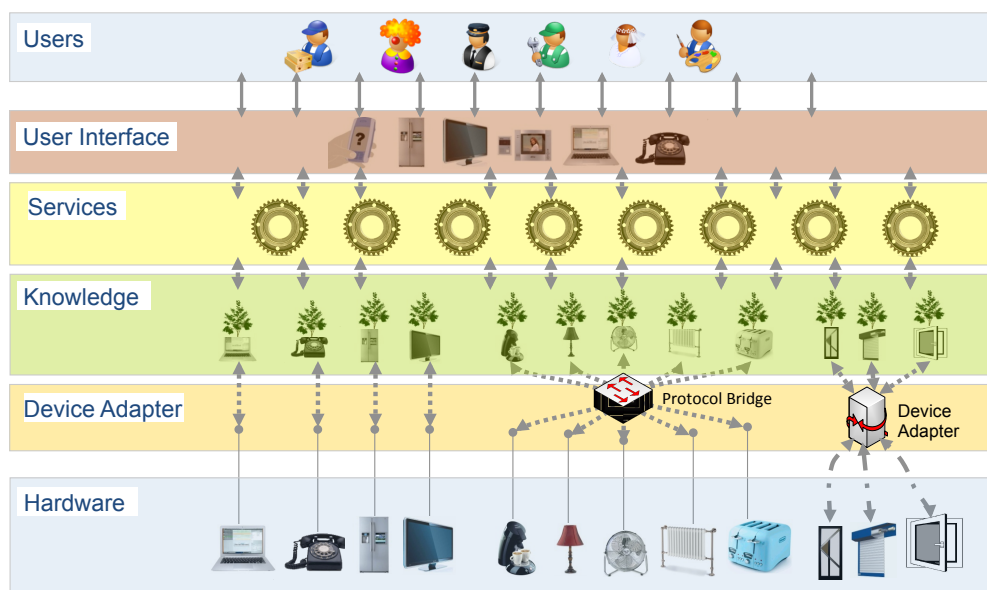


Figure 2.19. ds2os layers [PNS<sup>+</sup>09]

**User Interface** The user interface is the part of the platform the end user interacts with. Currently it is a web-based interface, where user can interact with their

environment, by manipulating their actuators. User interface is located on top of the service layer that is described further.

**Services** Services can be seen as software that user is using through the user interface to manipulate its environment. Services in DS2OS are running on nodes with sufficient resources inside the network. A service might for instance control the climate of a room or the energy consumption inside a house. A service might also take care that the music follows us through the house or even adapts to us and our mood outside, when we enter an autonomy ready environment like a pub where the music might adapt to the guests. Services are logically situated on top of the big abstraction layer "knowledge agent" that is described further.

**Smyrna** Smyrna is a service that is running in a smart environment. It is developed by Cuneyt Caliscan as a master thesis work. This service acts as a mediator between application market and smart environment. It is responsible for installing and removing of applications, run applications in an optimal way. It is also responsible for collecting information about environment and applications that are currently running on the environment. Smyrna is also responsible for migrating the services inside the network to provide load balance in the system and make sure that services get enough resources to perform their tasks. Market that is developed in this thesis resides between users and service layer of DS2OS.

**Knowledge plane** The Knowledge Plane (shown in Figure 2.20) is the core of the architecture. It contains the digital representation of the world. Every entity (device, service) that wants to profit from control and management architecture has to connect to the knowledge plane. This happens via the Knowledge Agent. The Knowledge Agent is the node's logical communication endpoint for everything concerning the knowledge overlay. If data from other nodes has to be retrieved this happens by requesting the data at the local agent. The agent will retrieve the data then and deliver it to the inquirer.

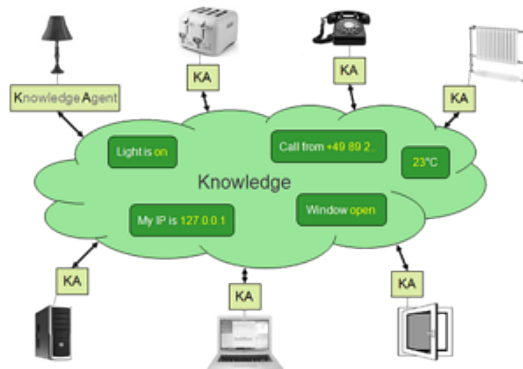
**Control plane** The Control Plane connects the hardware to the knowledge platform. In the Figure 2.19 Control plane is presented as Device Adapter layer.

## 2.9 Questions to be answered

After introducing problems with existing smart environments ??, how mobile phone environments deal with similar problems and desired properties, we can conclude to the goal of this thesis. The goal of this thesis is to answer the following central question that is further decomposed to more detailed sub-questions:

**How could a market for the smart environment be designed ?**





**Figure 2.20.** Knowledge plane

The above central question can be split into more detailed and specific questions that will be answered throughout the thesis.

*How can we apply and adapt tools and mechanisms of existing markets to smart environments ?* Application markets already exists, but not for the smart environments, 2012-04.

*How can privacy of a user be preserved ?* Exchange of information between local and global environments exists. Information that is exchanged can be private.

*How can analyzed security need be met ?* Malicious applications exist, that can harm environments and user security by exposing private information.

*How can described requirements of different user groups be met ?* Users with different rights (defined in section 2.5) are going to use the market. Different user types are going to perform different actions, that are unique and are not accessible by other types.

*How can market support described character of the smart environment with its unattended machines ?* Not all smart environments have user interfaces to control it. Applications can crash while performing given tasks.

*How to provide compatibility with different environments with their heterogeneous base?* Smart environments can bullfrog different amount of sensors and their types. Applications provided by the market have their own requirements to perform their tasks.

To address these questions, prototype of the market is designed in Chapter 4. Not all solutions of the questions stated above can be simulated. Solution for

questions that can be simulated and tested, are evaluated in Chapter 6.

## Chapter 3

# Related work

In this chapter important research that addresses or presents solutions to the current issues of smart environments mentioned in this thesis is presented.

Market solution presented in this thesis is based on already existing markets, implemented for different environments, not for smart environments. Review of already existent solutions is also presented that can be used in market for smart environments.

### 3.1 Explicit and Implicit Ratings of application

In order to distribute applications, developers publish them on a centralized portals (market places) from which consumers can directly search, download and then install applications onto mobile devices [HO09]. At present these marketplaces let users rate applications by giving rating on a one to five star scale. Explicit rating are the most common and obvious user interest indicator, with users telling the system their opinion about some objects (such as a video clip) or piece of information (such as a blog post). Even though this rating technique offers a fairly precise way to measure user interest, it has several serious drawbacks: to enter explicit rating to the system, users have to alter their reading or browsing behavior, they might even stop providing ratings unless they perceive a benefit. In general users tend to read much more articles, than they rate, as it was found through research on the GroupLens system [CBLW01]. Even though explicit ratings are common and trusted, they might not be as reliable as often presumed. [DN06] found out, that users are more likely to review applications only if they find them very good, or very bad. To overcome this limitation, [GM10] introduced a new way to rate applications - implicitly.

Implicit ratings provide a good alternative, since they remove the cost from a user associated with examining and rating items. What is more, every user interaction with a system can be converted to an item evaluation. Events when user update, install or uninstall application contribute in application rating. Although each im-

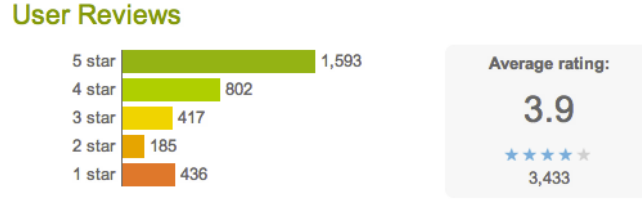


Figure 3.1. Google Play Market users rating review screen

licit rating item can be less accurate than an explicit rating, it can be gathered for "free".

Every time user install, update or remove an application, these events contribute to the rating algorithm. The assumption behind this approach is that good applications are not removed once installed, whereas applications not liked, tending to be removed from device. AppAware defines acceptance rate  $v$  for an application  $app$  as a value going from 0 to 100 computed with the formula in 3.1, where  $U$  is a set of users having at least one event for  $app$  [GM10].

$$v(app) = \frac{\sum_{user \in U} last(app, user)}{|U|} \quad (3.1)$$

$$last(app, user) = \begin{cases} 0 & \text{if last event of user for app = removed} \\ 90 & \text{if last event of user for app = installed} \\ 100 & \text{if last event of user for app = updated} \end{cases} \quad (3.2)$$

Formula 3.2 defines that only most recent event of a specific user for a certain application is counted. In the process, an update procedure is considered the most valuable event, even more than installation. The belief is that an application's update procedure indicated that users interest in that piece of software still exists. At the same time it indicates that developers put effort to keep application up to date. It is also possible to include time spans between installations and removals of applications, to give bonus to programs that remain installed for a longer time. Additionally, due to the nature of certain applications (games, season related applications), check for possible systematic biases might be useful.

## 3.2 Content recommendation

With the current amount of information coming to the market - over 500.000 applications [Inc08c], choosing among so many options is challenging for consumer. To help users cope with this issue, there exists recommender systems that provide users with the recommendations of items that are likely to fit their needs [SKKR00]. Association rules mapping can be used to develop recommender system. In [SKKR00] association rules are defined as: discovering association between two sets of products such that the presence of some products in a particular transaction implies

that products from other set are also present in the same transaction.

One of the most successful systems that provide recommendations to the users are Collaborative Filtering (CF) systems [KMM<sup>+</sup>97], [SM95]. Numerous commercial on-line based companies like Amazon.com, Half.com and cdnow.com apply this technology to provide recommendations to their customers. Unlike content-based information filtering systems that are based on artificial intelligence or information retrieval technologies, filtering decisions of content in CF are based on human, not on machine analysis. Users of CF rate items that they have experienced and build their own profiles of interests by doing so. CF match together users of similar interests or tastes. Ratings from similar people are used to generate recommendations for user. Since CF is not based on error-prone machine analysis of content as traditional content-based filtering is, it has many significant advantages. Text, art work, music, video can be filtered for the user by using CF techniques [HKR00]. The drawback of collaborative filtering approaches is that they are computationally expensive [LSY03].

