nTag – an image management system on camera-equipped smartphones

Qi Hu

Kongens Lyngby 2008
IMM-MSC-2008-

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Abstract

This thesis presents an novel image management system – nTag. It targets smartphone users, who are supposed to confront the trouble in managing a large collection of images when the time of mobile image explosion arrives. The current version runs on Nokia S60 platform, works with a GUI and is controlled via conventional mobile phone numeric keypads. nTag provides users with tags to describe images, and images who have the same features are likely to be described by the same tag. In this way, nTag accomplishes image classification and organization using tags. Moreover, the nTag system also introduces a search engine helping users quickly locate images. It adopts a method dedicated to tag searching on numeric keypads, which is operated similarly to "T9" mobile input method, but is better than the latter.

The whole development process of nTag is a practice of software engineering theory, from the domain analysis until software implementation and test. Finally, nTag is evaluated by three means: a series of software testings, an usability experiment, and a functional comparison with other image management solutions on mobile devices. From these evaluations, it can be seen that nTag is a dependable and useful product with high performance. It can locate images 20% faster than Nokia N73's tagging solution and more than 4 times faster than conventional image browsing method on average. nTag is user friendly and people can easily put hands on it without the manual.

Preface

This thesis was prepared at the Center for Information and Communication Technologies in Department of Informatics and Mathematical Modelling, Technical University of Denmark in partial fulfillment of the requirements for acquiring the M.Sc. degree in engineering.

This M.Sc. thesis is the result of work carried out in the period from October 2007 to April 2008 with the amount of 30 ECTS points workload.

Lyngby, Denmark, April 2008

Qi Hu

For the great expectation in humble bodies – both of mobile devices and human beings.

Acknowledgements

First and foremost, I would like to thank my supervisor Jakob Eg Larsen for providing me this interesting topic and enlightening me during the entire thesis period plus the preparation phase. I especially thank him for those very helpful emails, many of which are coming out of his private time.

I would like to show my gratefulness to my best friends in DTU, those who have helped me during my worst time; those who have shared with me during their best time; and those, who have accompanied me during all our time in Denmark. Especially, to Da Gen, PP, Hans, Fu Rong and AnAn.

Also I have to thank my parents, no words are enough to express here, but at least thank you for enduring my absence for these three years.

And also thanks to all my teachers in my whole student life, since maybe it is my last moment of being a "professional" student. I really miss my teachers.

The last part is left for two girls: Cissy and Teresa. Thank you for being part of my life. You have given me a lot of important moments, which I regard as my treasure. I will never forget you. I silently put your names here, in memory of the past time.

Contents

Chapter 1

Introduction

Once you make the switch to digital, it is all too easy to get overwhelmed with the sheer number of photos you take with that itchy trigger finger. Albums, the principal way people go about organizing photos today, are great – until you get to 20 or 30 or 50 of them. They worked in the days of getting rolls of film developed, but the "album" metaphor is in desperate need of a Florida condo and full retirement.

– from **Flickr** website introduction [14]

1.1 Background

In the first quarter of 2008, Nokia, the international leader of mobile device industry, released its N96 smartphone[1] on the official website for N series products. This smartphone is expected to equip with a 5.0 megapixel camera with professional renowned Carl Zeiss optics, a 16G built-in memory, a luminous 2.8" QVGA display, and a perdurable power which can endure a 40 hours video time [30]. Don't you think this specification is too much for a portable, wireless telephone?

Not surprise to hear the news of N96, besides Nokia, almost all large mobile phone producers are competing to equip their products with higher resolution cameras, bigger storage memory, larger displays and longer perdurability battery [5][10][28]. Just as Nokia claims its N96 as a "multimedia computer", rather than a mobile phone, it reveals a fact that modern mobile phones have already evolved from simple telecommunication tools into high technique, multimedia equipments.

Among all the multimedia functions supported by mobile devices today, using of the embedded camera is one of the most popular and widespread. This judgment can be supported by many surveys on mobile devices market, and there are people believing that the prevailing mobile phone camera usage has brought a societal evolution [26]. Moreover, study of [33] shows that mobile phone owners almost always carry their devices, which gives people more chances to take pictures using the mobile phone camera than conventional cameras.

All the above evidence shows a trend, actually already being a fact, that in the scope of mobile phones(or mobile devices, smartphone, even "multimedia computer", whatever), image generating, storing and displaying becomes more and more easier, which will end up a huge amount of images on mobile phones. A report called *Mobile Imaging Study Results* [23] estimates that the total number of images captured on camera phones will exceed the number of photos taken on digital still cameras and film cameras combined by 2010. Such a potential of mobile phone image explosion arouses new challenges.

[1] The thesis uses the term "smartphone" to denote all advanced mobile phones which have PC-like functionality [3]

1.2 Problem and Aim

As mobile images keep on accumulating, it becomes a trouble for users to locate and organize them. The conventional solutions of mobile devices are presenting images one by one, or showing limited number of image in thumbnail size in a grid view, which will become infeasible if trying to find an image among thousands, because such methods are timely consuming, physically tiring and even psychologically annoying!

To deal with this problem, this thesis aims to present an image management system on camera-equipped smartphones – nTag[2], to help users manage large mobile image collections. The system will consider issues including hardware and softare limitation of mobile devices, as well as human habits. This new system will make the experience of mobile image locating and organizing faster, easier and happier.

1.3 Solution and Methodology

The new image management system uses tag[2] to build a structure for image organization on smartphones. Moreover, it introduces a new searching method, similar to T9 [11] mobile phone input technology, which can greatly shorten the image locating time and reduce users' the physical movements.

The thesis project uses software engineering theory to guideline the development of this system. Especially it applies the methodologies of domain engineering and requirement engineering from Dines Bjørner's software engineering triptych paradigm to the analysis process of this image management system. In the end, comparison is made with other application solutions from function and performance perspective, to evaluate this invention.

[2]The naming of nTag mimics Apple's iXyz style and "n" here denotes Nokia Nseries.

1.4 Thesis Structure

- *Chapter 2.* Study and delineate the domain where this case resides
- *Chapter 3.* Prescribe the requirements description
- *Chapter 4.* Design the nTag system
- *Chapter 5.* Implement the nTag system
- *Chapter 6.* Test the product and evaluate nTag
- *Chapter 7.* Highlight the contributions and present the future work

Chapter 2

Domain studying

At the beginning of this thesis, the domain of the topic is thoroughly studied and a domain description will be done in this chapter. By a domain, or the same, an application domain, indicates anything to which computing may be applied. Domain description tells the domain's entities, functions, events and behaviors, which together delineate the notion "domain" as it is [17].

By following the guildline of domain engineering, the first part of Dines Bjørner's triptych theory of software engineering, the thesis in this chapter describes the domain in these steps: first, sketching out the domain's phenomena and concepts; and then, identifying stakeholders; after that, outlining the business processes and, finally, figuring out the different domain facets.

In the process of this domain studying, the scope is managed to cover the issue of interest, which is image management on camera-quipped smartphones. Within the domain description, important properties such as entities, functions, events and behaviors, are identified to reveal their specialties. the result of domain studying provides a full and accurate gist, where researchers can find the main problems to solve and the goals to achieve.

2.1 Overview

In software engineering, requirements must be understood before a software is designed; however, further before the requirements are specified, it is foremost to understand the application domain [16]. the efforts paid here to understand the domain are the work on describing the domain in a consistent and relatively complete way. thesis will apply the domain engineering theory in [17] to this concrete case – image management system on camera-equipped smartphones, and the results of each description about the domain feature reflect the analysis and understanding on the domain of the case. therefore, in the following parts, these stages are executed to describe the domain:

- **Outline phenomena and concepts:** sketch out the entities, functions, events and behavior
- **Identification of domain stakeholders:** list out stakeholders of this domain and their perspectives
- **Identification of business processes:** identify collections of interrelated behaviors, which solve a particular issue respectively
- **Identification of domain facets:** present what a domain model contains

All section (except the first and last ones) and subsection titles as well as definitions in this chapter, if without specific references, are derived from theory characterizations in [17]. This intends to show how the thesis uses the theory to analyze the practical problems of the domain of interest.

2.2 Phenomena and Concepts

To describe a domain is to describe its observable phenomena and phenomena's concepts, which include entities, functions, events and behaviors. the following parts outline the these phenomena and concepts of an image management system on camera-equipped smartphones.

2.2.1 Entities

Entities represent some objects(or things) which can be abstracted to computer data with types and values.

Entities	Types	Descriptions
Hardware (HW)	HwW	working status
Camera(C)	CR	resolution
KeyPads (KP)	KL	layout
Key (K)	KSiz	size of each key
	KSta	physical status
Display (D)	DS	screen size
	DR	screen resolution
Memory (M)	MC	storage capacity
	MM	embedded or memory card
Data Transfer Equipment (DTE)	DteW	working status
Software (SW)	SwW	working status
Underlying System (US)	UsS	processing speed
	UsC	compatibility
	UsE	expandability
Data Structure (DS)	DsO	organization
Digital Image (DI)	ImgS	digital size
	ImgN	name
	ImgL	location in system
	ImgMeta	metadata
Image Set (IS)	IsC	capacity
	IsN	name
	IsL	location in system
	IsMeta	metadata

Table 2.1: Domain entities and their types

Main entities in the domain of this thesis, together with their interested types and type descriptions, are listed in Table 2.1. the table in divided into 3 subtable, the first entity contains the rest entities of in each subtable, where the latter are called sub-entities.

2.2.2 Functions

Functions are phenomena or concepts which apply to entities in order to test for some property, observe some subentity, or change the entity value. Functions in Table 2.2 are corresponding to entities in Table 2.1.Although such as *add image* can be seen as a behavior in that it first asks for a memory space to store the image data, and then creates an image data structure on underlying system; however, thesis regards it as a function to emphasize its influence on the Digital

Function names	Applied entities
add image	Digital Image
remove image	Digital Image
move image	Digital Image
edit image	Digital Image
locate image	Digital Image
view image	Digital Set
add set	Digital Set
remove set	Digital Set
move set	Digital Set
edit set	Digital Set
locate set	Digital Set
view set	Digital Set
sys. execute	Underlying System
sys. wait	Underlying System
take photo	Camera
download image	Data Transfer Equipment
keypads input	KeyPads
display	Display
mem. load	Memory
mem. save	Memory
mem. release	Memory

Table 2.2: Domain functions and their applying entities

Image entity and also to simply the complexity of this domain analysis.

2.2.3 Events

Events are phenomena which occur instantaneously and may convey information to affect parts inside or outside domain. Table 2.3 include main events which possibly happen in this domain.

2.2.4 Behaviors

the phenomena are behavior, when they proceed in timely by performing functions, generating or responding to events, or interacting with other behaviors. A behavior is a sequence of actions(functions which change the entity value) and events. In this domain main behaviors are listed in Table 2.4

Events	Explanations
no images	no images on the smartphone
no sets	no image, no image sets
image not found	target not at the assumed location
set not found	target not at the assumed location
memory full	no room for new image
photo time	an event stimulates taking photos
good images	find out interesting images on other resources
desire images	various reasons to retrieve images
demanding space	other applications request storage
image explosion	images in a set reach a certain level [25]
system occupied	other applications are using system
keypad malfunction	dead keys or any other abnormal situation
display malfunction	blue screen or any other abnormal situation

Table 2.3: Domain events

Behaviors	Series of Events and Actions
photograph	photo time, take photo
download	desire images, add image, add set
empty	demanding space, image explosion, remove image, remove set
search	desire images, locate image, view image
organize	image explosion, move image, move set, edit image, edit set
operate	photo time, good images, keypads input, sys. execute, display

Table 2.4: Domain behaviors and contained events and actions

2.3 Stakeholders

A domain stakeholder is a group of subjects who have common interest in or dependency on the domain. these subjects can be one or more persons as well as companies, governments, etc. Generally, there are two kinds of stakeholders:

- General application stakeholders
- Software development stakeholders

2.3.1 General application stakeholders

General application stakeholders are those who are not involved in the activities of IT business area. Since the images management system is a software product

and this thesis is not going to discuss the influence outside the scope of producing and using this system, the domain interested here is closely IT-related. therefore, no general application stakeholders in this domain.

2.3.2 Software development stakeholders

the image management system on camera-equipped smartphones targets many users or smartphones and intends to meet their needs(or potential needs) on smartphone image management. Thus the domain stakeholders consist of two categories: the producers and the users. Moreover, according to the way that users introduce new images to smartphones, the users stakeholder can be further divided into two parties: mobile photographers and mobile image downloaders.

2.3.2.1 Mobile photographers

This party of users are the authors of the their images on smartphones. One can be entitled as the "author" by the only means of taking photos with the camera equipped on the smarphones where images are stored. In other words, if the user takes a photo with a digital camera or with a embedded camera on another mobile phone, and then transfer this photo to the current smartphone, this user cannot belong to the mobile photographers stakeholder. On the other hand, this user is among the mobile downloaders.

2.3.2.2 Mobile downloaders

The term "mobile downloaders" is used in this thesis to refer to a general application stakeholder, which is a group of smarphone image users who get images from other resources. These images can be either downloaded from Internet or be transferred from other devices, but cannot be generated by the camera on the current smartphone.