Recommender systems predict user preferences based on a range of available information. For systems where users generate content (applications in our case), users may rate produced content that they read or used (in this thesis case, applications are considered to be used), and be given accurate prediction about items they may also like. Access to such data allows companies that provide content recommendation features to improve their recommendations and also to profile users. Such profiles can be later monetized, e.g. through advertising.

In the same time, the increased monetization of private data has been met by a sharp rise in privacy concerns within advocacy groups like the Electronic Frontier Foundation and regulatory bodies like the US Congress. In [IICM11], distributed mechanism for predicting user ratings that avoids the disclosure of information to a centralized authority or an untrusted third party is designed. Users disclose ratings and their evaluations to certain content only, to a user that produced that content. This content filtering approach helps users to view and access relevant, interesting content without releasing their private information to untrusted third parties. At present, when user creates content on Facebook or a Twitter feed, his profile information (what user liked in the past, who liked his posted content) is kept centralized. [IICM11] approach to share content and ratings with each other is in a peer-to-peer fashion. Their collaborative filtering scheme operates under the constraint that information is shared only between content producers and its subscribers.

Another profile building technique, used in content recommendation is described in [KASJ11]. To build an individual user model/profile, tags and ratings are connected to infer user's topics of interest, in which each topic is composed of tags. To provide model with more diversity, valuable topics in term of both likes and dislikes are enriched in collaboration with other similar users.

In [DLL<sup>+</sup>10], YouTube's video recommendation system is presented. It delivers personalized sets of videos to signed in users based on their previous activity on the YouTube site. This recommendation technique is focused to be recent and fresh,

and relevant to users recent actions. The set of recommended videos is generated by using user's personal activity (watched, favorite, liked videos) as seeds and expanding the set of videos by traversing a co-visitation based graph of videos. The set of videos' is then ranked using a variety of signals for relevance and diversity. In general, content data and user activity data are considered. Content data can be seen as video metadata, such as title, description etc. User activity data is divided into two parts, explicit and implicit. Explicit activities include rating a video, favoring/liking a video or subscribing to a specific uploader/content provider. Implicit activities are datum generated as a result of users watching and interacting with it, e.g., user watched large portion of a video (long watch). YouTube recommendation system uses a well-known technique known as association rule mining [AIS93].

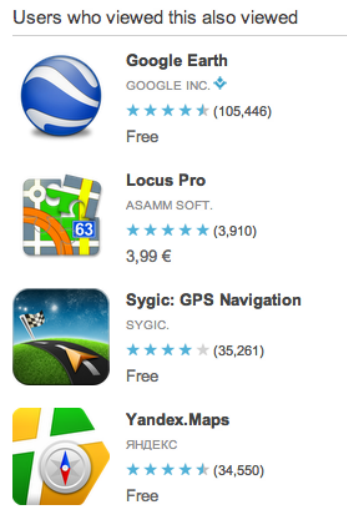
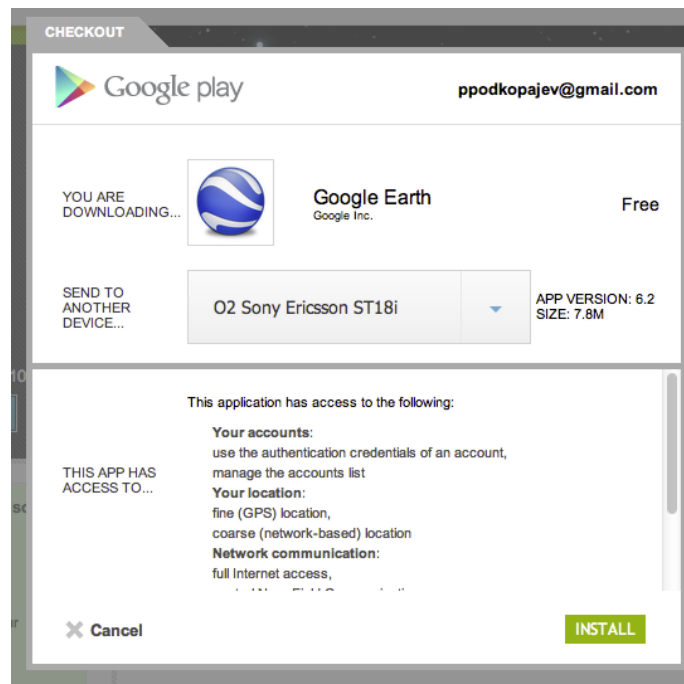


Figure 3.2. Google Play Market content recommendation

### 3.3 Provisioning

Provisioning of applications to a market let their users to upload applications to it, install to their environments and update if a new version of application was released by developers.

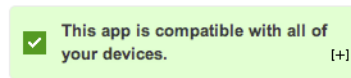
Currently, users of Google Play, have three ways to deploy/provision applications to their devices. First way is to download it through markets application on mobile device. Second way is to download application manually, for example to a personal computer, than transfer download file to a mobile phone and open it with Markets package manager that will install application. Third way to install application on a mobile device, is to do it remotely. Users are allowed to browse for applications through web browser of their personal computers. Through same web browser it is possible to send application to one of user devices (Figure 3.3), where it is



**Figure 3.3.** Google Play Market remote provisioning guide screen

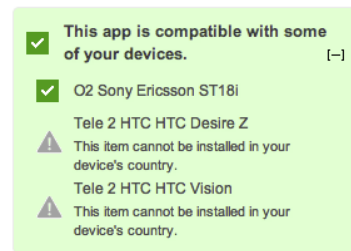
automatically installed by local package manager. Smart environment defined in this thesis, does not have user interface, what makes third way of provisioning the most interesting for this thesis. In Apples App Store, users are allowed to remotely provision applications to their devices. To that, special client software is required, Apple iTunes [Inc08d]. Through this client, users are allowed to download application to their personal computers, and later synchronize content on client software with their smart phones (iPhone, iPad). This approach is different from the one presented by Google Play. In Google Plays' approach, remote provisioning can be done purely through Internet, without requiring device to be directly connected to a personal computer on which provisioning action is performed. In Apples App Store approach, device on which remote provisioning is done, have to be connected to a personal computer on which provisioning action is performed. Starting from iTunes version 10.5 and iOS version 5.0 and above, users of Apples App Store have possibility to install application to their phones remotely, by using WiFi, if this option is activated. Whenever iOS device connects to a WiFi network, to which personal computer is connected to, applications downloaded to iTunes are transferred to a mobile device [Inc08e].

Different applications can have requirements that are not supported by some of environments. Some applications are restricted to a specific countries [Inc08h]. In Google Play market, users are informed a priori if application that they are trying to install is compatible with their device environment (figure 3.4). If users have several



**Figure 3.4.** Google Play Market application compatibility view

devices, they are informed with which devices application is compatible with (figure 3.5).



**Figure 3.5.** Google Play Market application compatibility view with all devices available to user

### 3.4 Update management

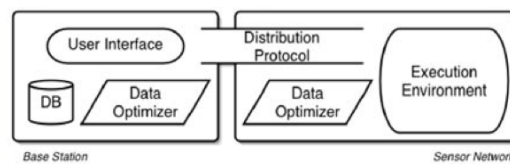
Software updates/patching are common in all systems. Tools to help update software in traditional systems have been used for a number of years. System administrators often need to check for software updates, download them, verify the integrity and distribute them. There are several tools from different vendors that help administrators to cope with this problem. Microsoft Systems Management Server includes support for planning application deployment, selective targeting of application deployment, security, patch management and the ability to propagate differential changes to software rather than the entire application image. The Tivoli Configuration Manager is another tool from IBM, which provides automatic patch distribution, multicast based software distribution, and pervasive device configuration management. Existing update management (UM) tools are designed to aid system administrators with two primary problems. First, the diverse set of software and hardware configurations present in a large network presents a great deal of complexity that cannot be effectively handled by a human. UM tools provide selective installation based on specific software/hardware combinations to help reduce this complexity. Second, UM tools provide a mechanism for content distribution in the network [HKSS05].

Update management model presented in [HKSS05] is focusing on sensor networks. A sensor node is typically equipped with a low bandwidth radio, an MHz (instead of GHz) micro-controller, and kilobytes (instead of megabytes) of RAM to provide prolonged operations under limited power.



The model is made up of five main components:

1. Software updates planning, selection of potential targets based on system parameter and monitoring the progress of the updates is done through **User interface**.
2. **Database**, that is used for marketing software images for updates and their planning.
3. **Data optimizer**, that is used to minimize the size of updates before they are distributed.
4. **Distribution protocol**, that is used to transmit updates into targeted sensor nodes.
5. **Execution environment**, that is used to execute programs on a sensor node.



**Figure 3.6.** Update Manager Model [HKSS05]

An example of update process that involves all five components could be as follows. System administrator initializes/plans update through UI. UI pulls new and old sensor drivers from database and forwards them to data optimizer module. This module compares two versions and creates a patch of optimal size. This patch is afterwards passed to distribution protocol to deliver update to nodes. When patch arrives at a destination node, the execution environment uses patch to update sensor driver.

Another solution to manage home appliance systems is presented in [KWKI03] (System overview is presented in figure 3.7). Bluetooth for a communication medium and a cellular phone as the control terminal is used in their proposed system. Bluetooth is used to establish communication between appliances and mobile device. Mobile device is used to control appliances and to perform software update. Java application deployed on a device provides users with further functionality:

- Remote Control
- Program Update
- Fault Diagnostic

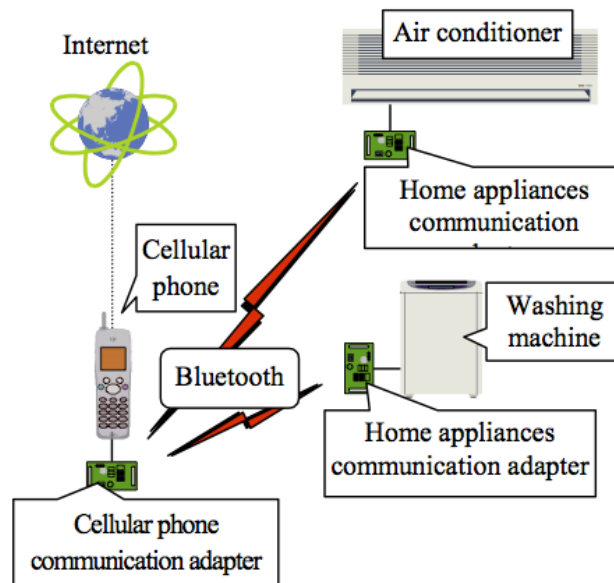


Figure 3.7. System Overview [KWKI03]

Mobile phone acts as a mediator between service providers, developers of software and appliances and their users, home occupants. Mobile phone gathers necessary data, and sends it throughout the Internet.

### 3.5 Privacy

Smart house environments can store a large amount of data. This data can be exchanged with a market, to provide test results for developers or any other kind of feedback from environment occupant. Previous approaches to privacy in ubiquitous computing systems either implement privacy by default, or assess the context of the environment in order to determine whether privacy should be implemented, or not. The privacy measure is then implemented by rule-based approach. Each user had static predefined privacy preferences that were taken into account in rule appliance. This approach is limited due to the binary influence of the context. If privacy measures are in place, environment does not influence observer's view of data. These results in a single privacy view for a given observer. Smart house environments is dynamic and flexible, [MVW08] presents a framework that implements privacy in such environments. It provides an interface between the implementation of a privacy and occupants.

The framework dynamically determines what data an observer can access given the situation, or context of environment. A dynamic approach is required to balance the needs of occupants with usefulness of the system in achieving its purpose. These two aspects are in conflict. For an occupant, privacy policy should minimize intru-

sion into occupant's life (maximize level of privacy experienced). However, for the system to function adequately, observers, developers for example, should be able to access sufficient information to perform their duties, that is in this thesis case, to provide necessary feedback, crash reports and information about user.

Even though information gathered from such environments can bring benefits to occupant's life, it also can harm it. Currently, users of other smart environments (smart phones, tablets) report [Reu08] issues, that violate their privacy. Smart phones from Apple, iPhone, were transmitting information about users location, current and in the past, to third party developers, without their permission. By collecting location information embedded in location-based service queries, an adversary who has compromised the location-based service server can infer sensitive privacy information about service recipients, such as their home locations, lifestyles, political/religious associations, and health conditions [SJCH12]. Information like owners home location, smoker or nonsmoker, coffee drinker or no, or even age; can be find out from location-based service data.

### 3.6 Security

Current application markets, before letting users to use their facilities require users to register in a system. Google Play requires performing first time registration from a physical device, in this way system will relate specific device to a user. Apple requires connecting mobile phone to Internet and will allow users to use its device only after registering it to his Apple account. After performing these steps, users of both application markets can start using functionality provided by a markets.

Malicious applications started to attack application markets [DCGMV11], [Aim08]. Different application markets have their own approaches to eliminate danger that this type of applications present to its users. In Apple App Store, all applications that are submitted to a market are manually processed through application approval process. In Google Play market, Bouncer application is introduced which monitors applications published to a market and removes the one that is suspected to be malicious. After introducing Bouncer application in a Google Play, it was seen a 40 percent drop in malicious apps [Ras08]. While Apple provides safer environment than Google Play [Aim08] for its users with its manual application approval, it comes with a price. Application approval takes longer time [MLLC10] comparing to approval time in Google's Google Play Market. Money is spent on people who evaluate software and check if it is malicious or not.

In order to reliably determine whether code that was produced by developers has been modified by someone or not, code has to be signed [KPS02]. Code signing can be used to:

- ensure the integrity of the code; that is, it has not been altered
- identify the code as coming from a specific source (the developer or signer)

To enable signed code to fulfill all of these purposes, a code signature consists of two parts:

- A seal, which is a collection of checksums or hashes of the various parts of the program, such as the main executable, the resource files, and so on. The seal can be used to detect alterations to the code.
- A digital signature, which signs the seal to guarantee its integrity. The signature includes information that can be used to determine who signed the code and whether the signature is valid.

It is important to understand that signing code cannot:

- Guarantee that the code is free of security vulnerabilities.
- Guarantee that a program will not load unsafe or altered code such as untrusted plug-ins during execution.

### 3.7 Transparency

Transparency of applications can be seen as a security concern [Meu08], since marketplaces give user ability to install third-party applications. While existing transparency mechanism in Android allow user to see which resources are required by application (Figure 3.3), or which resources are going to be accessed (Figure 3.8), she/he has no choice than to allow access to all requested permissions to use application or reject them. What is more, there is no way for a user to grant only some of permissions, while denying others. There is no way of restricting resources based on runtime constraints such as the number of time a resource has already been accessed/used. If user wants to revoke permissions once they were granted, the only way to do it, is to uninstall the application. While building market for smart environments, where users can have same issues, it is important to cope with these issues from the beginning.

Users of Google Play Market have a solution for such problems - Apex [NKZ10]. This application extends functionality of Google Play, by giving users possibility to dynamically control permissions for their applications. With a help of this application, it became possible to limit specific permission for a specific amount of time. For example, if service requires permission to send SMS, it is possible to limit this permission to 100 times. In this way customer is confident, that no more than 100 SMS will be send.

Users of smart house environments can have a service, for example, that keeps track of products in a refrigerator and orders them online. It can be the case that user want to deny permission to buy groceries online, but still to keep track of products inside refrigerator. To make sure that service is not ordering anything without users' permission. Previously granted permission to do it, should be revoked, but other permission, that grants access to refrigerator, should not. This possible procedure makes issues described in [NKZ10] relevant to a market for smart environments.

### Permissions

**THIS APPLICATION HAS ACCESS TO THE FOLLOWING:**

#### **YOUR LOCATION**

##### **COARSE (NETWORK-BASED) LOCATION**

Access coarse location sources such as the cellular network database to determine an approximate tablet location, where available. Malicious apps may use this to determine approximately where you are. Access coarse location sources such as the cellular network database to determine an approximate phone location, where available. Malicious apps may use this to determine approximately where you are.

##### **FINE (GPS) LOCATION**

Access fine location sources such as the Global Positioning System on the tablet, where available. Malicious apps may use this to determine where you are, and may consume additional battery power. Access fine location sources such as the Global Positioning System on the phone, where available. Malicious apps may use this to determine where you are, and may consume additional battery power.

#### **NETWORK COMMUNICATION**

##### **FULL INTERNET ACCESS**

Allows the app to create network sockets.

#### **STORAGE**

##### **MODIFY/DELETE USB STORAGE CONTENTS MODIFY/DELETE SD CARD CONTENTS**

Allows the app to write to the USB storage. Allows the app to write to the SD card.

#### **SYSTEM TOOLS**

##### **PREVENT TABLET FROM SLEEPING PREVENT PHONE FROM SLEEPING**

Allows the app to prevent the tablet from going to sleep. Allows the app to prevent the phone from going to sleep.

Show all

**Figure 3.8.** Google Play Market permission screen



**Part II**  
**Design**

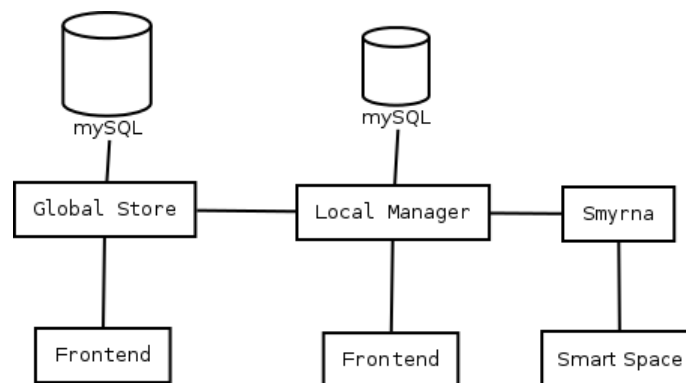




## Chapter 4

# System Design

The following chapter presents the design choices made to solve the problems presented in Section 2.9. In figure 4.1 entities of the system designed in this thesis are presented. Functional components/modules that are implemented on specific entities are presented below. Description of these modules is also provided.



**Figure 4.1.** Entities of the system

**Global Market** consists of further functionality modules:

- Application and information storage
- Version control
- Signature/Integrity check
- Information collection from users and environments
- Application description
- Application recommendation
- Implicit rating
- Dependencies

- Malicious application removal

**Global Market frontend** consists of further functionality modules:

- User authentication
- Publish Application
- Remove Application

**Local Manager** consists of further functionality modules:

- Signature/Integrity check
- Private information filtering
- Crash reports
- Environment specification gathering
- Transparency of applications
- Compatibility
- Malicious application removal

**Local Manager frontend** consists of further functionality modules:

- User authentication
- Application preview
- Search
- Top charts
- Application Control

Smart Space entity indicates Smart Space runtime environment. Applications in the smart environment are deployed and controlled through this entity by the local manager entity.

## 4.1 Application and information storage

Global Market entity collects information. Users of the system, comments and rating they submit, applications that market provides are stored in Global Market entity's database.

It markets information about applications and users of a market. Application information that is stored in a database consists of application description, average rating, list of users that installed or rated application, preview images and recommended items for specific application. Preview images and application files are all stored locally in global market to provide availability. User information that is stored, consists of usernames and passwords, items owned and rated, ratings of these items and comments made for those applications.