2.3.2.3 Difference and overlap

In reality, it's more common that users are expected to be both of mobile photographers and mobile downloaders than users are only either of them. But the domain stakeholders are defined by their interest in or dependency on the domain. Therefore, it is not the users themselves but their certain behaviors

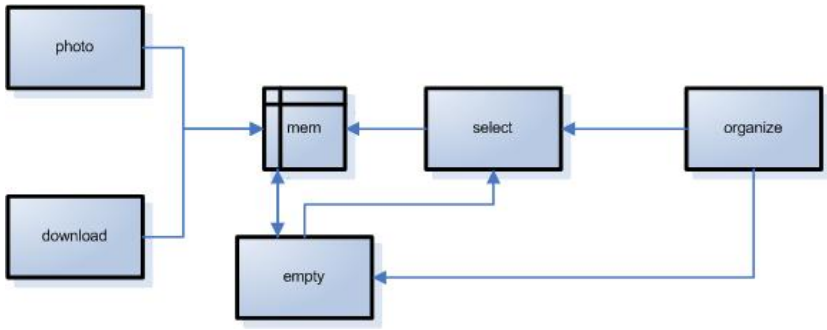

Figure 2.1: An image organization behavioral system abstraction

and other properties in the scope of this domain decide which stakeholders they belong to.

2.4 Business Processes

Business processes are sets of routines, which regularly occur in a certain forms of ways. In the domain of image management on camera-quipped smartphones, **Image Organization Business Process** is the business process that this thesis interests in.

To describe the **Image Organization Business Process**, the following description identifies a set of related behaviors of interest; entities characteristic of each behavior; functions that apply to or yielded from each entity; and events that each behavior shares with others.

The main business process behaviors of an image organization system are: *photograph and download, empty, search, organize* and *operate*. These behaviors can be found in Table 2.4. Søren S. Jensen in his master thesis [24] made a systematic study in the domain of information management and photo(paper and digital) management. The behavioral system abstraction in Figure 2.1 is drawn based upon Søren's description on the photo organization business process.

Activity descriptions of each behavior in this business process are presented as follows:

- **Photograph and Download**: when events *photo time* or *good images* happen, *photograph* and *download* behaviors generate new images on *sys-*

tem, which check *memory* whether the enough storage space is available and calls for *empty* to *remove image* if no enough space.

- **Empty**: remove image when behaviors *organize* need *mem. release*. Behavior of *search* helps to *locate image* or *locate set* for *empty*.
- **Search**: behavior *search* helps *empty* to locate the target to conduct its action on. *Search* also helps *organize* to locate targets.
- **Organize**: behavior *organize edit image* or *move image* when event *image explosion* happen. *Search* is engaged in *Organize* to *locate image* and *locate set*.
- **Operate**: Although not drawn in Figure 2.1, all the other behaviors need *operate* either to synchronize or communicate with each other, or to execute their own actions.

2.5 Domain Facets

A domain facet is one of a finite set of views, which respectively describe the different perspectives of the domain and together cover the whole domain. Basing on the description of **Image Organization Business Process** in the previous section, three domain facets: *intrinsics, support technology, human behavior*, are identified in this section.

2.5.1 Intrinsics

Domain intrinsics are basic to other facets and should cover at least one stakehoder perspective.

From the view of *mobile photographers* Image Organization involves operating camera to take pictures, instructing system to organize pictures, or even maybe to remove the new pictures.

From the view of *mobile downloaders* Image Organization involves operating data transfer equipment to receive images, instructing system to organize pictures, or discard receiving.

Table 2.5 shows the domain intrinsics from perspectives of mobile photographers and mobile downloaders respectively. The different places are highlighted, which are both regard to the generation of new images. The reason to keep these

Intrinsics	mobile photographers	mobile downloaders
Entities	**Camera**	**Data Transfer Equipment**
	KeyPads	KeyPads
	Display	Display
	Underlying System	Underlying System
	Digital Image	Digital Image
Functions	**take photo**	**download image**
	keypads input	keypads input
	display	display
	all image functions	all image functions
Events	**photo time**	**good image**
	image not found	image not found
	desire image	desire image
	image explosion	image explosion
Behaviors	**photograph**	**download**
	search	search
	organize	organize
	operate	operate

Table 2.5: Intrinsics in view of mobile photographers and mobile downloaders

domain features as intrinsics is because either they reflect the difference between difference stakeholders, or play a important role in Image Organization Business Process.

2.5.2 support technology

Søren in his thesis [24] made a survey and analysis on the technologies applied to image management on mobile devices.

The support technology prescribe the methods or tools which implement the domain features(entities, functions, behaviors and events). Considering the situation here, in the domain of image management on camera-equipped smartphones, the support technology concerns both mobile devices and image organization methodologies, which are introduced in below.

2.5.2.1 Mobile device issues

- Keypads: Qwerty keyboard are adopted by many smartphones, which resemble traditional PC keyboard. The common keypads of mobile devices

including smartphones consist of numeric keys, direction keys and a few other function keys. Soft keyboard is either a numeric or a qwerty keyboard displayed on screen. It is activated by touching on the screen.

- Touch screen: Touch screen replaces traditional LCD screen on some mobile devices [15], and this technique also eliminate the existence of hard keypads on these devices. A famous example is iPhone [20].

- Camera: Manufactures keep on increasing the resolution and techniques of the embedded camera on mobile devices [31], such as Nokia N96 has a 5 mega pixel camera with Carl Zeiss optics [30]

- Input system: With numeric keypads, T9 input system is a powerful tool [11]. On qwerty keyboard, or soft keyboard, the input method is the same as using PC keyboard. There is a technique called voice dialing, which uses voice to input.

- OS: Symbian, Linux, Windows Mobile, OS for iPhone are popular operating system on smartphones. The introduction and analysis between them will be covered in section 5.1.

- Presenting images: In Nokia and Sony Ericsson mobiles present images on screen in thumbnail size in form of list or grid, and support traversing images by stepping in. iPhone also can show multiple image thumbnail in grid, but image can be accessed by direct touching [20].

2.5.2.2 Image organization issues

These three structure candidates for organizing images are presented and discussed in [24]. They can be seen as technologies applied on methodology of implementing image organization:

- Hierarchies: each information entity has a single location labelled with controlled vocabularies and all these entities are organized in a hierarchical structure.

- Facets: each information entity is assigned with any number of facets, which are based on controlled vocabularies. Entities are organized in the way that they can be reached via any assigned facet.

- Tagging: each information entity is assigned with any number of tags, which are freely defined words or symbols. Entities are organized in the way that they can be reached via any assigned tag.

2.5.3 human behavior

Domain human behavior is to qualify how people carry out a task, since it is important to capture salient features of what it means to be a human factor. Users of the image management system is the human factor of this domain. Therefore, thesis assumes a user who uses the image management system to organize and operate images on his/her smartphone. Then:

- a user is characterized as organizational, if the user organizes every new image
- a user is characterized as systematic, if the user organizes images periodically
- a user is characterized as sloppy, if the user seldom organizes images
- a user is characterized as chaotic, if the user never organizes images

2.6 Summary and Goals

Walking through the previous sections, various aspects of this domain are identified. The domain stakeholders are specified as the interest groups of the image management system on camera equipped smartphone. Then domain's phenomena and concepts are sketched out in the form of entities, functions, events and behaviors. Based on this sketch, thesis further figures out the business process of image organization and examines it from different domain facets. These facets include intrinsics from different view of stakeholder, techniques on both devices and methodologies which play as the means and ways of implementing the phenomena and concepts, as well as the criteria to assess human behaviors involved in the business process.

With regard to this domain studying, the aim of this thesis is to create an image management system product that works on camera-equipped smartphones, under the scope that the domain description defines. This system cares about how to help users, who uses smartphones to take photos and collect new images, organize their images on the smartphones. Though these users may hold different attitudes and habits on the task of image organization, the thesis will expound and prove that the new system adopts the most suitable technique and implementation methods to improve smartphone image organization experience of one or more categories of users.

Chapter 3

Requirements Analysis

Before a system is designed and implemented, the designers and developers should have ideas in their minds about what exact they are going to design and implement. The "what exact" here is the requirements. This chapter's task is to prescribe a requirements description of the image management system on camera-equipped smartphones. The requirements analysis is based on the result of domain study, which is done in the previous chapter. Thesis will "transform" the domain description into the requirements prescription following the way of requirements engineering promoted by Dines Bjørner in [17]. The requirements prescription, as the result of this chapter's efforts, will play a role as the guidelines and constrains for the further work, including design, implementation, test and evaluation, etc.

3.1 Overview

Requirements engineering defines "machine" as the combination of hardware and software in the domain for the computing system development. Requirements should define what entities the machine should retain, what functions and behaviors the machine should offer and what events the machine should handle. [17]

In the domain study, thesis delineates phenomena and concepts of the domain in perspectives of stakeholders, business processes, as well as domain facets. These descriptions in different perspectives will be analyzed and thus be transformed to the ones with regard to requirements during the requirements engineering. This chapter will follow the steps shown below to accomplish the description changes from domain focus to requirements focus:

- Describe requirements stakeholders
- Business processes re-engineer
- Prescribe requirements facets

Section (except the first one), subsection and sub-subsection titles as well as definitions in this chapter, if without specific references, are derived from theory characterizations in [17]. The intention is to show how to use the theory to analyze the concrete case in this thesis.

3.2 Requirements Stakeholders

Producers, mobile photographers and mobile downloaders are the domain stakeholders which have been defined for the domain of the image management system. In the perspective of requirements engineering, these stakeholders are classified into two groups: customers driven and market driven.

3.2.1 Customer driven stakeholders

The mobile photographers and mobile downloaders are primarily interested in using the system to help them manage their images on smartphones, i.e. organizing and searching images, and their requirements should show the desire for

their "own" custom-made software. In the requirements prescription, they are called customer driven stakeholders.

Since the purpose of this thesis is to create a useful smartphone embedded software to deal with the potential image management problem, focus will be put on the "needs" and "desire" of mobile photographers and mobile downloaders. Therefore, the rest of requirements prescription is primarily regarding to this category of requirements stakeholders.

3.2.2 Market driven stakeholders

The producers who are classified to market driven stakeholders care more about the developing(the process and management of developing itself), marketing and selling. The discussion of this category of stakeholders has to be binded to a specific groups of producers. It is also required to have knowledge of their developing, marketing and selling strategies. None of the above two factors are the interest of this thesis.

3.3 Business Processes Re-engineering

The business processes re-engineering means reformulating the business processes described in the domain studying. If there is no changes with regard to the original business processes, neither re-engineering is needed, nor corresponding requirements. Will this thesis challenge the traditional business process? The answer is confirmative.

From the study in [24], camera-equipped smartphones are different from either traditional cameras or digital cameras regarding to the time-gap. Time-gap is the delay between pictures are taken and are organized. Before pictures are properly organized, paper photos have to wait for being printed out from the films, and digital photos have to wait for being transferred to a PC; however, the photos(also images) on camera-equipped smartphones are possible to be organized immediately when they are generated, which indicates with little time-gap. Moreover, thesis [24] has explained the advantage of shrinking the time-gap. Therefore, this thesis re-engineers the business process of image organization described in Section 2.4. The behavior abstraction diagram of the new business process is shown in Figure 3.1, where demonstrates this business process provides a chance of advancing the organization activity immediately after image generation.

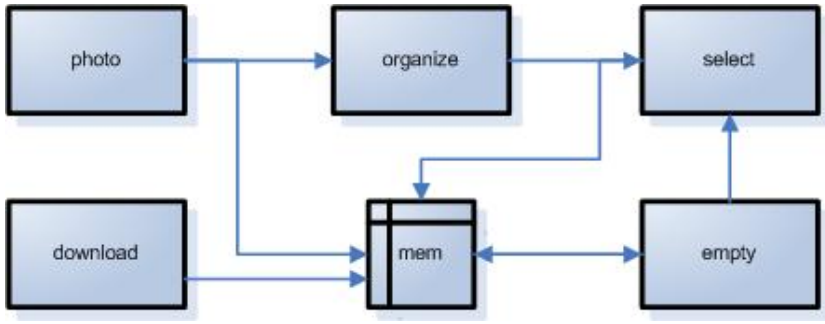

Figure 3.1: An image organization behavioral system abstraction

3.4 Requirements Facets

The requirements can be decomposed into four facets and each of them contributes constrains and guidelines to the further development in a specific perspective, namely:

- **Business processes re-engineering requirements**
- **Domain requirements**
- **Interface requirements**
- **Machine requirements**

3.4.1 Business processes re-engineering requirements

This requirements is drawn to examine, review and replace(if necessary) every description of domain facets from the original business processes. Since the re-engineering of this thesis only adjust the sequence of some behaviors; however, the intrinsics, support technologies and human behaviors are not going to change. Therefore, there is no business processes re-engineering requirements for this project.

3.4.2 Domain requirements

Domain requirements specify which parts of the domain should be emulated by the system and in what *"shape, forms and contents"* [17]. The specification is

cut into five pieces and they are elaborated one by one in the rest of this part.

3.4.2.1 Domain projection

Thesis uses domain projection to narrow down the domain and only presents the issues of interest in the rest of requirements prescription and software implementation. The projected domain of nTag image management system is only:

- about applying computing system to the image organization business processes, but not about the improving and implementing the hardware attributes or underlying system's functions.
- about the entities, functions, behaviors and events with regard to the images which are existing on the smartphone, not about images out of smartphone, e.g. neither images uploaded to Internet, nor images before downloaded to the smartphone.
- about the entities, functions, behaviors and events with regard to the camera which is under control in the environment of this image management system, not about the camera out of this system, i.e. neither the camera's action will activate the image management system, nor the system monitors the camera out of it.
- about using **tag** as the main tool to build the image organization structure, not focusing on taxonomy using hierarchy or facet structure.