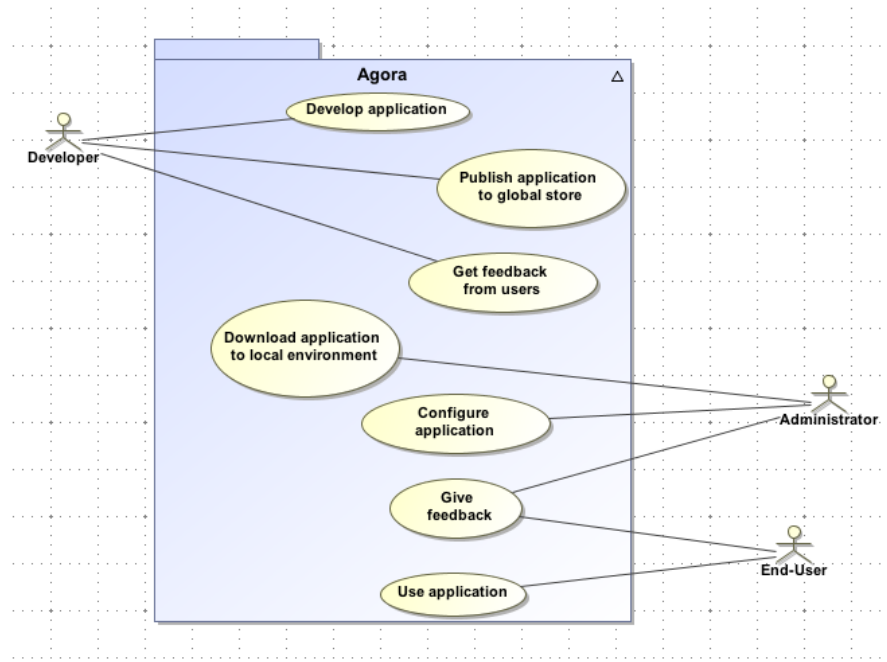


Figure 4.2. User interaction with a system

## 4.2 Version control

Whenever developers release new version of their application, they publish it to the market with different, increased version number. Previous and current versions of applications are stored in the market storage. This is done to support users that may not have compatibility with new version requirements, but still have resources to work with a previous version of application.

## 4.3 Signature/Integrity check

Applications that have security issues may damage users environment by exposing private information to a third parties. In section 4.4 these issues are addressed. Public key infrastructure is used to provide digital signature, which provides integrity of data. Certificate authorities can issue certificates to prove that specific public key belongs to a specific signer. More information about public key cryptography and certificate authorities can be found here [KPS02].

## 4.4 Malicious applications removal

Applications that have security vulnerabilities, load untrusted plug-ins during execution or perform actions in background that are not described, pose a threat for

smart environments. Built in functionality of the market - bouncer, monitors such applications. Users are allowed to explicitly mark application as malicious, also, it is implicitly checking for communication of such applications with malicious plugins and services automatically. It is done by applying three different strategies: popular, not popular and random. If bouncer is executed with random strategy, it chose one application per round at random, checked if it malicious or not. After determining application as malicious it is removed from the market, so that other users won't be able to get infected. Also environments that have such application deployed in their environments before it was marked as malicious, are forced to remove it. It is done by remote method call on local environments. By executing this method, application is removed by local manager.

## 4.5 Application recommendation

The market provides number of applications to its users, it is important to have recommendation system, that helps users to find application that may be useful for them [SKR01]. There are many different content recommendation techniques like collaborative filtering, content-based filtering or combination of them [LSY03] [SKKR00].

In our case we are focusing on collaborative recommendation approach. For each user we construct a list (item list) that keeps track of all items user rated or have purchased in a past. Whenever a user consider buying or installing another application, we present a list of applications that other users have bought together with the one user is currently considering acquiring.

This list is updated in background once a day, to lower the load on the system. If this list would be updated every time new user installs a new application, it would increase accuracy of recommender system, especially in the beginning, but would increase system load for a system lifetime. A list with recommendations is stored in a database.

## 4.6 Dependencies

If an application that user is trying to install is dependent to another application or several applications, and user does not have these applications in his local environment. Application market informs user about missing dependencies. A list with missing applications is presented with an opportunity to install missing components. This helps user to run application that is dependent on other applications without which it wouldn't operate.

## 4.7 User authentication

Users with different roles and rights interact from local environment with a global market and vice versa. User authentication is required. System authenticates users by accepting username and password. User database is centralized and stored in Global Market entity. Authentication is done through Secure Hypertext Transfer Protocol (HTTPS), which provides encrypted communication. All user informations are stored in databases that are located in both global and local environments. Before user is able to use the market, it is asked to enter user name and password. If match is found, user is granted

## 4.8 Provisioning

Provisioning is done through Local Manager entity. Users that have access to this entity can browse applications that are available in Global Storage. Local manager provides install, remove methods, which are used to install or remove applications from environment that local manager is responsible for. After collecting configuration options for a specific application, local manager entity send install request to "Smyrna" (defined in section 2.8), which further installs application with provided configuration. To remove application, local manager entity sends "remove" request with an application identifier to "Smyrna" service.

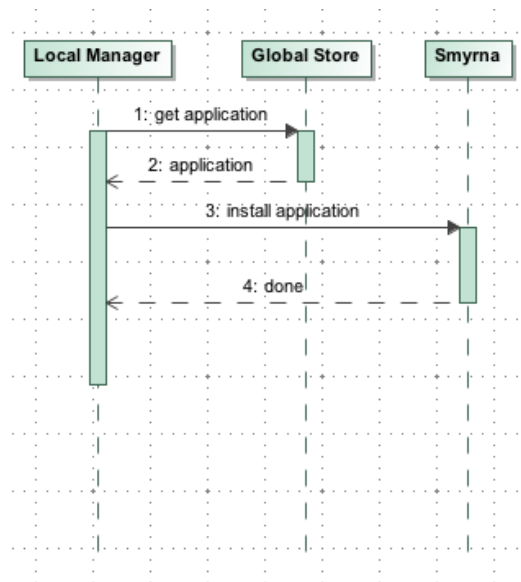


Figure 4.3. Provisioning

Action	Description
Start	To start an application, user selects an application from the list of installed applications. Start option is available if application is not already running.
Stop	To stop an application, user selects an application from the list of running applications. Stop option is available to stop an application.
Update	To update an application, user selects application from the list of applications that have update available.
Configure	To configure an application, user selects application from the list of installed applications. Configure option is available if application is configurable.

**Table 4.1.** Application control features and description

## 4.9 Application control

Control of application is provided by local manger entity described in Table 4.1. User is able to start, stop, update and configure applications. This control options are available to applications that are already installed on the environment. These requests are sent to "Smyrna" service.

## 4.10 Top charts

Top charts features are important options of every market, since it helps users to find solutions that majority of people are using. In our thesis we provide these four top chart lists:

### Top Free

This list provides first ten free, most downloaded applications on the market.

### Top Paid

This list provides first ten paid, most downloaded application on the market.

### Top Grossing

This list provides first ten, most downloaded during last week application on the market. This helps new applications that attract attention of user to be promoted. In this way new or better realization of previous versions of applications can compete with applications that are in a top list for a long time.

### Featured

This list provides high quality applications. Market staff members select them.

## 4.11 Rating

### 4.11.1 Explicit rating

To rate applications explicitly, we use same concept that current application markets are using - five star rating scale. User is able to choose value from one to five stars. This value is summed up from all users that rated their applications, and average value is presented to other users. Rating option is available only for those users who have application installed on their environments. This option helps to gather rating more accurately, since users that didn't try application, barely know its value. Also, statistics of how many users voted for a specific rating value is presented. Comments have only two options to rate - "helpful" or "scam". A "Neutral" option is default value for all comments.

- Helpful vote gives 1 point to an overall comment rating value.
- Neutral vote gives 0 points to an overall comment rating value.
- Scam vote gives -1 point to an overall comment rating value.

Every user after reading a comment can decide if information provided was helpful or related to an application. After gathering ratings for comments, more useful comments are presented first. This helps new users to get the most useful comments first, skipping spam or unrelated scam messages. The same technique to evaluate comments is used in many places where people exchange their opinions, since more and more spam messages start to appear in such communities.

### 4.11.2 Implicit rating

As stated in section 3.1 users of markets are tend to review applications if they perceive them as either very good or very bad. To overcome these issues, our market has implicit rating technique. Which rates applications automatically, depending on user usage.

In this thesis we are implicitly rating applications according to the last action performed with an application (install, update, uninstall) by its users, keeping in mind time spans between these actions, and nature of specific appellations.

Application gets positive points for installation and update actions, or zero, for removal action. We treat update mechanism as maximum possible rating - 100 points, since it shows that user is willing to use application further and developers support their application. Install of an application gives 80 points. Removal of an application contributes with 20 points to an overall rating. These values are changed from 100, 90 and 0. Value 0 is changed to 20 points to rate application with one star value, so that it would get negative rating instead of being unrated. Value 90 is changed to 80, so that application would be rated with 4 star value, instead of 4.5. Also, to give some bonus points to those applications that were removed later (the one that were used for longer period of time), we calculate time spans. Some

category of applications are removed because there is no more use of it, for example user completed all levels of game, or he is moving from demo version of application to a full version. We deal with such issues by not rating demo applications, even though it is possible to check for example if user installed full application version after removing demo version of it. Crash reports described in 4.12.1 also contributes in implicit ratings. Applications that are running without crashing for longer time gets bonus points or lose it, if keeps crashing.