The reason to constrains the above issues is to concentrate limited time and efforts of this thesis project studying on the topic of personal mobile images management. Hardware and underlying system changes, however, should also consider other mobile application usages; images upload and download involves Internet, telecommunication protocol as well as issues of interfaces with other equipments; camera-nTag activation and monitoring functions concentrate more on camera configuration and software integration issues. In a word, all of them are diverted from the core interest of "IMAGE MANAGEMENT ON smartphones". Therefore, thesis leaves them as future work, discussed in 7.2.

Coming to the organization structure, [24] has discussed and compared strategies, based on tag, hierarchy and facets, for structuring and organizing photo collections, and it comes out that tag beats other when applied on mobile images, for its less strict requirements of orthogonality, freely defining keywords to describe image contents, as well as supporting multiple classification. Therefore, the following requirements agree on that tag based strategy is adopted for the

new image management system presented in this thesis. Other structures will not be considered into the requirements.

3.4.2.2 Domain determination

In this part, functions and behaviors are further specified during the process of domain determination so that they are predictable.

- the *search* behavior should execute in the way that users input features of the target, then system executes and returns target images with metadata.
- the function *take photo* should activate the camera and give the control to users. When a photo is taken, it triggers the rest behaviors in the image organization business process.
- the scope of images can be organized should include all images in different folders on the smartphone.

3.4.2.3 Domain instantiation

Domain instantiation further constrains domain description:

- the number of images for processing should reach 100 in the experiment release version software system, and this number should reach as far as 8,000 in the formal release version software system. The number 8,000 is estimated based on the fact that Nokia N95 8G has a 8G built-in memory and can also plug a 8G mini SD card, so the capacity will reach 16G, which can store 16,000 1mb-large images if without considering other applications usage.
- the number of unique tags can be processed should reach at most 150 in the experiment release version software system, and this number should reach as far as 500 in the formal release version software system. These two number are estimated based on the study in [34]. This study shows that more than 80% tag users only use less than 150 unique tags, and only less than 5% users use more than 500 unique tags.
- the number of tags mapped to an image should reach 10 in the experiment release version software system, and this number should reach as far as 100 in the formal release version software system. These two number are

estimated based on the analysis in [24]. It references a study showing that around 60% tag users only use less than 10 tags to map a single image, and more than 90% users choose less than 100 tags, while the rest a few percentage of users can consume more than 100 tags.

3.4.2.4 Domain fitting

These requirements prescribe the request on interfaces to other areas of domains that have already been readily established. For the domain of this thesis concerns, new image management system should be compatible with other smartphone applications. The following domain fitting requirements define how to achieve the compatibility:

- application system should still function and satisfy other requirements, if the images organized by this system have been deleted, removed or new images have been added to the smartphone in ways that are outside this system.
- application system runs conforming to other requirements, should not result in breaking other image management systems on the same smartphone, if there is any.
- application system should still function and satisfy other requirements, if other applications on the smartphone are running concurrently.

3.4.3 Interface requirements

The interface requirements care about the shared parts between the domain and the machine, where the machine is defined as the combination of hardware and software. In this thesis, a machine is a smartphone equipped with the new image management system and thus the interfaces are shared parts between the smartphone and its users. These interface requirements for nTag image management system are prescribed in the following parts from different views.

3.4.3.1 Man-machine dialog

These requirements define the syntax and semantics of communication between human and machine:

- when users operate the machine, the latter should respond and send feedback
- when the machine sends information to users, the latter should operate the machine to continue an action or should give up further operation to stop the communication
- when the machine sends information to users, the information should be a language(or symbols, etc.) which can only be explained in one way in the context, i.e. without equivocal explanations.

The first two requirements establish the communication syntax between users and smartphones: the machine has no choice but respond users, however, users receive feedback and decide continue or end the communication. The last requirement defines the communication semantics which prevents machine's feedback from confusing users' judgment.

3.4.3.2 Man-machine physiological interface

This part is about which technologies should be used to execute the dialogs between human and machine:

- the system should use GUI to send feedback to users
- the system should use numeric keypads to receive operation from users

The reason why only uses numeric keypads is because the numeric keypads are the basic style of mobile devices input tools. Since no strong evidences have shown that any other input technologies(qwerty keyboards, touch screen soft keyboard, or voice input) will replace the numeric keypads, for the consideration of universality, this system uses numeric keypads.

3.4.3.3 Computational data and control interface requirements

These requirements define the input forms which control the flow of computation:

- users start controlling the machine by operating the keypads and end controlling the machine by stopping operating the keypads. The "operating" here means any operation on keypads.

3.4.3.4 Shared data initialization

As it is defined in other requirements that the system uses tag as image organization structure and uses GUI to convey information, data structures include images, tags, metadata, menus, warning and notes, as well as focuses should be initialized on the machine.

3.4.3.5 Shared data refreshment

Share data refreshment requirements prescribe the frequency and methods of refreshing shared data. Therefore, according to the shared data initialization requirements, their refreshment requirements are:

- images refresh when images are operated by means of deleting, generating and moving
- tags refresh when tags are operated by means of deleting and generating
- metadata refresh when metadata are operated by means of deleting, generating and editing
- menus refresh when the GUI refreshes
- warning and notes refresh when the information they convey refreshes
- focuses refresh when users control them to change

3.4.4 Machine requirements

Machine requirements are only restrains on machine itself and can be obtained from five facets: performance requirements, dependability requirements, maintenance requirements, platform requirements and documentation requirements. Applying them to the topic of this thesis, the machine requirements are explained and listed in the following parts.

3.4.4.1 Performance requirements

These requirements confine the machine's consumption in any form:

- the system should be used by one user at a time, i.e. supporting neither concurrent usage, nor distributed usage
- the average response time should be at most 0.1 second for basic operation, including refresh focus, show feedback of input typing; at most 1 second for GUI menu invocation; at most 10 seconds for displaying images(not including GUI plaform) and database query related operations.
- the power consumption for a day should be less than 100% of a full power battery's energy.
- the storage consumption(not including the consumption by images) should be less than 10Mb.

The figures of requirements on response time come from article [27], which explains that when response time is 0.1 second, people regard the system reacting instantaneously; when response time is 1.0 second, people can feel the delay but won't feel interrupted; if response time is over 10 seconds, people will lose their attention on the system. [12] suggests 0.1 second response time is used for keypress and movement; 1 second for completing ordinary operations, such as closing windows; the 10 seconds limit should be used to graphics displaying and complicated tasks.

Power consumption should be guarantee a full day work, so that users can recharge after the work without interruption. The 10 Mb storage consumption is moderate, which is only comparable to 3 images under the largest resolution in Nokia N95. Once the smartphone can process hundreds or thousands of 1Mb size images, this figure for storage consumption of image management system could be greatly enlarged.

3.4.4.2 Dependability requirements

Dependability is the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers [7]. The requirements on the relevant dependability attributes of the image management system are:

- Accessibility - guarantee the fairness of access for multiple users: Not applicable, because nTag is a system for only one user at a time.
- Availability - guarantee the system can achieve time to process in certain time interval: no matter which image searching, no matter how many,

users should be expected to get either a result or a note telling this execution is not applicable, within the time span confined in performance requirements.

- Integrity - describe a system has no faults, errors, failures and it is not affected by environment's impairment: the nTag system is not required to have this attribute
- Reliability - it is a time which measures continues correct service: 30 minutes of correct function is for this attribute.
- Robustness: define how much the system retains its attributes after failure and maintenance: the records of tags, images and other information still retain their attributes after failure. If the nTag fails, it should not affect other application attributes.

3.4.4.3 Platform requirements

The requirements in this part are:

- the application system should run on one of the most common software platform, which will discuss in 7.1
- the application system should run on the hardware platform containing numeric keypads

  The reason for the above two requirements regarding to platform is that nTag system is targeting potential users, once the image explosion arrives. Therefore, it should reside on one of the most popular software platform to guarantee a market. Coming to the numeric keypads, it has been explained in section sec:phys of this thesis.

3.4.4.4 Documentation requirements

There are also some documentation requirements to safeguard the quality of this software. This project will write and maintain the following documents:

- codes and development documentation[1]
- domain description, see chapter 2

[1] Discuss with author for the detailed coding and test documentation

- requirement description, see this chapter
- test documentation[1]
- user manual, see Appendix B

3.5 Summary

This chapter has re-examined the domain descriptions and has transformed it into requirements prescription. Requirements are analyzed and prescribed from many views, including domain requirements, interface requirements and machine requirements. All of these requirements provide an aim for software developers, who have to bear in mind that these requirements are the criteria to evaluate the software product. Therefore, the next chapter will design the nTag based on the result of requirements prescription.

[1] Discuss with author for the detailed coding and test documentation

Chapter 4

Design of nTag System

In the previous chapters, the idea of a mobile image management system is thoroughly analyzed from its domain aspect. Furthermore, based on the domain description a detailed requirements prescription is generated. Therefore, to flow the process of software engineering, the task of this chapter is to design a camera smartphones based image management system, which is named "nTag" by the author of this thesis.

As it has been mentioned at the very beginning, nTag is designed specifically to solve the potential management problems aroused by mobile image explosion. Therefore the smartphone, including both hardware and software environment, serves as the platform which nTag is based on. nTag will use "tag" as the main tool to structure the image organization and adopt smartphone's built-in hardware equipments such as display and numeric keypads to facilitate the communication between human and machine.

4.1 System Overview

The process of design nTag, is the process of establishing a model, which directs the implementation towards the goal set by requirements. Therefore, all the design of this system is driven by the prescription of requirements. At requirements analysis phase, section 3.4.2.1 domain projection requirements have decided the tag-based organization strategy.

nTag is going to utilize the advantages of tag based taxonomy: tags can be freely user defined keywords, tags can be used to describe the contents of images, as well as tags enable a multiple classification [24]. When organizing mobile images, users of nTag are expected to choose or define one or more keywords to help them characterize images and use these keywords as clues to locate the images in the future. In this scenario, there are three entities(use cases) involved: tag, image, and the relation between them. Consequently, nTag builds its function modules from these three use cases, namely: tag repository, image repository and tag-image link library, see Figure 4.1 Diagram A.

Using tags to describe and reference images is the basic scheme of nTag; however, if tags are overwhelming, the trouble of navigating a tag among thousands is exactly the same with the trouble of locating a mobile image in a large collection. Although domain instantiation requirements(see section 3.4.2.3) restrict the number of tags to 500 at most, this figure is already large enough to scary away nTag's potential users if they have to browse the whole tag list to find a tag. Therefore, to cope with this problem, nTag provides a tag search engine. This search engine serves as an interface between tag repository and users. The user case diagram from A transforms to B in Figure 4.1, after the introduction of the tag search engine.

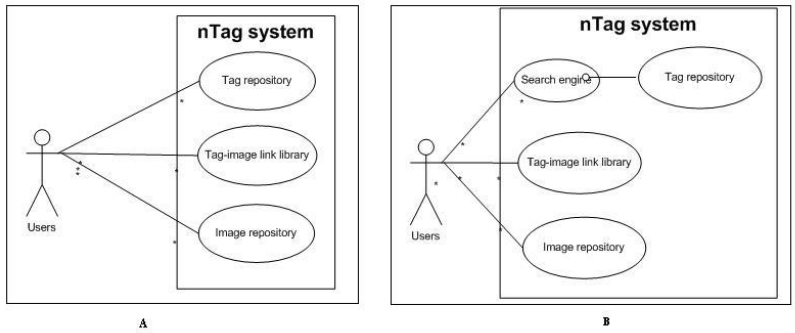

Figure 4.1: User case diagram of nTag system

In the following sections, illustrations will go into details about the above functions modules in nTag. Firstly, thesis describes the tag repository construction and management. Secondly, image repository is illustrated in a similar way. Thirdly, focus will be put on how to establish and manage the relation between tags and images. Finally, the passage deals with tag search engine.

4.2 Tag Repository

Tag repository contains all tag entities and functions applied to these tags. In nTag, there two categories of tags in form of their generation:

- System auto generated tags
- User defined tags

The name of the first category of tags is self explanatory – they are auto generated y the nTag system. These tags describe the contextual information and they are added to images by the system. System auto generated tags, or put it short – system tags, serve as the basic resource for image organizing. They are especially necessary in the situations, when users do not have time or do not bother to assign tags to images. So with the aid of these tags, system can still hold the clue to the images.

The contextual information could be date, time, user profiles, calendar events, time zones as well as geographic locations (gain from mobile base locations or GPS, whatever). These information are helpful to systematically classify images; however, if there are only system tags, it is not enough to convey the descriptions about the contents of images. Moreover, system tags are chosen by the system instead of users. One of the most important reasons of using tags to construct the image organization is because they can be freely defined by users. Therefore, nTag system cannot live without the other category of tags – user defined tags.

The user defined tags are created by users. These tags can be used to describe anything information of the images and it is only a matter of their creators' decision. They are freely to be attached or removed from images(not like system tags, which are controlled by system), as well as to be easily deleted just as easy as their naissance. User defined tags are stored in a flat structure, which means there are no hierarchies between tags. No duplicated tags, but one tag can be used to map as many images as possible. These tags in nTag system are

presented to users in a textual form and they can be fetched through the tag search engine swiftly. The design of tag search engine will be explained in section 4.5.2

One special case is that before users define their own tags, they may be provided by the system vendors with some already defined tags. These tags can be called default tags; however, they are exactly identical to the user defined tags from both functional and operational aspects. That is the reason why default tags are not extracted as another category. One idea to make them special is to organize them in a hierarchical structure and permit users to customize this structure; however, it is not in the requirements of this thesis(see section 3.4.2.1), so this could also be one of the future work 7.2.

4.3 Image Repository

The image repository is not the physical container of a smartphone's digial images, but an abstract unit which contains image entities, corresponding functions and takes responsibility in reacting to events and trigger behaviors.

It monitors the physical collection of images and update its image information records if any image's path or name have been altered because of events or functions being applied to images. These events and functions include taking new photos, adding images to or removing images from the mobile phone, switching images to different folders, as well as simply changing image names (can be found in domain studying, section 2.2). It is necessary to keep such monitor and update, because nTag system has to guarantee that the correctness of reference between tags and images cannot be impaired by image operations, no matter they are carried inside or outside of nTag system – domain fitting requirements, see section 2.2.

4.4 Mappings of Tags and Images

Tag-image link library manages mappings(links) between tags and images. This module connects tag repository and image repository through free mappings: a tag can be attached to many images while an image can be referenced by many tags. Nonetheless, a tag(except system tags) doesn't have to map any image; however, an image should be attached at least by one system tag. These are shown in Figure 4.2, where **tag4** has no mapping, but img3 has to be attached by sys.tag2.

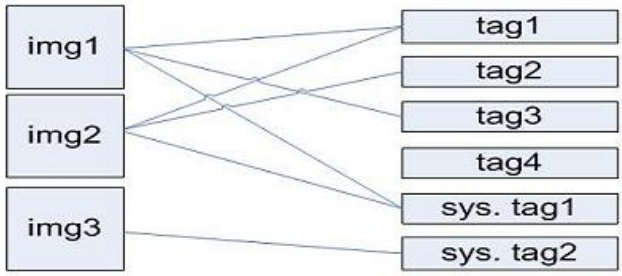

Figure 4.2: Diagram of mapping logic

Link library is accessed by either end of a link: from a tag or from an image. Links stored in library are established on the basis of tag-image pair, which means if either end of a link is not longer available, the link breaks and its record is removed from the library. It is not necessary on the contrary, however. Tags can exist without any link, while the bottom line for images is the only mapped system tags. Therefore, in nTag, Tag-image link library is embedded between tag repository and image repository.

In tag repository, when users name a tag, link library immediately gathers all links originating from this tag and go to image repository to ask for all images connected to these links. If this the tag doesn't has any link, link library will return a feedback to inform this situation to tag repository. The sequence of these operations is shown in Figure 4.3.

If a tag is to be deleted from tag repository, tag repository has to inform link library first, then the latter will check if there is any link connected to this tag. Link library must delete all relevant links before it permits tag repository to delete the tag. The sequence of tag deleting operations is shown in Figure 4.4.

In image repository, when users view an image, they can let the image ask to establish new links between one or more tags. In this case, link library will go to tag repository, find out all requested tags and register their mappings with this image. An image can also abandon links between one or more tags, which needs link library to delete mapping records of these tags with this image. The sequences of operations on creating and abandoning links are shown in Figure 4.5.

Once an image's information has been deleted from image repository, who has to inform link library to delete all links connected to this image. The sequence of this operation is similar to the sequence shown in Figure 4.4, just replace tag repository to image repository.

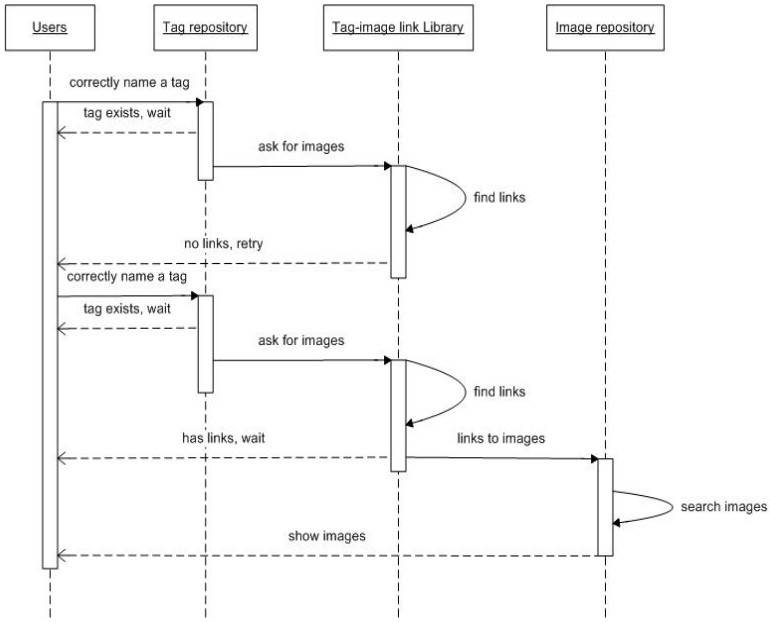

Figure 4.3: Sequence diagram of referencing images by tags

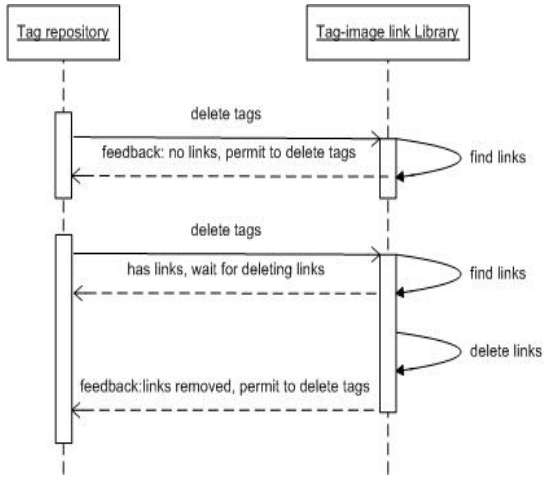

Figure 4.4: Sequence diagram of deleting tags

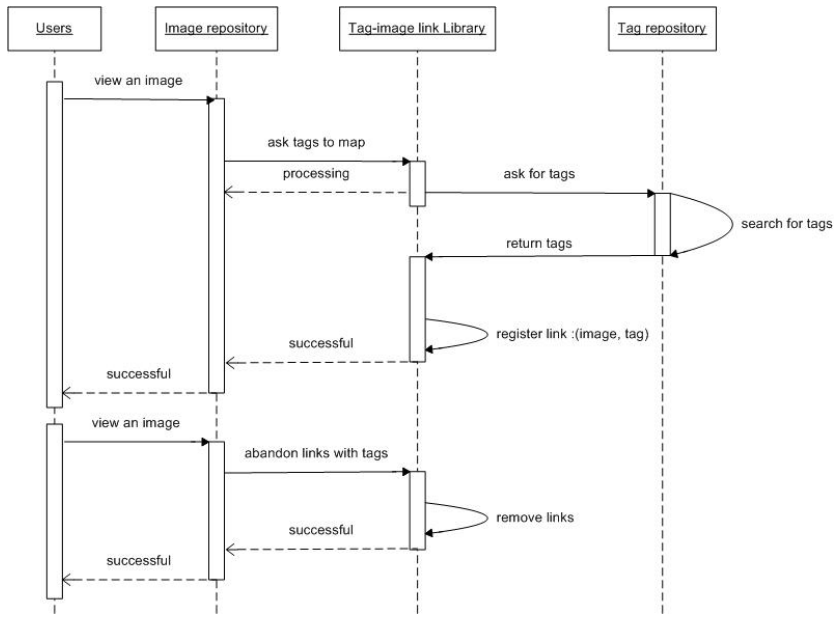

Figure 4.5: Sequence diagram of creating and abandoning links

4.5 Tag Search Engine

4.5.1 Candidate searching methods

As tags are organized in a flat way, when the amount increases, the efforts to locate a tag also increase, the complexity is $O(\mathrm{n})$.

There are several ways to locate a tag; however, nTag is going to pick up the most suitable one which can fit the requirements prescription as well as the big picture of the design. One intuitive idea to find a tag is by browsing. But because tags are organized in a flat structure, the complexity of this method is $O(\mathrm{n})$, which means when the amount of tags increases, the efforts to locate a tag linearly increase. As thesis mentioned in section 4.1, nTag will lose its value if it adopts browsing as the search method to locate a tag, even the number of tags hasn't passed the limit of requirements.

One alternative to browsing tags method is to call for a tag by typing its name directly. It is intuitive; however, there are also problems for this solution. First of all, users cannot be supposed to remember all tag names precisely, especially

when there are scores or hundreds of tags. But to correctly call for a tag by this method, one has to avoid spelling errors. Moreover, typing a tag's full name by using mobile phone's numeric keypads requires many actions. Even T9 input method can reduce key presses [11], the keypresses time is at least as long as the length of the word. Once tag names are not meaningful words(because tag names are freely defined), T9 cannot do anything about it, e.g. words such as "b6", etc.

4.5.2 nTag's searching method

nTag's method is enlightened by both of the above two candidate solutions, especially the T9 input scheme. This solution is based on the intuitive idea of searching tags by calling their names, but instead of typing the full name, it only needs the sequence of letters in tags' names.

Mobile phone users input a tag name by assuming they are typing keys on a qwerty keypad, without differentiating letters on every numeric key, i.e. no need to distinguish "a" and "b" but just use the numeric key "2" ("a" denotes both "A" and "a" in the following examples about this method). Before completing the spelling of a whole word, each key press generates a new string. Search engine looks up the tag repository and return tag names beginning with this new string. Therefore, the searching executes after each key has been pressed, and after several key presses, this method will gradually narrows the qualified tag names down to the target. The philosophy of this method is shown by Table 4.2, in which the tag "animal" is found by generating 3 keypresses "264".

tag names
actions
American
animals
b6
Bxz
bye
can
cn

Table 4.1: Tag list

It is also allowed to step back in the whole procedure of searching and the previous rounds of results are returned when this case happens. After executing a stepping back operation, user can choose to proceed with any key, or to step back further. No limits on the times of stepping back and proceeding with any

Keypress	Numeric Key	Key Sequence	Returned Results
1st.	2	2	Table 4.1
2nd.	6	26	American animals b6 cn
3rd.	4	264	animals

Table 4.2: A full process of searching for the tag "animal"

key. But to mention, it is not meaningful to step back when it is already the beginning of key presses(before the starting of a keypress sequences).

The process of changing keys is shown in Table 4.3, in which the 4th. keypress is an stepping back operation, so the results of the previous round of searching is shown, which are returned results of the searching after the 2th. keypress. Then in the 5th. keypress, it changes the key from "4" in the 3rd. keypress to "3" and get the final result tag "American" instead of the tag "animals".

Keypress	Numeric Key	Key Sequence	Returned Results
1st.	2	2	Table 4.1
2nd.	6	26	American animals b6 cn
3rd.	4	264	animals
4th.	step back	26	American animals b6 cn
5th.	3	263	American

Table 4.3: A process of using "step back" operation to change keys

4.5.3 Advantages of nTag's searching method

But what the difference between this searching method and directly inputing a tag with T9 input method? The difference is nTag's method is faster, suggestive and can handle "non-word" tag names. In the scheme of nTag searching method, every round of tag name searching is carried out only among available tag names in repository. It does not return every possible combination of letters generated by numeric key sequence, but only the existent tag names – no matter it is a

meaningful word or not(T9 returns possible combination of letters which can construct a word). For example, In Table 4.2, if the 1st. keypress is key "3", because there is no tag name beginning with "d", "e", "f" or "3", no results are returned, but a feedback telling that no qualified results available. It also prevents users from further searching based on the sequence beginning with key "3". To explain the advantage of dealing with non-meaningful word, let's go back to the previous example of "b6": if a tag is called "b6", T9 cannot provide you this choice by pressing numeric keys "8" and "8" in sequence, however, nTag can! See the returned results after the 2nd. keypress in Table 4.2.

nTag's tag searching is suggestive because it allows users to see all candidates from the returned results. Users have to remember the spelling of the whole word when using T9, but here, usually only a few of the beginning letters of the tag name are enough, e.g. nTag returns "itisquitealongword" among candidate results when users just press the key "4". See Table 4.2 and Table 4.3 for more examples. More than that, in every round of returned results, tag names are arranged in alphabet order. Such arrangement is quite convenient because users don't have wait until the last minute, when the target tag is the only one left in the result list. Users, instead, can use the conventional browsing method to locate the target as soon as the number of returned candidate results is small enough, e.g. in Table 4.3 users can easily find out tag "American" among returned results after the 2nd. keypress, so there is no need to step back and change keys.

Although this search engine still requires users to have impressions about the tag names which they are looking for, it prevents the search from terminating due to spelling errors. As users can get suggestion from the returned results after each key press, they only need to have a general idea of the tag names which are named by themselves.

4.5.4 Image Search Strategy

The purpose of locating a tag is to search for an image which is described by this tag. Tags are named to describe images. Therefore, the image searching procedure is the process of describing the desired images.In nTag, two strategies are provided to aid the searching procedure: inclusive search and exclusive search.

The inclusive search helps to resolve the problem that a same characteristic of images can be described by different words(tags). This strategy is an iterative process which starts searching with one tag and if this doesn't get the desired images, one more tag is used to search and the results are accumulated for each

round until the set of results contain the target. In nTag, this strategy applies to the OR-search scheme, which asks users to input several tags at one time and return a set of images described by any or some of these tags.

The exclusive search is used in the situation where the characteristics of images need more than one word(tag) to describe. This strategy is also an iterative process which starts searching with one tag and if this gets back too many results, one more tag is used to search for images among the results from the last round of search. This continues until the collection of returned images contain only the target. In nTag, this strategy applies to the AND-search scheme, which asks users to input several tags at one time and return a set of images described by all of these tags.