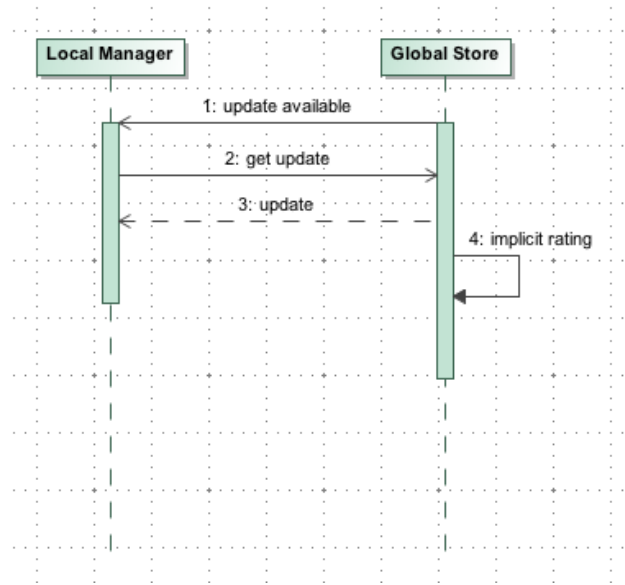


Figure 4.4. Implicit rating

## 4.12 Feedback

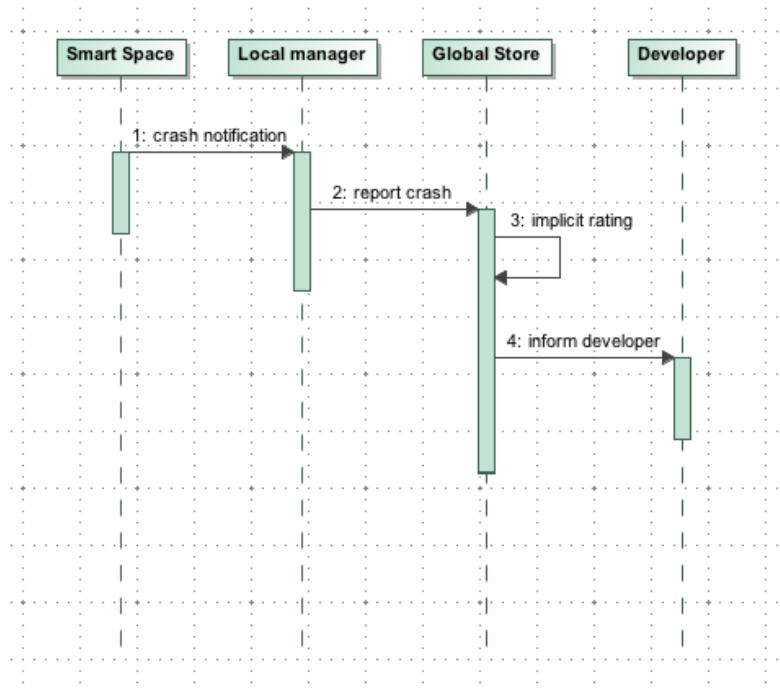
A prompt response from an application developer gives strong feelings about the solidity of the application and the people supporting it [MLLC10]. Our solution provides two-way communication between users and developers of applications. Not only users are able to write comments and rate applications, but also developers are able to reply to comments that users made on their applications.

### 4.12.1 Crash report

Crash report help developers to improve quality and solve problems in any of applications they produce. Since applications in smart environment are running unattended, automatic crash report feature exists to submit these reports to developers. these reports are also used to evaluate quality of application (Section 4.11). These reports are submitted to Global Market entity from Local Manager entity. After



this report is submitted to the Global Market entity it is sent to responsible developers. Crash reports are reported to the local manager entity by "Smyrna" service. To preserve occupants privacy and save amount of traffic between local and global



**Figure 4.5.** Crash report

market entities, crash reports are aggregated during the day, and are sent to the global market once a day. Aggregated crash reports are stored in a local manager's database. Report store application id that has crashed, time when the event occurred and environments issues if any. All reports that are stored in the database are submitted to the global market entity once a day.

## 4.13 Transparency

To overcome transparency issue defined in section 2.9, the market gives its users opportunity to review which sensors will be accessed and, if any, applications that will be interacted. This list of permissions that application requires is presented during the installation process. List is separated into two parts. First, the most critical items of permissions are presented, this will help user to focus on the most critical permissions, without going through the full list of items. Also, if user wants to review the whole list of permissions, expanding the list can do it. This list is build from a manifest file, that developers construct while developing application. Devel-

opers are allowed to use/access specific sensors or perform actions on environments appliances only after explicitly defining them in manifest file.

#### **4.14 Private information filtering**

Feedback that is submitted by users of the market is saved in global market entity. To preserve users privacy, information about users is never exchanged and used in statistics. Only feedback values are exchanged. Also, it is possible for a user to set up a privacy filter, where he or she can decide which information can be sent to the global market entity.

#### **4.15 Compatibility**

To help market users understand if application that they want to install is compatible with their environment, local manager automatically checks whether required components are present in the environment. Compatibility status can be of three different types: green, yellow and red. If all components are present, compatibility status is displayed as green and user can proceed with the installation. If application has different versions, and previous versions are compatible with the environment, compatibility status is yellow, user can change version of an application to the one that is compatible with the environment. If environment is not compatible with all versions of application, compatibility status is red. Also user can check which requirements are missing in his/her environment to consider an upgrade to meet all requirements.

## Chapter 5

# System Implementation

This chapter provides implementation specific details about the system presented in this thesis. First, system topology is presented, what technologies are used in implementation of the system. Further, implementation of specific functions is presented.

### 5.1 System topology

Global and Local market entities are build in PHP language. There are two databases - one for each entity. Data about users of the market, environments and applications and its ratings are stored there. MySQL relational database management system is used to operate with databases.

To communicate between Global and Local entities (Firgure 5.1), JSON is used to invoke remote methods of the entities. So, for example, whenever Global market entity requests local manager entity to remove malicious application, request is serialized into JSON. On the other side, local manager entity is waiting for a requests, where JSON is deserialized into PHP and necessary function calls are made, with given parameters. Communication between local manager entity and Smyrna is done with cURL and Sockets. Local manager entity send requests to Smyrna through Socket while Smyrna replies through cURL.

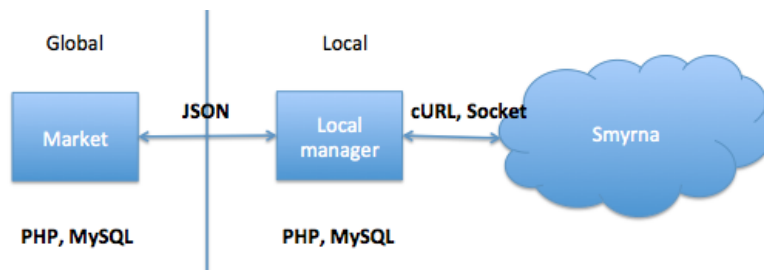


Figure 5.1. Agora system topology

## 5.2 Global market

Global market entity, named Agora in this thesis, is implemented in PHP language. HTML, CSS and JavaScript were used to create graphical interface. 875 lines of code is written for it's functionality part only. It is executed by Apache server. One database is running on global side of the system. MySQL Server is used to run the database. phpMyAdmin tool is used for database administration. The database consist out of 22 tables. Application information and statistics, user information, environments and their installed/purchased applications, ratings, crash reports, statistic results for every round and simulations are stored in these tables. The relation between these tables is control by PHP script that stores or retrieves information from database when needed.

Initial population of the database was done by utilizing Apple App Store web service. JSON was used to transfer applications, their descriptions and all necessary information from Apple servers.

## 5.3 Local manager

Local manager entity is also implemented using PHP, HTML, CSS and JavaScript. One, smaller database is running on local side, to support applications that require to store data in database. Also, this database is used to store crash report and other data in case of internet connection is lost during communications, so that data can be submitted later, when connection is resumed. Local manager entity is build out of 400 lines of PHP code and 5 tables in MySQL database.

## 5.4 Implicit rating

Implicit rating is a service that is running on the Global market entity. Whenever user installs an application, system will log the time that it was installed on, and a user id that installed application. These values are stored in a separate table in the database. Whenever system calculates average rating for the application, these values contribute to it.

If user removes an application, system will log the time it is uninstalled and calculate the difference with an installed time. Afterwards, this difference is compared to the time intervals to assign necessary rating. In our system we are using four time intervals: 0 - 15 minutes, 15 minutes - one day, one day - more than one week and more than one week. If time difference between install and uninstall of an application is between the first time intervals, it will give the lowest rating value - one star. If time difference is in the second interval, it will assign two stars to an application, for the third time interval it will assign three stars, and for the last one, it will rate application with four stars.

The highest rating is assigned if application is updated.

## 5.5 Recommendation

This recommendation list is build in a following way:

Global Market entity picks all users that have bought or rated specific application, and looks through all other items they have in their item lists. Whenever system finds item that two or more users have in common, item is pushed to the recommendation list. Items in recommendation list are sorted according to number of users that have it. Applications that are repeated in larger number of users are considered more relevant since more users are using them.

## 5.6 Malicious application removal

Bouncer feature has three different strategies to choose from: "popular", "not popular" and "random". When bouncer strategy is set to "popular", every round the most popular application at that round is tested if it is malicious. When bouncer strategy is set to "not popular", every round the least popular application at that round is tested if it is malicious. When bouncer strategy is set to "random", every round random application at that round is tested if it is malicious.

If application is malicious, it is removed from the market, and all environments that are using this application are informed. If application is not malicious, it is remembered to be safe. New version of already checked application is considered to be new application and is checked again.

## 5.7 Test environment

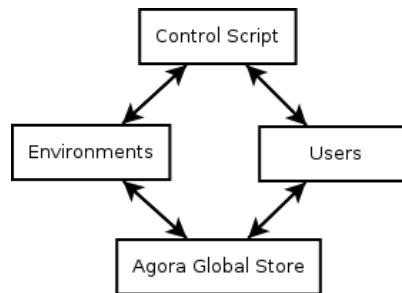
To check whether designed prototype of the market address defined issues, simulation is developed. Users of the market are simulated. Developers create and support applications, Administrators install, update, remove and rate applications and Environments that can make application unstable and crash, or submit specific data that is described further (topology can be seen in figure 5.2). End-users are not simulated since end-users are mainly involved in control of an applications, which is not the case that we want to simulate.