Chapter 5

Implementation of nTag System

The task of this chapter is implementing nTag. Based on the design of the previous chapter, the idea of this novel mobile image management system turns into a concrete embedded product on Nokia S60 3rd. Edition platform, for its experimental release. The whole implementation process is executed under the surveillance of previously prescribed requirements, and especially many efforts are devoted into the interface realization with the consideration of providing an easy and comfortable using experience.

5.1 Platform

The motivation of creating a nTag-like application is generated under the background of smartphones are dramatically grabbing the market share of mobile devices [36], while smartphone vendors are advancing their product's specification to win higher competitiveness, see section 1.1. Behind the war of smartphone devices, however, there are also several OS products are competing for the smartphone market. When choosing a platform to implement nTag on, besides satisfying the requirements prescription, it is also important to select a well accepted and promising product, in order to guarantee nTag's practicability and competitiveness. The main smartphone OSs include: Symbian OS, Linux, Windows Mobile and iPhone OS. The following parts will give a short introduction for each of them and come out a winner after an analysis for nTag system to base on.

5.1.1 Symbian OS and S60

Symbian OS is an operating system designed for advanced 2.5G and 3G mobile phones [35]. There are different mobile device platforms built on Symbian OS and Nokia S60 is one of them.

Nokia S60 is a user interface framework and it runs all S60 devices. It is claimed as the most popular software for smartphones and is licensed by world leading mobile phone manufacturers, including Nokia, Samsung, Lenovo and LG Electronics [9].

5.1.2 Linux and LiMo

Linux is an free, open source and widely used computer operating system [19], and beloved for its low cost and open for modification, it has been transplanted to mobile devices as so called embedded Linux.

LiMo a Linux based mobile platform operating system promoted by the LiMo Foundation, which is an alliance founded by Motorola, NEC, NTT DoCoMo, Orange, Panasonic, Samsung Electronics, and Vodafone in January 2007. Until Feb. 2008, there are 18 handsets from 7 vendors using LiMo as the OS [18].

5.1.3 Windows Mobile

Windows Mobile is Microsoft's mobile device platform, with operating system and other basic applications based on Win32 APIs. It follows the style of desktop versions of Windows and support for third-party software. The current version of Windows Mobile for Smartphone use is "Windows Mobile 6" [4] [8].

5.1.4 iPone OS

iPhone OS is an operating system developed by Apple Inc. for its smartphone product – iPhone. This operating system is based on the Mac OS X, which is designed for the computers of the same company.

Apple has released a free beta version of iPhone 2.0 software package, which includes iPhone SDK for third party developers to build native applications for iPhone; However, the iPhone simulator can only be run on computers of Apple Inc. [21].

5.1.5 Compare and analyse

Symbian is not open source software, but the APIs are public and developers can self-sign to be granted the basic developing capabilities. More advanced capabilities require certification and sign via the *Symbian Signed program* to ask for phone manufacturer's approval [6]. It is a potential problem for third party developers, but so far, nTag just requires the basic capabilities, such as file writing, etc. The big advantage of Symbian lies in the strong industry backup. Many statistics in reports and articles [13][1] shows that Symbian group stably seizes the majority of market. This suggests lots of efforts and capital are put to sustain and improve this product, which guarantee its vitality now and in the near future.

Linux OS is open source and free, so it is good at providing chances for users to customize their devices by themselves, but such customization requires profound learnings and skills on the system usage. However, what nTag looks for are simplicity and efficiency, because mobile image explosion may happen on any smartphone user, regardless their mobile OS knowledge. Moreover, until LiMo Foundation emerges, Linux doesn't has a strong and organized supporting group by mobile phone manufactures like Sybmian do. And thus it hasn't established a mature platform with large and stable user groups comparing to Sybiam with

Nokia S60.

Windows Mobile and iPhone OS also bring a lot of threaten to Symbian's leader position in market, especially in North American [1]. And for the powerful backup from Apple in image processing and entertainment supporting, iPhone OS could be a good candidate for our nTag. However, both of Microsoft and Apple are not complete open for third party developers and iPhone is still not released in most of countries outside U.S.A.

Conclude from the above analysis, since Symbian has been accepted by most of smartphone vendors and users and it is also available for nTag development, Nokia S60, the most popular Symbian based platform is chosen to implement nTag on.

5.2 Language

When OS and platform is chosen, to choose which development language is considered in this part. It be can picked up among three popular languages which are widely used by third party developers to develop software on the S60 platform. These languages are: Symbian OS C++, Mobile Java and Python for S60.

5.2.1 Symbian OS C++

Symbian OS C++, or put it short, Symbian C++, is specifically implemented C++ as the native language of Symbian OS. Nokia S60 platform provides SDK for Symbian device application developers. As being the native language of Symbian, it is the most powerful tool for developing. However, it is the most difficult language among all symbian develop languages.

5.2.2 Mobile Java

Mobile Java is Java language usage in the environment provided by Java ME (short for Java Platform, Micro Edition) on top of mobile devices. Java ME includes flexible user interfaces, robust security, built-in network protocols, and support for networked and offline applications. Its portability can enable application to function across different platforms. [22]

5.2.3 Python for S60

Like C++ and Java, Python is also an object-oriented language. Moreover, it is open source. It offers strong support for integration with other languages and tools; Nonetheless, Python is very easy to learn. Python for S60, or shortly, PyS60 is based on Python version 2.2.2. It supports many modules from Python Standard Library and also specific modules for mobile platform. PyS60 provides an easy way of rapid application developing and prototyping.

5.2.4 Compare and analyse

Symbian C++ and Mobile Java are both powerful language for symbian targeted software development, but they are much more difficult to use than PyS60. Especially the Symbian C++, as symbian requires special techniques such as descriptors and the cleanup stack, it is not economic for developing nTag which doesn't use complicated operations and techniques. And due to the strict security scheme of Java ME, a charged certificate is required for unrestricted accessing for components which are vital to nTag, such as file system, camera, etc.

On the other side, Python for S60 is easy to use, compatible to realize nTag's functions and free of charge. Therefore, in the consideration of simplicity, usability and economy, PyS60 wins for the development language of nTag.

5.3 Architecture

Chapter 4, the design shows that nTag consists of 4 functional parts, which are tag repository, image repository, tag-image link library and tag search engine. In the implementation level, an architecture follows the *Model-View-Controller*(MVC) pattern is used to realize the four functional modules. nTag separates database and file operations(*Model*) from GUI(*View*) by events handling(*Controller*), according to the MVC pattern which is simply depicted in diagram 5.1.

The *Controller* part is further realized following the event-driven programming application architecture. This architecture sets the application as a main loop which divides into event detection parts and event handling parts. In nTag system, the event detection parts are the parts who monitor and detect GUI

events; the event handling parts are the parts who choose responding operations towards the events.

5.4 Model

The job of this module is to deal with database and files operations. These operations include tag and image organization tasks, their management via database transactions, as well as operations on physical image files. This part of program can be divided into two sections:

- Several classes for database entries operations
- A class for database transactions and querying, as well as file operations

The database is using Symbian's built-in relational database engine, which is accessed via the interface in module e32db provided by PyS60. This module supports database transactions and querying with a subset of Simple Query Language(SQL). The class for database operations contains 3 local database tables to save information of tags, images and links between them. The entry classes define and operate the fields in each database table.

5.4.1 Entry classes

The *Model* module creates a database, which stores information of tag repository, image repository and taq-image link library into three tables. A database

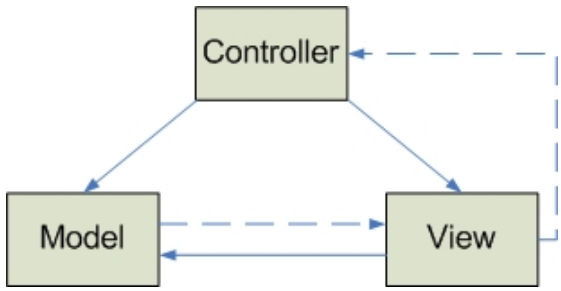

Figure 5.1: A simple diagram of MVC pattern. Note: the solid lines indicate a direct association, and the dashed lines indicate an indirect association

class creates these tables and takes on jobs of database transactions and querying. The fields of tables, however, are defined in each entry class respectively. Every entry class has to specify the table creation SQL commands and deliver these SQL commands to the database class for processing. Table 5.1 shows the entry classes and the tables they take care of, and detailed descriptions about the contents of this table follow it.

Table	Class
keytags	KeytagsEntry
tagimgs	TagimgsEntry
images	ImagesEntry

Table 5.1: Database tables and relevant entry classes

5.4.1.1 Table keytags and class KeytagsEntry

As shown in Table 5.2, the table keytags contain three fields: TagNum, KeySequence and Tag. In each entry of table keytags, Tag records the name of a tag, which can be any category of tags(see section 4.2). KeySequence contains a series of number which represent the numeric keys who generate the tag's name. The order of this series of number must conform to the key pressing sequence of numeric keys, i.e. "24462" is the only KeySequence of tag "China", but "24462" can also be KeySequence of tag "BigMc". Therefore, in this table, KeySequence can be duplicated, but Tag cannot. TagNum is a reference number for the tag to index.

type	COUNTER	VARCHAR	VARCHAR
field	TagNum	KeySequence	Tag

Table 5.2: Table keytags

Besides defining the table keytags, class KeytagsEntry also encapsulates field operations, all of which are listed in Table 5.3. These functions are also handles for the database class. Among these basic functions,*get_form* and *set_from_form* need some explanation. They are used for editing entries in the GUI, where PyS60 adopts a "Form type" to implement a dynamically configurable, editable multi-field dialog. Function *get_form* fill the GUI dialog with the contents of one keytags entry, while function *set_from_form* updates an keytags entry with the contents in the *form* parameter.

class KeytagsEntry
get_tag_number()
get_key_sequence()
get_tag()
set_tag_keysequence(skeysequence)
set_tag(tag)
get_form()
set_from_form(form)

Table 5.3: Functions in class KeytagsEntry

5.4.1.2 Table images and class ImagesEntry

Shown in Table 5.4, the table images contains two fields: ImgNum andImgName. ImgName contains the image name and ImgNum is the reference number for this image. In the experimental release of nTag, all images are required to be stored in one folder. Therefore, to lighten the database burden, ImgName only contains image file name without paths and suffix(it also assumes that an image's suffix is either jpg or JPG). No duplicate names are allowed in this table. All functions contained in class ImagesEntry are listed in Table 5.5, which are quite similar to the ones in other entry classes.

type	COUNTER	VARCHAR
field	ImgNum	ImgName

Table 5.4: Table images

class ImagesEntry
get_img_number()
get_img_name()
set_img_number(img_num)
set_img_name(img_name)
get_form()
set_from_form(form)

Table 5.5: Functions in class ImagesEntry

5.4.1.3 Table tagimgs and class TagimgsEntry

In Table 5.6, the table tagimgs contains two fields: ImgNum and Tag. This table records the mappings between tags and images, therefore Tag field contains tags in table keytags while ImgNum contains the image reference number in table

images. Since one tag can be referenced by several images as well as an image can be attached by many tags, each field in table tagimgs can be duplicated. Functions contained in class TagimgsEntry are listed in Table 5.7.

type	COUNTER	VARCHAR
field	ImgNum	Tag

Table 5.6: Table tagimgs

class TagimgsEntry
get_img_number()
get_tag()
set_img_number(img_num)
set_tag(tag)
get_form()
set_from_form(form)

Table 5.7: Functions in class TagimgsEntry

5.4.2 Database class

The database class named DatabaseClass, not only creates the three tables defined in entry classes, but also carries all operations on these tables. The relation between tables is established as Figure 5.2 shows. This diagram also reflects the relation between function modules of tag repository, image repository and tag-image link library, section 4.1. Seen from this database relation diagram, both columns Tag and ImgNum in table tagimgs are foreign keys. Therefore, if the records of these keys are deleted in tables keytags and images, the entries of table tagimgs which contains these records should be deleted as well.

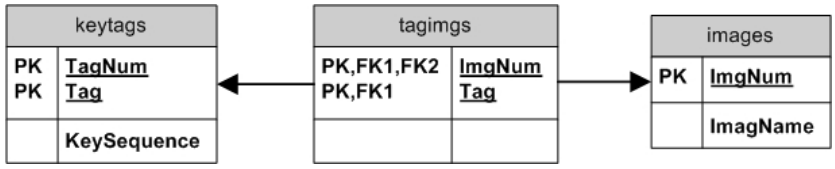

Figure 5.2: nTag database diagram: relations between three tables

DatabaseClass has to provide interface for *Controller* module to call for operations on tables. *Controller* uses the functions defined in this class to realize the controlling actions. Table 5.8, 5.9, 5.10 show the functions which are applied on database tables to implement the operations of function modules(tag repository, image repository and tag-image link library) respectively.