This simulation consists of different modules (that are further presented) and is a round-based simulation. Modules that are used in simulation are as follows:

- Control script - Script that is executed by shell. This script defines actions and modules that are used during the simulation.
- Environments - PHP implementation of home module, that is invoked by control script when action from home is needed. Actions that have to be performed by different homes are defined with probabilities.
- Developers - PHP implementation of developer module (266 lines of code), that is invoked by control script when action from developer is needed. Ac-

tions that have to be performed by different developers are defined with probabilities.

- Agora Global Market - this is the module that our simulation is made to test. This module is fixed, no random events are happening inside this module. It is also responsible for outputting necessary data to a csv file, to log necessary data.



**Figure 5.2.** Simulation topology

Simulation starts with the population of users and environments. Afterwards system cycles through rounds.

## Chapter 6

# Test/Evaluation

In this chapter, the system is evaluated with respect to the thesis problem statements and objectives. Section 5.7 gives an overview of a simulation framework and Section 6.1 presents the simulations scenarios and their results.

### 6.1 Evaluation

#### 6.1.1 Rating

In rating test we wanted to test implicit rating feature that we introduced. We were expecting to have more accurate rating results, that will help to determine quality of applications more accurate since no applications will be left without rating.

In figure 6.1 current rating distribution of Google Play Market applications is presented. Almost 50% of all applications are not rated [App08b].

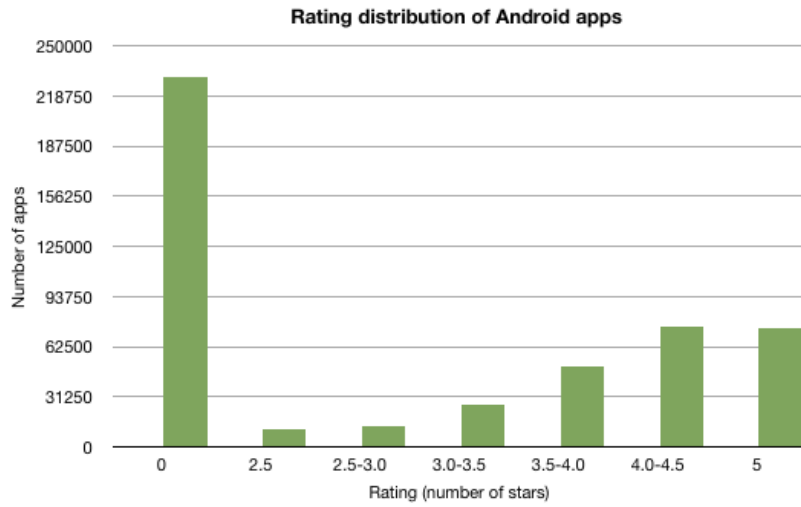
After running the markets simulation, all user events were correctly caught and correct rating values were assigned. All applications that were installed by at least one user were rated with according rating values. Implicit rating feature provided designed market with 50% more rating event if these results are compared with the current Google Play Market rating distribution.

#### 6.1.2 Crash reports

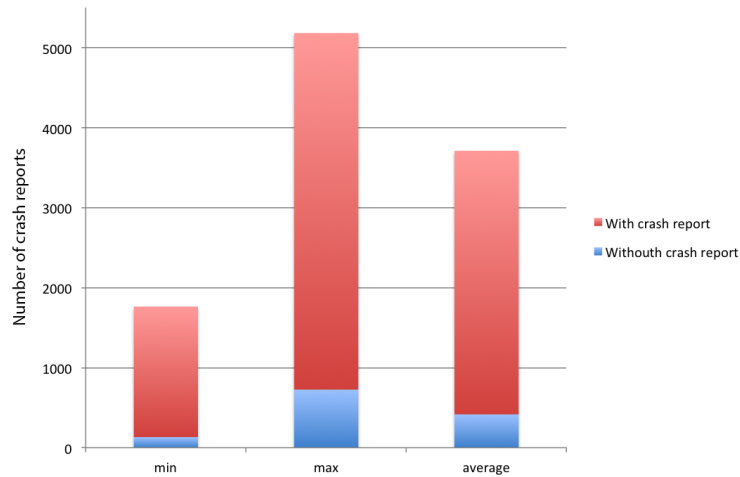
In Figure 6.2 results of a simulation of automatic crash reports is presented. Simulation was performed in following way:

By default every application that is published to the market has probability of up to 10% that developers will notice errors of a specific application, even if there are no crash reports from users. First, the amount of update events was captured if our system would not have automatic crash report system. Afterwards, automatic crash report module was added to the simulation. This module, keeps track of crash reports that users of the market submit to the global market entity, and informs developers about these events.

Results presented in Figure 6.2 show that after enabling automatic crash report



**Figure 6.1.** Rating distribution without implicit rating feature [App08b]



**Figure 6.2.** Automatic crash report contribution

module developer managed to collect 10 times more feedback events on average. These results show that automatic crash report is working correctly.

### 6.1.3 Malicious application removal

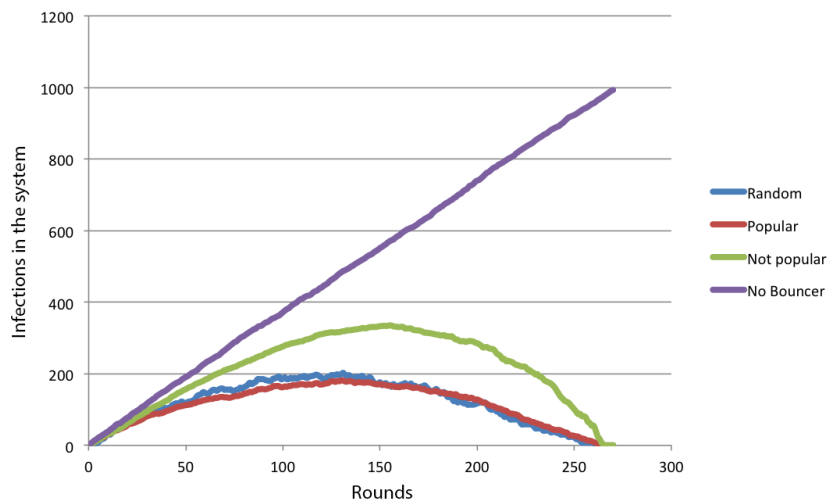
By testing bouncer feature, we were expecting it to remove malicious applications from houses that are already infected by them. During this test we tested three different strategies of a bouncer feature: random, popular and not popular. These



tests show how well malicious applications are detected with a specific bouncer strategy.

This test was performed with 100 active environments performing actions during 300 rounds. Out of 264 applications available in the market, 100 of them were set to be malicious. Simulation was repeated with the same parameters but with different bouncer strategies, to compare performance results of different strategies.

In figure 6.3 results of a bouncer feature evaluation is presented. During this test



**Figure 6.3.** Bouncer strategy effect

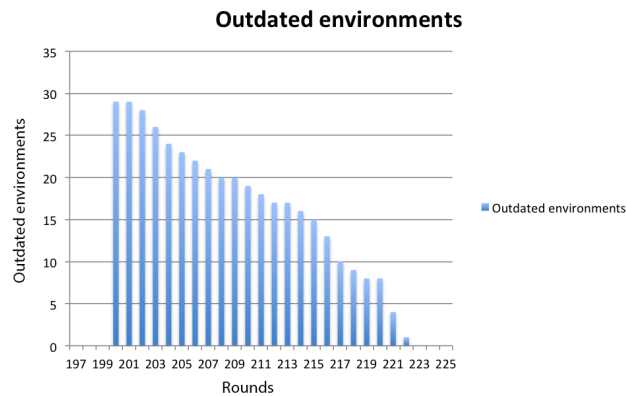
maximum amount of applications available on the market is 264, and bouncer feature is set to check one application per round, after 264 rounds number of infections reaches zero. This is happening because bouncer managed to check all markets applications.

Simulation results show, that automatic malicious application removal tool manages to cope with its goals - removing malicious application from the system. Also, strategy rules set to be "popular first" were proven to be the safest strategy, since it will keep number of infections in the systems on the lowest level.

#### 6.1.4 Update

By testing update mechanism we were expecting to see distribution of houses with new application version and older versions. To perform this test, system was running without updating any application for 200 rounds. In round 200, developers introduced updates for some of their applications. In Figure 6.4 results of this test are shown.

During this test, update interval of every environment was set to be once every 24 hours. Whenever new environment join the system, starting from this time, every 24 hours environment checks for updates. If new versions were found, environment



**Figure 6.4.** How fast all environments receive update after a new version is released

automatically requests required version from the market, and installs it. Simulation results show, that all environments managed to get latest application versions in at most 24-hour period. These results showed that automatic update feature is working.

## 6.2 Summary

In this section, the implementation of a core functions used in the market design were evaluated. The evaluation was performed by running simulation using custom made simulator for this specific system.

The experiments show that introduced implicit rating function provides automatic feedback from every user that is using the market, thus helping to determine applications quality more accurate, by rating it according to its algorithms.

Introduced automatic crash report function positively contributes to the application quality in the system by letting developer know about the crashes of their applications and helping them assign priorities on which application to update if any.

Evaluation of three different bouncer strategies show, that bouncer strategies set to random or popular application first helps to find and remove infections from the market and environments better than strategy policy set to not popular applications first. It was expected that bouncer strategy policy set to not popular application first will outperform other strategies, since it will remove infections when it is first installed on less environments. But, simulation results have proved that this assumption is wrong.

Automatic update mechanism is also showed to be working and helping to bring quality applications to every environment that uses the market in at most 24 rounds after a new version is introduced, if update interval of every environment is set to be once in every 24 hours.