Table keytags
get_keytags_entries(keysq="")
get_single_keytags_entry(tag)
tag_exist(tag)
get_tags_for_keysequence(keysq="")
add_tag(e)
delete_tag(tag)

Table 5.8: Functions applied on table keytags to implement operations in tag repository

Table tagimgs
get_tagimgs_entries()
add_tagimgs_entry(e)
delete_tagimgs_entry(tag=None, img=MAX)
get_tags_for_image(img_num)
get_images_for_tag(tag)

Table 5.9: Functions applied on table tagimgs to implement operations in tagimgs link library

Table images
get_images_entries()
get_all_images()
image_exist(img_name)
add_image(img_name)
delete_image(img_num)
get_id_for_name(img_name)
get_name_for_id(img_num)

Table 5.10: Functions applied on table images to implement operations in image repository

Besides the database operations, there are some functions for file management and string operations, shown in Table 5.11. When the system initiates the database and creates tables, it uses function *get_photos_from_album* to loads all image names and calls function *handle_add_photos* to write these names into table images. After that, every time the system boots, it recheck the image folder and updates table images in function *reload_tagimages_database*. This function adds new image names into table, remove names of images which no longer exist in folder, and also remove corresponding entries if such images have mappings with tags. The importance of this function is that it guarantees the implementation satisfy the domain fitting requirements, section 3.4.2.4. *strip_path_and_end* and *wrap_path_and_end* are two The string functions, which remove or wrap the image file path and suffix.

file and string operations
get_photos_from_album(path)
handle_add_photos(photo_album)
reload_tagimages_database(photo_album)
strip_path_and_end(img_path)
wrap_path_and_end(img_name)

Table 5.11: Functions working on image files and image name strings

5.5 View

View module is the GUIs of nTag, which consists of 4 modes:

- Tag Text View
- Image Grid View
- Image Large View
- Camera Mode

Multiple modes are provided to realize different interfaces when system is running in different function modules. GUIs have configure interface mode in the callback functions of menu items in that *Controller* module detects all actions on keypad but is not responsible for differing reactions between various working environments. The following parts present the interfaces in each mode.

5.5.1 Tag Text View

This mode is defined as the main view of nTag and is also the first interface showing up when the system boots. It is a *listbox* style GUI and is used for tag browsing, tag searching and tag operating, see Figure 5.3.

Figure 5.3: Screenshot of nTag, GUI in Tag Text View mode

This interface is drawn to mimic the style of Nokia's contact book, at first all tags are shown in alphabetic order. Move `ArrowUpKey` and `ArrowDownKey` to change the focus for browsing tags. The scroll bar at the right of the screen shows the focus position among all tags. The query field at the bottom displays the query string from user input. When a numeric key is pressed, search engine(in *Controller* module) will execute the querying on database and then GUI replaces the original tag list with the querying results – a new list of tags. All displays during the process of tag searching conform to the sequence of nTag's searching method explained in section 4.5.2.

In query field, only the beginning substring of the first returned tag name is shown. In Figure 5.4 the first screenshort, "am" is generated by sequence "26" but there are 4 candidates, because "26" can also generate "an", "b6" and "cn". Nonetheless, only the substring of the first tag name "american" is shown in query field, i.e., "am".

GUI screenshots in Tag Text View can be seen in user manual, Appendix A to F.

The graphic display is implemented by redrawing the canvas. Callback function

Figure 5.4: Screenshot of nTag, tag query in tag search engine

_redrawCallback is a handler of `canvas` events and it contains several functions to draw specific parts of the display, such as focus bar, scroll bar, tag items and query contents. These functions are listed in Table 5.12.

Graphic display of Tag Text View
_redrawCallback(aRect=None)
_calculateFocus( )
_drawItems( )
_drawScrollbar( )
_drawQuery( )

Table 5.12: Functions to display graphic view in Tag Text View mode

Menus are hided at the position of `Options` – the left corner of the screen. They are triggered by pressing the `LeftSoftKey` and they are used in a way the same as normal Nokia's application menus. GUI mode changes when some menu items are selected so that the system runs into other function environments. In Tag Text View mode, menu items, their functions and the modes to be changed to, are shown in Table 5.13. The corresponding graphic view can reference to C.1 in Appendix C, and there are lot of other menu examples within the user manual, specifically from Appendix C to F.

5.5.2 Image Grid View

Under this mode, images are compressed to thumbnail size and are displayed in 3X3 grid view. Users control cursor to focus images by direction keys. Pressing `SelectKey` can let users view the image in large scale and the GUI mode will

Menu in Tag Text View mode		
Menu item	Function	Next mode
Add	add a new tag	N/A
Delete	delete the currently focused tag	N/A
Set default	set as the default tag	N/A
View tagged images	browse all images referencing to the currently focused tag	Image Grid View
View all images	browse all images	Image Grid View
Take a photo	use camera to take a new photo	Camera Mode

Table 5.13: Functions to display graphic view in Tag Text View mode

switch to Image Large View. In Image Grid View, action on numeric keys will not be responded because function *_eventCallback* decides that these keys are only responsive in Tag Text View. In this mode, images can be attached by or removed from tags via choosing these functions from the menu. Menu items, their functions and the modes to be changed to, are shown in Table 5.14.

Through the menu item Continue to search, the GUI will switch to Tag Text View mode and users are allowed to search another tag. It is the implementation of image searching strategy: exclusive search, or, AND-search scheme, explained in section 4.5.4. The selected tag is examined within the image results from the previous round of searching. If any image contains a link with this tag, it is chosen to the new result set.

Menu in Image Grid View mode		
Menu item	Function	Next mode
View image	view image in large scale	Image Large View
Add tags : input tag name : select from tag list	attach tags to current image by inputting one tag name by selecting tags from list	N/A
Remove tags : input tag name : select from tag list	remove tags from current image by inputting one tag name by selecting tags from list	N/A
Continue to search	AND-search	Tag Text View
Back to main	go back to the main interface	Tag Text View

Table 5.14: Functions to display graphic view in Image Grid View mode

Compressing and displaying images is a time consuming task for mobile devices, for they don't have a specific graphics card. Long response time is a practical problem in displaying images in grid. The platform compresses images into thumbnails and display them on screen one by one. When images are more than

3X3, the rest images are displayed in next pages. Due to the delay, erroneously displaying will happen if a page changes to the next one but some images in current page haven't come out yet. These old images will cover the new images in the new page. To solve this problem, a canvas is used as a buffer. All 9 thumbnails for one page are first drawn on this buffer canvas, and then this buffer canvas is displayed on the screen. In this way, 9 thumbnails show up at the same time.

The complete layout and interaction procedure screenshots under Image Grid View mode are available user manual, especially in Appendix C.

5.5.3 Image Large View

GUI turns into this mode when any image is viewed in large scale. Besides the forbidden using of numeric keys, this mode further shields the response of direction keys and leaves users the only interaction means – the menu. This menu is almost identical to the one in Image Grid View, expect providing a choice to go back to the grid view. A series of screenshots are shown in Figure 5.5 to give a general picture of this mode.

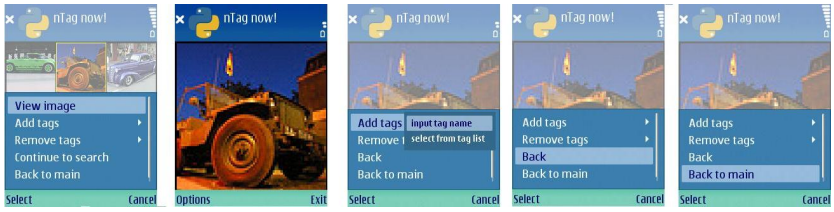

Figure 5.5: Screenshots: GUI under Image Large View mode

In Image Large View and Image Grid View, users can choose menu items Add tags or Remove tags to organize images by adding or removing tags. For a better human concerned interaction, each menu item provides two means: directly input tag names and choose tags from tag list. To add or delete tags by inputting tag names, only one tag can be executed at a time. If a tag to be added to the current image doesn't exist in tag repository, it will be created in the database table keytags, and its link with this image is registered in table tagimgs. After all these operations, a message is shown on screen to inform users the execution is successful. If the tag to be removed doesn't have a link with the current image, the operation will be cancelled and a message is shown to inform users this situation. However, such situation won't happen if users choose to remove tags by select from tag list, because this list only contains the tags which have been mapped to the current image. This function permits multiple selections and so

does the one to add tags,so several tags can be operated at a time. Function menu under Image Large View mode is shown in Table 5.15.

Menu in Image Large View mode		
Menu item	Function	Next mode
Add tags : input tag name : select from tag list	attach tags to current image by inputting one tag name by selecting tags from list	N/A
Remove tags : input tag name : select from tag list	remove tags from current image by inputting one tag name by selecting tags from list	N/A
Back	view image in grid scale	Image Grid View
Back to main	go back to the main interface	Tag Text View

Table 5.15: Functions to display graphic view in Image Large View mode, note: N/A means not changing mode

5.5.4 Camera Mode

This mode is set for taking photos under nTag environment. It is mentioned in requirements description section 3.4.2.1, that camera activities out of nTag system will not be monitored. So, if users want to use nTag to feel the "`None Time Gap`" organization for newly taken pictures, they can use the camera under this Camera Mode in nTag. `*Key` and `Select Key` are activated for controlling the camera shutter, the `♯ Key` is used to stop the view finder and leave this GUI mode. When a new picture is taken, a default name is given and system tags are attached to this image automatically. Moreover, user are suggested with a tag(which can be customized by users before taking pictures) and they can choose accept or give other tags to this image. After all of these immediate organizations, GUI switches to Image Grid View mode, where more organization and other operations can be done. The GUI screenshots of the whole procedure of taking a new photo and immediate image organization can be found in Appendix F.

5.6 Controller

PyS60 supports event driven programming, so nTag doesn't have to implement a self-designed controller to detect the events and trigger the callback functions. PyS60 do it implicitly. This thesis regards events handling in callback functions

as a part of interface implementation which has been explained in *View* module section. So this section elaborates how *Controller* module implements keypad operations and tag search engine.

5.6.1 Class Keyboard and _eventCallback

Class Keyboard is a handler for actions on mobile phone's keypad. It detects the events, and then it judges whether a key and which key is being pressed down or being released or has been held down for a while. When this handler gets the result, it triggers the *_eventCallback* which is a callback function referenced by class Keyboard in the form of parameter *aOnevent*. The *_eventCallback* is implemented in class nTagClass, where GUI operations and other controlling are carried out on the results from the keyboard handler.

Apart from judging the keypad status, class Keyboard also translates strings into key sequences. When a new tag is being input, *Controller* has to "encode" the tag name into a series of number. And then *Controller* forwards this key sequence together with other fields to be processed in database inside *Model* module. All functions in Class Keyboard can be seen in Table 5.16.

Class Keyboard
handle_event(aEvent)
is_down(aScancode)
pressed(aScancode)
get_keysequence(in_string)

Table 5.16: Functions in class Keyboard

5.6.2 Search engine

Search engine is implemented in class nTagClass and it contains two parts: one part is the function *_searchTagLib*, and the other part is the callback function *_eventCallback*.

Search engine detects every query letter and divides them into three categories: Right Input, Wrong Input and Back Space. The work flow of search engine is depicted in Figure 5.6. Different actions are taken with regard to different queries:

- When an input query falls into Right Input, function *_searchTagLib* at-

taches this key number to the previous key sequence. Here *Controller* uses this new key sequence to ask for tags through function *get_keytags_entries*.

- If users press `Backspacekey`, *_eventCallback* removes the last key number in the current key sequence and labels this query letter as a Back Space. In this case, *_searchTagLib* calls for *get_keytags_entries* using the shortened key sequence as the parameter.

- Wrong Input is used to prevent further erroneous querying after a query has already resulted in empty answers. Once a searching returns None, *_searchTagLib* marks it as Wrong Input. If the following pressed key is not the `BackspaceKey`, *_eventCallback* won't save the new key number and *_searchTagLib* won't call for database operation either. Until users press `BackspaceKey`, key sequence resumes back to the correct query, *_eventCallback* resets the status back to Right Input.

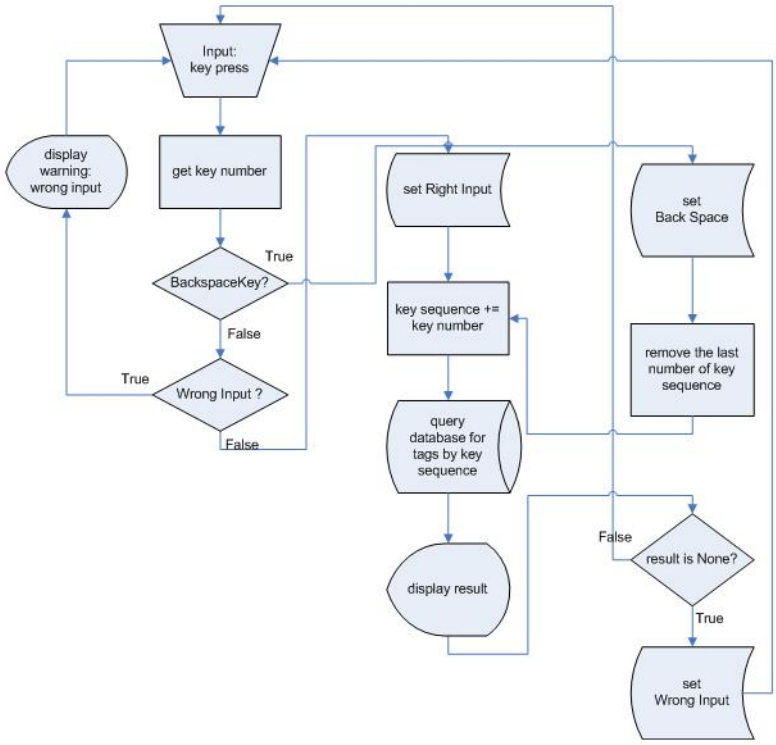

Figure 5.6: Diagram of search engine's work flow

5.7 Limitation

nTag of current version is a experimental release, the implementation has following limitations regarding to the requirements prescription and design:

- nTag has not been made into a .sis file, it has to boot from python environment, see Appendix B
- Image management among multiple folders is not supported, images have to be stored in a certain folder
- Search engine does not support tag searching out of GUI mode Tag Text View, i.e. currently PyS60's appuifw.multi_selection_list is used to implement the function of multiple tag selection for adding or removing tags under the menu item select from tag list
- OR search has not been implemented

Besides these, all design mentioned in Chapter 4 and shown in Appendix A to F are implemented.