**Part III**

**Conclusion**



## Chapter 7

# Conclusion & Future Work

### 7.1 Conclusion

In this thesis, a market for the smart environment is designed with the main goal to apply and adapt tools and mechanisms of existing application markets to the smart environments. The design uses features of already existing application markets for mobile phone environment, such as Google Play and Apples AppStore, adapted to work in the smart environment. Since applications in the smart environment are running unattended, in comparison with mobile phone environment, extra features are introduced. Automatic crash report feature, to provide more crash events to developers. Implicit rating feature, that rates applications based on users behavior automatically, thus giving feedback for every installed application. Automatic malicious applications removal feature, that removes malicious applications from environments, where such applications are installed, automatically. The system was evaluated in various simulation scenarios. Automatic crash report feature helps to get up to ten times more crash reports. Implicit rating feature provided all, at least once installed, applications with a feedback event, thus providing up to 50% more feedback events comparing to the current feedback distribution of Android Play market.

To summarize, the system design proposed in this thesis successfully provides quality applications to the smart environment by utilizing automatic crash report, implicit rating, automatic update and bouncer features in addition to the tools and mechanisms adapted from already existing application markets.

### 7.2 Future Work

**System design** Current system design does not utilize power of mobile devices and their mobile browsers. Even though the system operates well on mobile devices, fluid web design techniques combined with HTML5 technologies would make designed application market more user friendly on all platforms by automatically adjusting its style and control depending on the system and size of a screen it is

running on.

**Application recommendation** Application recommendation approach presented in this thesis can be improved by considering applications of the same categories. For example when user is navigating through games category, system recommends him/her only games that other users find out interesting or worth looking at. This improvement would lower amount of applications systems has to consider during construction of a recommendation lists.

**Implicit rating** Implicit rating mechanism can be further improved. If end-user, after uninstalling demo version of a specific application installs full version of that application, system should rate demo version with the highest possible option.

# Bibliography

- [Aim08] Joe Aimonetti. Android malware up 76 percent, nonexistent on ios - [http://reviews.cnet.com/8301-19512\\_7-20096832-233/android-malware-up-76-percent-nonexistent-on-ios/](http://reviews.cnet.com/8301-19512_7-20096832-233/android-malware-up-76-percent-nonexistent-on-ios/), 2012.08.
- [AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.
- [Ant11] G. Anthes. Invasion of the mobile apps. *Communications of the ACM*, 54(9):16–18, 2011.
- [app08a] Publishing your app for user testing - [https://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios\\_development\\_workflow/35-Distributing\\_Applications/distributing\\_applications.html#//apple\\_ref/doc/uid/TP40007959-CH10-SW1](https://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios_development_workflow/35-Distributing_Applications/distributing_applications.html#//apple_ref/doc/uid/TP40007959-CH10-SW1), 2012.08.
- [App08b] AppBrain. Ratings of apps on the android market - <http://www.appbrain.com/stats/android-app-ratings>, 2012.08.
- [Aut08] Authone. ds2os - [http://pahl.de/?site=\\_\\_research](http://pahl.de/?site=__research), 2012.08.
- [Bal09] P. Ballon. *Control and Value in Mobile Communications: A political economy of the reconfiguration of business models in the European mobile industry*. PhD thesis, Vrije Universiteit Brussel, 2009.
- [CBLW01] M. Claypool, D. Brown, P. Le, and M. Waseda. Inferring user interest. *Internet Computing, IEEE*, 5(6):32–39, nov/dec 2001.
- [CDS04] Cook, Diane Das, and Sajal. *Smart Environments : Technology, Protocols and Applications*. Wily, Hoboken, NJ, USA, 2004.
- [cI08] cnet Inc. Third-party iphone applications to arrive monday - [http://news.cnet.com/8301-13579\\_3-9958320-37.html](http://news.cnet.com/8301-13579_3-9958320-37.html), 2012.08.
- [DCGMV11] F. Di Cerbo, A. Girardello, F. Michahelles, and S. Voronkova. Detection of malicious applications on android os. *Computational Forensics*, pages 138–149, 2011.

- [DLL<sup>+</sup>10] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM, 2010.
- [DN06] C. Dellarocas and R. Narayan. What motivates consumers to review a product online? a study of the product-specific antecedents of online movie reviews. In *WISE*, 2006.
- [ea11] Michael Vakulenko et al. Mobile platforms: The clash of ecosystems. *Electronic Commerce Research and Applications*, page 68, 2011.
- [Gee05] D. Geer. Malicious bots threaten network security. *Computer*, 38(1):18–20, 2005.
- [GM10] A. Girardello and F. Michahelles. Explicit and implicit ratings for mobile applications. In *Workshop “Digitale Soziale Netze” and der*, volume 40, 2010.
- [HKR00] J.L. Herlocker, J.A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.
- [HKSS05] C.C. Han, R. Kumar, R. Shea, and M. Srivastava. Sensor network software update management: a survey. *International Journal of Network Management*, 15(4):283–294, 2005.
- [HO09] A. Holzer and J. Ondrus. Trends in mobile application development. In *Mobile Wireless Middleware, Operating Systems, and Applications-Workshops*, pages 55–64. Springer, 2009.
- [IICM11] S. Isaacman, S. Ioannidis, A. Chaintreau, and M. Martonosi. Distributed rating prediction in user generated content streams. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 69–76. ACM, 2011.
- [Inc08a] Amazon Inc. Amazon appstore for android - <http://www.amazon.com/b?node=2350149011>, 2012.08.
- [Inc08b] Amazon Inc. Step 5 in getting started with the amazon appstore - <http://www.amazon.com/gp/feature.html?docId=1000626391>, 2012.08.
- [Inc08c] Apple Inc. Apple app store - <http://www.apple.com/mac/app-store/>, 2012.08.
- [Inc08d] Apple Inc. Apple itunes - <http://www.apple.com/itunes/>, 2012.08.



- [Inc08e] Apple Inc. ios: Syncing with itunes - <http://support.apple.com/kb/HT1386>, 2012.08.
- [Inc08f] GetJar Inc. Getjar - <http://www.getjar.com/>, 2012.08.
- [Inc08g] Google Inc. The androidmanifest.xml file - <http://developer.android.com/guide/topics/manifest/manifest-intro.html>, 2012.08.
- [Inc08h] Google Inc. Paid app availability - <https://support.google.com/googleplay/bin/answer.py?hl=en&answer=143779>, 2012.08.
- [Inc08i] Google Inc. Publishing on google play - <http://developer.android.com/guide/publishing/publishing.html>, 2012.08.
- [Inc08j] Nokia Inc. How to install java me application in mobile phone - [http://www.developer.nokia.com/Community/Wiki/How\\_to\\_install\\_Java\\_ME\\_application\\_in\\_mobile\\_phone](http://www.developer.nokia.com/Community/Wiki/How_to_install_Java_ME_application_in_mobile_phone), 2012.08.
- [Kam08] Andrew Kameka. Google bans about 1<http://androinica.com/2009/08/google-bans-about-1-of-android-market-apps/>, 2012.08.
- [KASJ11] Heung-Nam Kim, Abdulmajeed Alkhalidi, Abdulmotaleb El Saddik, and Geun-Sik Jo. Collaborative user modeling with user-generated tags for social recommender systems. *Expert Systems with Applications*, 38(7):8488 – 8496, 2011.
- [KMM<sup>+</sup>97] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
- [KPS02] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security: Private Communication In A Public World, 2/e*. Prentice Hall, 2002.
- [KWKI03] H. Kanma, N. Wakabayashi, R. Kanazawa, and H. Ito. Home appliance control system over bluetooth with a cellular phone. *Consumer Electronics, IEEE Transactions on*, 49(4):1049–1053, 2003.
- [Lin08] LinuxPlanet. Linux package management: Keeping up with the times - <http://www.linuxplanet.com/linuxplanet/tutorials/4161/1>, 2012.08.
- [LSY03] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

- [Meu08] P. Meunier. Software transparency and purity. *Communications of the ACM*, 51(2):104–104, 2008.
- [MLLC10] E. Miluzzo, N.D. Lane, H. Lu, and A.T. Campbell. Research in the app store era: Experiences from the cenceme app deployment on the iphone. In *Proc. of The First International Workshop Research in the Large: Using App Stores, Markets, and other wide distribution channels in UbiComp research*, page 4, 2010.
- [MRK09] Dorothy Monekosso, Paolo Remagnino, and Yoshinori Kuno. *Intelligent Environments Methods, Algorithms and Applications*. Springer, 2009.
- [MVW08] S. Moncrieff, S. Venkatesh, and G. West. Dynamic privacy assessment in a smart house environment using multimodal sensing. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 5(2):10, 2008.
- [NKZ10] M. Nauman, S. Khan, and X. Zhang. Apex: Extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 328–332. ACM, 2010.
- [PNS<sup>+</sup>09] Marc-Oliver Pahl, Christoph Niedermeier, Mario Schuster, Andreas Müller, and Georg Carle. Detection of malicious applications on android os. *Knowledge-Based Middleware for Future Home Networks. In IEEE IFIP Wireless Days conference Paris, Paris, France*, 2009.
- [Ras08] Fahmida Y. Rashid. Google bouncer finds, blocks malicious apps from android market - <http://www.eweek.com/c/a/Security/Google-Bouncer-Finds-Blocks-Malicious-Apps-from-Android-Market-778912/>, 2012.08.
- [Reu08] Reuters.com. Apple sued over apps privacy issues; google may be next - <http://www.reuters.com/article/2010/12/28/us-apple-lawsuit-idUSTRE6BR1Y820101228>, 2012.08.
- [SCNSP10] S. Siersdorfer, S. Chelaru, W. Nejdl, and J. San Pedro. How useful are your comments?: analyzing and predicting youtube comments and comment ratings. In *Proceedings of the 19th international conference on World wide web*, pages 891–900. ACM, 2010.
- [SJCH12] KG Shin, X. Ju, Z. Chen, and X. Hu. Privacy protection for users of location-based services. *Wireless Communications, IEEE*, 19(2):30–39, 2012.