Chapter 6

Evaluation

In this chapter, quantitative and qualitative data are collected and analyzed to give an overall evaluation on the design and implementation of nTag. There are three questions are expected to be answered after this evaluation:

- Whether the implementation of nTag has fulfilled the requirements?
- How well nTag solves practical problems of mobile image management?
- What pros and cons does nTag have, comparing with other solutions?

6.1 Overview

The evaluation job is carried out in three phases to answer the three questions respectively. First, a series of software testings are applied to the nTag program to determine whether the product has realized the concept of the design. Second, a usability experiment is performed by 4 volunteers to investigate the user acceptance and the practicability of nTag, when it is used in solving real problems of image management on smartphones. Last but not the least, an functional analysis is given, which is based on the comparisons between nTag, mobTAG and Nokia N73's built-in image management systems.

6.2 Software Testing

Apply the case of nTag to software engineering, audits and tests are carried out in different phases of this project, which include requirement review, unit test and regression test, integration and system test, performance and acceptance test.

6.2.1 Requirement review

Requirement review is the first audit immediately after the design of nTag is sketched out. Here the requirements include both the functional requirements and non-functional requirements. Functional requirements describe what a system must do while non-functional requirements act to constrain the solution, such as availability, testability, maintainability, and ease-of-use. The review checks whether every requirement can be answered by a yes to each of the questions in a set of requirement testing items promoted by Suzanne Robertson in her article [32].

6.2.2 Unit test and regression test

Then unit tests are performed and repeated to check the functions until they fulfill requirements. nTag is first developed on PC S60 simulator, which is provided by Nokia in the package of S60 developer tools. Unit tests and regression tests are also done on this simulator, except the Camera mode function which is tested on Nokia N73 because simulator doesn't support camera.

These unit tests use both black-box-test and white-box-test. Black-box-test is used on function level, that each function is tested to guarantee that it outputs the desired results and performs only designed activities. This is not enough, however, S60 simulator doesn't contain a powerful IDE and it's complier is too simple to show ample error information on the limited screen, so white-box-test is used as a complement to help further locate errors.

Besides unit test, in the testing procedure of nTag, regression test is a often performed by means of repeating unchanged and previously passed unit tests on the modified software to prevent unintended consequence on those fine functionality.

Moreover, If bugs are found in later test phases, testing flow loops back to unit tests and regression tests level, until the software satisfies the requirements on the smartphone.

6.2.3 Integration and system test

After unit test, these units are grouped to be a larger aggregates to be tested on their function and performance as a whole. nTag is not a huge project, integration is not defined clearly between unit test and system test. But according to the different modes of GUI, each mode is under test separately, which can be regarded as an integrated test. It uses black-box-test and is carried on PC simulator.

When the program is moved from simulator to a real smartphone Nokia N73, system test is applied. This test also falls into the scope of black-box-test, where both software and hardware issues are considered in the test. Apart from testing the functional aspect, whether it fulfills the performance requirement specification is also among the testing contents. nTag's system test is performed on Nokia N73 smartphone by thesis author and there are a lot of functional testings are executed during the process of nTag manual's writing, see *A Useless User Manual*, from Appendix A to F.

6.2.4 Performance and acceptance test

The performance and acceptance test is executed by users. In this test phase, not only quantitative methods but also qualitative methods are applied. In this project, this test is emerged with a carefully designed usability experiment, which is elaborated in the following section.

6.3 Usability Experiment

When nTag has passed the system test, the examination on developer side is over and the further examination should be turned to the potential users. In order to check the user acceptance, to investigate the product's usability and performance in dealing with practical problems, 4 volunteers are invited to perform this experiment and show their using experience.

The experiment is divided into two rounds and are performed by 4 volunteer testers. The volunteers are chosen from DTU students. The volunteer selection is not absolute randomly, since their majors are considered[1] to avoid the situation that volunteers have much academic knowledge closely related to this product, in which case the experiment loses the intent on examining general users. About the number of volunteers(or call them testers) and tests, according to the study in [29], 4 testers can discover 75% of the usability problems. And this study also highly suggests dividing the whole test into small groups of tests, so every tester can do more tests from different aspects and thus contribute more suggestion on the improvement of the product.

6.3.1 Experiment Design

As stated, each tester has to carry out two rounds of tests. Testers are requested to use nTag to organize a set of test images for each round. One difference between the two tests is the image set. The other difference is the testing time, e.g. the second test is carried after the first test has been finished. Each test has two level. In the first level, testers organize images. In the second level, testers are examined for their organization. More details regarding to the test design are shown in the following subsections.

6.3.1.1 Experiment devices

- Nokia N73: 2 Nokia N73 basic version smartphones are used as the platforms to test nTag. Both of them are updated to the latest version of OS and all third party applications are removed. The short term "N73" is used in the following parts to denote these two Nokia N73 smartphones.
- Memory: Panasonic[2] 64MB mini SD card is used as the working storage for each of N73. Free space accounts 100% of the capacity for each card.

[1] from biotechnology, food, environment and mathematics majors
[2] http://www.panasonic.com/

- SecurID: A secured digital number generator from RSA[3], is used here to generate random number of test images.
- PC: A Lenovo [4]T61 laptop is used to show images to testers in the examines
- Timer: An electric sport watch, which can time two events simultaneously.

6.3.1.2 Experiment images and tags

- Images: Images are collected from Internet, the size of each one is around 8k. Images are divided into 3 sets: A, B and C (see Figure 6.1). A set and B set each contains 18 images, while C contains 150 images. No duplicate images among different sets and no duplicate images inside A and B.
  - A is used for the first round test and its images chosen from confusing ones, e.g. one image is the text "Denmark", another is a flag of Denmark, and there is also a flag of China. The purpose of such design is to force testers to use complicate expressions if they want to differentiate them.
  - B is used for the second test and its contents can be classified into five categories including architectures, cars, foods, animals and plants. The idea is to examine testers' classification habit.
- C is used for both tests in second phase test and images in C are all duplicate images. C is used for complicating the job of image locating by direct browsing all images in the collection. Another use of these images is to challenge the database!
- Tags: A list of tags are chosen from Flickr[5] and Amazon[6] most popular tags. The number of these tags is around 200. These tags are used as default tags provided for testers, and they are also used to complicating the job of locating a tag in the list. Again, another use of these tags is to increase database burden.

6.3.1.3 Experiment strategy

The following procedures are carried out in sequence:

[3]http://www.rsa.com/
[4]http://www.lenovo.com/
[5]http://www.flckr.com/
[6]http://www.amazon.com/

1. Install nTag to N73 in the storage of memory card and set the default tag list.

2. Set images from A to N73 for the first test; set images from B to N73 for the second test.

3. Testers to requested to use nTag to tag these images. It is the first phase of the test. They are informed that they will be asked to locate these images by their tags using nTag, which is the second phase of the test due to 3 days later.

4. No manual is given but only an instruction on how to launch nTag. During the test, if testers cannot proceed anymore without the manual, testers can ask for the manual, but a penalty is given by losing the thanks-gift* prepared for testers.

5. Testers are requested to use N73 built-in image management system to organize these images, i.e. use its "album" instead of nTag's tag. The number and names of albums should be the same as the number and names of tags for every image.

6. Every tester has to finish the first phase test within one day and give back N73.

7. Add images in C to N73; make the images from A or B evenly distributed among images from C; set the default album list to N73 built-in image management system, which should be the same as the default tag list set in strategy action 1. It also has the same use as the default tag list.

8. The second phase test is performed 3 days after the first. Testers are returned their test devices with images which have already been organized by themselves plus images in C. 6 images are randomly chosen(3 out of each 9) from 18 images(not include images from C). When one of the chosen images is shown on the PC screen, testing is executed in the following ways:

Figure 6.1: Image sets: the left cluster is A, the right cluster is B

(a) testers use **nTag** to locate and show this image on N73. Instead of browsing in the image set, testers can only use the image searching function provided by nTag. Time consumed for each locating is recorded. If testers fail to locate an image within **2** minutes, this locating is recorded as a fail and not accounting time.

(b) testers still use **nTag** to locate and show this image on N73. Instead of using the image searching function provided by nTag, testers can only browse images in the image set. Time consumed for each locating is recorded. If testers fail to locate an image within **2** minutes, this locating is recorded as a fail and not accounting time.

(c) testers use **N73 built-in system** to locate and show this image on N73. Instead of browsing in the image set, testers can only locate an image by using albums. Time consumed for each locating is recorded. If testers fail to locate an image within **3** minutes, this locating is recorded as a fail and not accounting time.

(d) testers still use **N73 built-in system** to locate and show this image on N73. Instead of using albums, testers can only browse images in the image set. Time consumed for each locating is recorded. If testers fail to locate an image within **3** minutes, this locating is recorded as a fail and not accounting time.

9. Remove all data and uninstal nTag from N73.

10. Begin test 2, repeat strategy from 1 to 9.

6.3.1.4 Experiment statistics

The answers for the following questions are recorded for the evaluation:

- How much time on average(Avg.T) to successfully locate an image for each round of test, by each method of strategy 8a, 8b, 8c and 8d?
- What is the maximum and the minimum time(Max.T &. Min.T) to successfully locate an image for each round of test, by each method of strategy 8a, 8b, 8c and 8d?
- How many times/persons fail(FTP) to find the image by each method of strategy 8a, 8b, 8c and 8d? Why?
- What is the response time on average(Avg.Rd) for displaying one image in nTag and N73 built-in system respectively?

- How much time on average used to add a tag to an image(Avg.Timg) using nTag and N73 built-in system respectively?
- What is the response time on average(Avg.Rt) for tag searching(every hit) in nTag?
- How many testers create new tags(NTp) for images instead of using the default tags?
- How many tags(AL.Tag) and unique tags(AU.Tag) are used in total by each tester?
- How many images assigned a same tag on average(AR.Tag) by each tester?
- How many tags on average(Avg.Tag) are used for one image?
- What is the maximum number of tags(Max.Tag) used for one image?
- How many time the nTag dump (Dump)during the whole testing procedure?
- How many testers ask for manual(Manu)?

6.3.2 Experiment result and evaluation

	nTag 'tag' strategy 8a	**N73 'album'** strategy 8c	**nTag browse** strategy 8b	**N73 browse** strategy 8d
Avg.T 1.(s)	13.0	16.0	36.0	70.4
Avg.T 2.(s)	12.1	15.2	32.2	49.1
Max.T 1.(s)	38.1	25.9	78.7	130.8
Max.T 2.(s)	35.3	30.5	58.9	70.2
Min.T 1.(s)	5.0	5.1	9.5	25.2
Min.T 2.(s)	4.9	5.5	11.1	9.2
FTP 1.(t/p)	2/2	2/2	0/0	0/0
FTP 2.(t/p)	2/2	2/2	0/0	0/0
Avg.Rd (s)	0.3	0.3	0.3	0.3
Avg.Timg(s)	24.1	30.7	N/A	N/A

Table 6.1: Result 1. statistics from 2 phases of tests with 4 strategies

	Avg.Rt(s)	**NTp**	**Max.Tag**	**Avg.tag**	**Manu**	**Dump**
nTag	<1	4	4	1.5	0	0

Table 6.2: Result2. Statistics from nTag using

	AL.Tag	AU.Tag	AR.Tag
Tester A	78	29	2.69
Tester B	36	35	1.03
Tester C	69	40	1.73
Tester D	36	23	1.57

Table 6.3: Result 3. statistics of every tester's tag habit

The test results of each tester's performance are computed and divided into three categories, shown in Table 6.1, 6.2 and 6.3, respectively. The meaning of each row in these tables are referenced to the questions in section 6.3.1.4. Analysis will be given on the statistics in each of these tables.

6.3.2.1 Performance evaluation

The whole usability test consists of 2 groups, 2 phases and using 4 strategies for each tester. The statistics relevant to image locating time of different strategies are shown in Table 6.1.

The statistics of average locating time(**Avg.T**) for each strategy shows that tag-based nTag scheme is the fastest solution. It is nearly 20% faster than N73's "album" strategy, which is also a tag-based solution. But both of them are tag-based, why nTag wins? This can be answered from nTag's design. In section 4.1 and 4.5.2, it is explained that nTag's tag search engine is introduced to deal with the problem of locating tag in a large tag list. In the usability test, both nTag and N73 built-in system are set a large default collection. Even under this circumstances, nTag just requires a few keypresses to locate a tag, but N73 has to browse among more than 100 albums for the desired one. So, in this deduction, N73's solution is slow because the conventional browsing method is slow, and the fact of the slowness of conventional browsing method is also demonstrated in this group of statistics – nTag browse and N73 browse strategies are slower than first two tag-based strategies.

Using nTag to browse is faster than using N73 system is because the image presentation style of nTag is better. nTag puts images in 3x3 grid and can proceed to the next 3x3 by 3 key presses; on the other hand, N73 normally allows to proceed one by one. Regarding to the difference between two image presentation styles, section 6.4.2 will explain in more detail. But in this test, another reason of conventional browsing strategies is lower than tag-based strategies, lies in the response time of displaying images, which is demonstrated by the statistics of Avg.Rd. This value doesn't differ much between nTag and N73 built-in system,

because they are both running on a same hardware and software platform. And also because this test uses very small size images(around 10kb), the algorithm difference on displaying images is hardly shown. Under this background, the number of images to be displayed is the crucial factor which decides the time difference between each strategy. Browsing strategies need to display every image before reaching the target, but tag-based strategies only display images which share the target tag. The statistics of **AR.Tag** in Table 6.3 shows that less than 3 images share a same tag on average, so tag-based strategies only display 3 images on average for the correct locating actions. That's why nTag's "tag" strategy is much faster than N73 browse!