- [SKKR00] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167. ACM, 2000.
- [SKR01] J.B. Schafer, J.A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data mining and knowledge discovery*, 5(1):115–153, 2001.
- [SM95] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.
- [VG10] Pieter Ballon Vânia Gonçalves, Nils Walravens. How about an app store? *2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*, page 8, 2010.



# Appendices

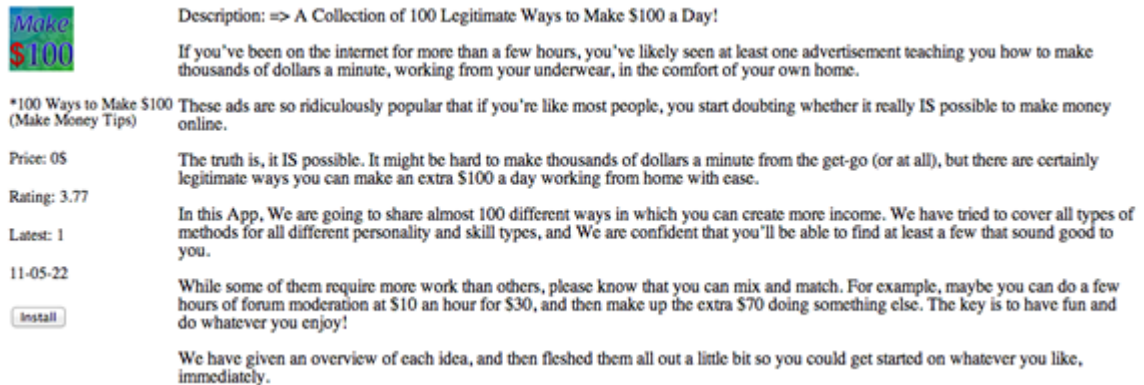


Version:

Permissions for this application goes here.

Copyright © 2012 TUM

Figure .1. Version compatibility check before installation



**Make \$100**

Description: => A Collection of 100 Legitimate Ways to Make \$100 a Day!

If you've been on the internet for more than a few hours, you've likely seen at least one advertisement teaching you how to make thousands of dollars a minute, working from your underwear, in the comfort of your own home.

\*100 Ways to Make \$100 (Make Money Tips) These ads are so ridiculously popular that if you're like most people, you start doubting whether it really IS possible to make money online.

Price: 0\$

Rating: 3.77

Latest: 1

11-05-22

The truth is, it IS possible. It might be hard to make thousands of dollars a minute from the get-go (or at all), but there are certainly legitimate ways you can make an extra \$100 a day working from home with ease.

In this App, We are going to share almost 100 different ways in which you can create more income. We have tried to cover all types of methods for all different personality and skill types, and We are confident that you'll be able to find at least a few that sound good to you.

While some of them require more work than others, please know that you can mix and match. For example, maybe you can do a few hours of forum moderation at \$10 an hour for \$30, and then make up the extra \$70 doing something else. The key is to have fun and do whatever you enjoy!

We have given an overview of each idea, and then fleshed them all out a little bit so you could get started on whatever you like, immediately.

Figure .2. Information presented to a user before installation



## A Market fo the Smart Environment

Welcome user1. User type:  
Environemnt ID: 1


Malicious

Copyright © 2012 TUM

Figure .3. Before uninstalling an application user can mark it as malicious

[Installed](#)
[Running](#)
[Featured](#)
[Top](#)
[Categories](#)
[Updates](#)
[All](#)

[Log out](#)



Alive Text - SMS/Email  
Groups & Templates

Price: 0.99\$

Rating: 4.09

Latest:  
10-01-22

5

\*\*\*\*\*  
 \*\*\*\*  
 \*\*\*  
 \*\*  
 \*

rate

[Uninstall](#)

Installed version: 1

[Update](#)

[ReportCrash](#)

Description: \*\*\*\*\*

iOS 5 iMessage ready on your  
 - i P h o n e  
 - i P a d  
 - i P o d touch

Start iMessaging with COOL SMS templates TODAY!

Create recipient Groups of friends, your football team, ANY group of people you can think of for SUPER FAST iM e s s a g i n g  
 \*\*\*\*\*

✓ Alive Text allows you to Create and Share Message Templates for you to access and modify on the Go, for super fast messaging with Groups, Family and Friends

✓ Alive Text has users in over 80 countries World Wide and is on the Top 100 Productivity list  
 ✓ Alive Text is a highly productive tool...

Usage:  
 - Select Favorite and Template Message  
 - Tap Text/SMS and SEND  
 - Tap Email and SEND  
 - Tap Skype, Paste and SEND  
 ...Its SUPER FAST

Description:  
 - Auto import users from Contacts  
 - Import Template Messages from our Web area <http://www.addalivetext.com>  
 - Share your templates with friends through Text/SMS or Email  
 - Send your current GPS position as a G... Map URL to friends and family by adding a text code  
 - Alive Text saves you valuable time and has a nice cover flow-style front end

Tips  
 ✓ Save and share frequent used messages with Alive Text  
 ✓ Save and share famous sentences with Alive Text  
 ✓ Save and share six word stories with Alive Text

Figure .4. Options available to a user after application is installed



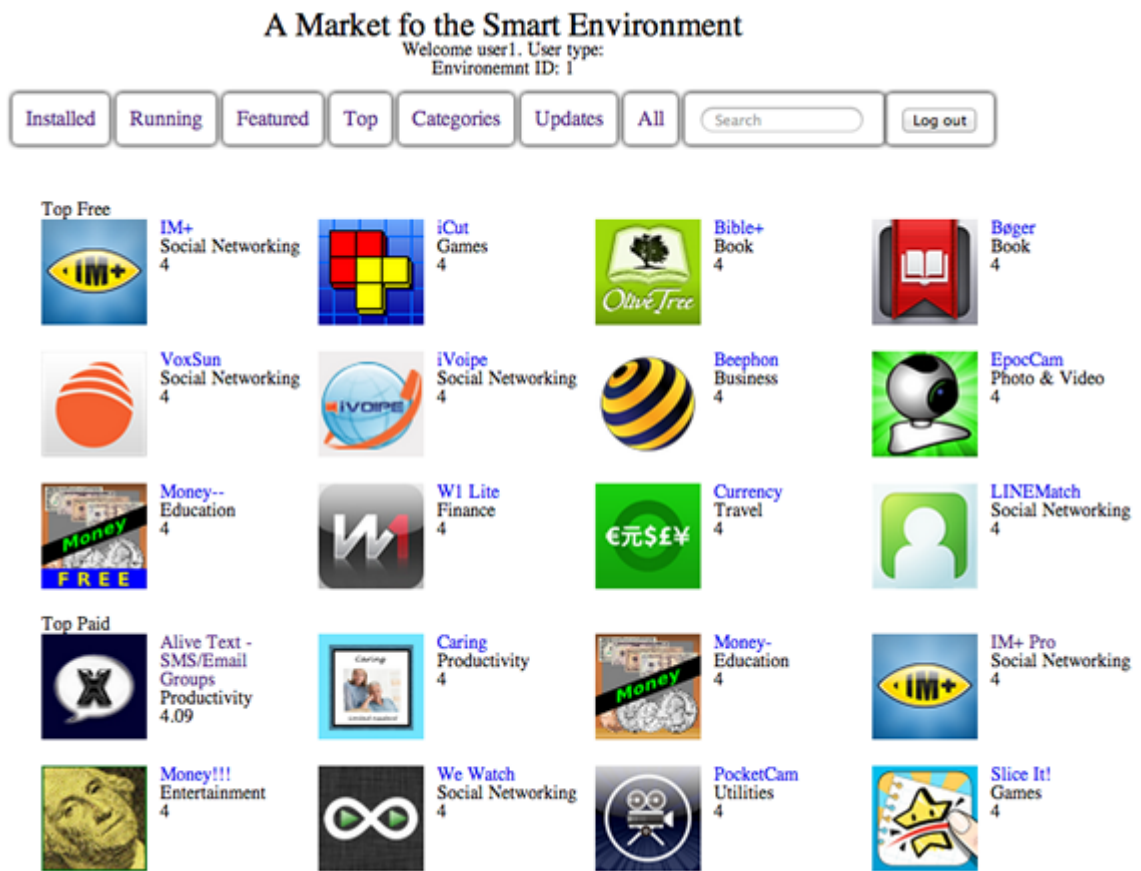


Figure .5. Top application list

**A Market for the Smart Environment**  
Welcome user1. User type:  
Environment ID: 1

	<b>Alive Text - SMS/Email Groups</b> Productivity 4.09		<b>Caring</b> Productivity 4.00		<b>Common Sense with Money</b> Lifestyle 4.00		<b>Counting Money</b> Education 4.00
	<b>Discover For iPad</b> Finance 4.00		<b>EpicCam</b> Photo & Video 4.00		<b>iDindi 2 HD - Money &amp; Expenses</b> Finance 4.00		<b>Internet-Telefonie selbst einr</b> Book 4.00
	<b>iTaxofon for iPad</b> Social Networking 4.00		<b>Mint.com Personal Finance</b> Finance 4.00		<b>Money Agent Lite</b> Business 4.00		<b>Money for iPad Free</b> Finance 4.00
	<b>Money Magnet Affirmation with</b> Productivity 4.00		<b>MONOPOLY for iPad</b> Games 4.00		<b>Penalty Soccer 2012 Euro</b> Games 4.00		<b>PocketMoney LITE - Checkbook</b> Finance 4.00
	<b>Shave Me!</b> Entertainment 4.00		<b>Slice It! Begins</b> Games 4.00		<b>StarMaker: Karaoke + Auto-Tune</b> Music 4.00		<b>W1 Lite</b> Finance 4.00

Copyright © 2012 TUM

Figure .6. List of applications that are already installed in the environment

---

## A Market for the Smart Environment

Please log in to the system

User name  Password:

Copyright © 2012 [TUM](#)

Figure .7. Agora login screen