Max.T and **Min.T** also reveal efficiency issues, but they are not as important as **Avg.T**. Because the location of an image among all images decide these two values of conventional browsing method strategies. And because N73's "album" strategy also contains conventional browsing feature, e.g. the target tag is "apple" and is situated among the top of the tag list, the locating time must be very short. But for nTag's "tag" on **Min.T** can only do better than N73, because nTag's "tag" strategy also supports browsing, and its tag searching just makes it better. That's why nTag doesn't lose on the statistics of **Min.T** comparing to N73. However, for the **Max.T**, nTag's "tag" solution costs more time than N73's "album". This should not simply be explained as a failure. Besides the factor out of design, It has something to do with the design of this test and will be explained in the following part 6.3.2.3 when discussing tagging habit.

6.3.2.2 nTag usability

Statistics of **Manu** and **Dump** in Table 6.2 show some important messages. Manu value is 0 means that no tester ask for the manual during the testing procedure; however, they all have successfully finished the test tasks. It indicates that nTag is very easy to use and the GUI is user friendly. Dump value is 0 means that the system is dependable in respective of the reliability, see requirements prescription at section 3.4.4.2.

Avg.Rt in Table 6.2 is not an accurate value, because using the current timing equipment and method cannot reach an more accurate but also correct value. The design of the test tries to show the average processing time on tag searching. However, the current value "<1" seconds is valuable and meaningful enough. It indicates the delay of tag searching won't cause diversion of users' attention, which satisfies the requirements on machine performance 3.4.4.1.

Another group of statistics, **Avg.Timg** in Table 6.1, also reflect nTag's usability

feature. The average time for an adding tag operation is 24.1 seconds, 6 seconds less than the same operation in N73 built-in system. One reason is that nTag allows users to search tags they want to add as well as directly input the tag name. N73, however, does not support tag(album) searching, and only permit inputting the name of a tag which doesn't exit in the tag list. Users can only scroll down the tag list no matter how long it is, if they want to use an existent tag. Therefore, the design of nTag shows more value in the usability aspect, simply because it provides convenience for tag reusing, which is a crucial feature of using tag to organize images. Tags are useless if they are not reused and every image has different tags!

6.3.2.3 tagging habit with nTag

This experiment is also designed to reveal, hopefully, some specific human tagging habit on mobile devices. The reason to set an interval period between two phases of test is trying to examine whether testers' taggings are consistent. The consistent tagging means testers will always use one or several tags to describe the same image. This feature has great influence on evaluating the practical value of the nTag design. Because looking for an image it is a practice of re-tagging, and target images can be found via correct tags only when re-tagging is identical to the original tagging. Since nTag only uses tag names to locate images, if users' taggings are inconsistent and they have already forgotten the tags of the images, they will lose the clue.

Seen from the statistics **FTP** in Table 6.2, nTag's "tag" strategy receives one time of failure from two testers each, in each round of test. This is the situation when users have forgotten the tags they used and re-tag inconsistently. It is also the reason why nTag's **Max.T** is higher than N73 tag-based strategy – testers used time to recall memory and re-tag. As mentioned in the earlier part, it is a flaw of this test's designing. Testers used time to recall memory and re-tag only for the nTag, which is the first strategy to get tested. After that, when testing N73's tag-based strategies, testers have got known the tag names for images and only executed the task to browsing. To improve the design of the test on this issue, using different images to test nTag and N73 might be a good choice.

The rest statistics such as **NTp** discover that testers are not bothered to use their own expression to describe images; however, both **Max.Tag** and **Avg.tag** show that testers do not use too many tags(less than 4 and 1.5 on average) to describe images. **AU.Tag** in Table 6.3 tells the unique tags have been used by each tester. There are 36 images in total for two round of tests, but Tester C uses 40 unique tags to describe these images. This may cause the attention of nTag designer to reconsider the requirements on tag capacity. But this test only

uses 36 images to test, it is possible that the number of unique tags will not increase as fast as the number of mobile images.

6.3.2.4 User feedback

Testers also have contributed some useful experiences and advises, both about nTag and image management on mobile devices:

- nTag is useful, interesting and especially easy to use
- when searching images by tag, the returned results should only contain mapped tags. This is useful when users are not sure their targets and want to check one by one from candidates.
- tag multiple selection supporting from nTag's searching method is greatly expected, since PyS60's multi-selection-list is not good to use and far behind nTag's search engine
- nTag should enlarge the input area
- before mobile devices can display images as fast as PC can, they won't use mobile to view a lot of images

6.4 Comparison

In the carefully designed using experiment, nTag has earned testers' acceptance for its practical design and good performance on solving real problems. In this section, nTag is compared to other application products which are also addressed to the problem of mobile image management. One is Nokia N73 built-in system, which the thesis has adopted in the usability experiment to compare with nTag in different strategies. The comparison in that experiment aims at the performance perspective. However, in this section, the aim is to examine the functionalities and during the comparison, every application is regarded as a concrete product. The other mobile image management system who joins this comparison is mobTAG, which is presented by Søren in his master thesis [24]. mobTAG uses a tag-based hierarchical structure to organize images and it adopts a key-mapping interaction scheme to facilitate the operation of the product.

6.4.1 Criteria

All of these three applications are using tag to help organize images. To compare the overall functionality on mobile image management, the following image management operations and management solutions are chosen as criteria:

- **Organization:** the structure to organize the images, see section 2.5.2.2 for explanation
- **Tag locating:** the way to locate a tag
- **Search Strategy:** the strategy to execute an image searching
- **Present Image:** the way to present images and the way to choose an image
- **Customization:** the solution to customize the image management
- **Time Gap:** the solution to organize a newly taken photo

The result of comparison is in Table 6.4

	Organization	**Tag locating**	**Search Strategy**
nTag	flat tags	search tag names	AND, OR
mobTAG	structural tags	traverse tag levels	AND, OR
N73	flat tags "album"	browse tag list	N/A
	Present Image	**Customization**	**Time Gap**
nTag	thumbnail in grid; focus	N/A	default / after / in advance
mobTAG	thumbnail in grid; key maps	organize tag levels	default / after / in advance
N73	slide show, thumbnail in sequence; focus	N/A	in advance

Table 6.4: Comparison of nTag, mobTAG and Nokia S60 3rd. built-in image management system in functionality perspective

6.4.2 Comparison and explanation

Nokia N73 uses "album" to organize images. But these albums are not real physical file folders, they are none but tags. These albums are organized in a

flat structure, the same with nTag which constructs tags in a flat structure in tag repository, section 4.2. The shortcoming of N73 lies in no tag searching function. So to locate an album, users have to scroll down the album list. It poor performance of this kind of conventional browsing has been discussed in section 6.3.2.1. nTag, however, is equipped with a powerful tag search engine and this helps a lot to locate a tag. mobTAG also uses tags to map images, but it organizes the tags in a tree structure, which is customized by users. Users reach the desired tags by going down the levels of the tag tree and they must have some clues to help them find out the correct location of the desired tag. mobTAG doesn't support tag searching. nTag and mobTAG both support image searching strategies, which are inclusive searching(OR-search) and exclusive searching(AND-search); while N73 doesn't.

N73 presents images in thumbnail size and they are displayed in queue sequence. N73 can show 11 image thumbnails in a screen. It also supports slide show mode, which provides an entertaining flavor but has nothing to do with image management. N73 utilizes left and right direction keys to choose change images and the step span is one; up and down direction keys are used to move at a larger step, but it uses a unknown logic to determine the step span. nTag and mobTAG display image thumbnail in 3x3 grid per page(1 page/screen). nTag uses direction keys to control the focus on thumbnails, while mobTAG maps 9 thumbnails to 9 numeric keys and chooses icons by pressing mapped keys. To view more thumbnails, users need to change pages. However, the current available mobTAG has not realized the change pages function.

The notion "Time Gap" is used in Table 6.4 to show how these applications shorten the time gap between image generation and image organization. When a new photo is taken, nTag and mobTAG immediately attach the system tags(the date) to this new image. It is a way to organize images by default. Then they remind users to add more tags for a better organization, but users can ignore this and just use the system tags. N73 doesn't ask users to organize photos after they have been taken; however, users can predefine an album to contain all new photos from then on. It is a way of organizing in advance, nTag and mobTAG realize this scheme by setting a default tag.

6.4.3 Summary

From the above functionality comparison, it can be seen nTag has the same organization solution with Nokia N73, but wins the latter with a tag searching engine and image searching strategies. mobTAG can compete with nTag on searching images. Moreover, mobTAG forces users to organize the structure of their tags. This feature, however, like a two edged sword. On the good side,

it provides a chance to those users who love personalization and they can build a favorite structure to help them locate tags and classify their images. On the other side, it is not only troublesome for users to find places for all tags, but also tricky for users to build a well-structured tag tree so that they won't get lost. Remembering the paths of all tags is mandatory, especially when mobTAG doesn't support tag searching by name. Nokia N73 enable users to organize images before they are generated by the smartphone's camera. mobTAG reminds users to organize photos immediately after taking photos and also provide choice of system tags to help organization by default. nTag provides all these solutions.

Chapter 7

Conclusion

This thesis has proposed an image management system – nTag, targeting to the smartphone users, who are supposed to confront the problem of managing a large collection of mobile images right now or in the near future. nTag runs on Nokia S60 platform, works with a GUI and is controlled via conventional mobile phone numeric keypads. It uses tags as tools to organize images, where multiple tags can be attached to a single image and different images can share a same tag or tags. Moreover, the system provides a search engine helps users quickly locate an image by names of related tags. The development of nTag is guaranteed by applying the software engineering methodology. Test has been executed during the whole process of design and implementation, and documents are maintained all the way, from domain description until user acceptance test plan and result.

nTag is evaluated by three methods. First, the application product undergoes a series of software testings and the results acknowledge that nTag's implementation has fulfilled the requirements prescription. Second, a carefully designed usability experiment is performed to investigate nTag's practical performance when solving real problems. In this experiment, both nTag and Nokia N73 built-in image management system are utilized to organize and locate images. The two use a same method to organize images, but nTag wins the game because it achieves the fastest image locating – 20% faster than N73's tagging strategy and more than 4 times faster than its conventional browsing strategy. nTag also bears other sparkle features, such as an easy operation, short response time and

good dependability. Finally, from a functional perspective, nTag is compared with mobTAG and Nokia N73's image management system. 6 mobile image management related issues are chosen as criteria to check how these applications deal with the them. nTag and mobTAG, each of them has some strong points to recommend. mobTAG forces users to customize the tag structure, which provides more free space for personalization but at the price of complicating the image organization procedure. nTag supports tag searching by name, which beats the mobTAG as well as Nokia N73. If there is no means to help swiftly locate a tag, the trouble of navigating in a large collection of images transforms to the problem of locating a tag in a long list of tags. Only nTag has solved this critical problem and its tag search engine is really prominent.

All in all, based on the evaluation results, it can be concluded that nTag is a practical design as well as a successful product. nTag can help users manage a large number of mobile images in an easy and fast way. Judged from this aspect, nTag has achieved the goal mentioned at the beginning of this thesis and it plays better or even much better than other solutions.

7.1 Contributions

The following achievements are considered as contributions of this thesis:

- nTag System - a well performed image management application product, targeting smartphone users to cope with mobile image explosion
- Searching Engine - a novel tag/name search engine on mobile devices, which can greatly reduce the time, the number and the complexity of inputting with the numeric keypads. This solution can also be used in other domains, e.g. in the contact list
- Usability Experiment - a usability experiment is carefully designed to investigate the usability and performance of mobile image management system, but also have revealed some issues regarding to people's habit on tagging as well as organizing images on mobile devices
- Comparison - a comparison is made between three mobile image management systems, from functional perspectives to examine their image management ability
- Methodology - providing a real example of practicing Dines Bjørner's triptych theory of software engineering to analyze, model and implement a software product. It is a meaningful try to study the problem of mobile image management from the methodology perspective.

7.2 Future work

Due to the limitation on time and human resources of this study, some features in the design of nTag have not been implemented. Moreover, the domain is limited to an afford scope, but this restrains the study from other interesting issues and promising technologies, which are in a bigger domain. Therefore, the following tasks are left for the future efforts:

- Accomplish the implementation: nTag should implement the tag selection list as a multiple selection list; replace the current PyS60 provided mult-selection list to one supported by nTag's own search engine; in image searching, returned suggestive tags should be only the image-mapped; provide the structured default tags.
- Improve the performance: Test could be executed in more thorough ways and by more testers; Change the implementation language from PyS60 to Symbian C++, which is expected to be more efficient.
- Larger application domain: Since N73 supports direct uploading images to Flickr, the Internet share of images on personal mobile devices is an interesting topic and promising application, which should be continued after the study of this thesis; furthermore, touch screen usage on mobile devices is also a promising technology, which can free the hard keypads completely, and thus, may trigger an revolution on input methods. The case is the same with camera-nTag activation and monitoring, which are useful functions and are worth further attention. Last but not least, using hierarchical default tags as a compliment to the ordinary flat structural user defined tags can also result in another thesis. All these domain extension issues were mentioned in Section 3.4.2.1, where they were excluded of out the domain of this thesis and remained for future investigation.

Appendix A

A Useless User Manual for nTag

This useless manual shows what you can get known within minutes of a hand on experience with nTag!

This manual includes Appendix B, C, D, E and F.

Figure A.1: N73 with nTag!

Appendix B

A Useless User Manual: Boot nTag

This part shows how to boot nTag using its script in python. It is considered to be the most useful part of this useless user manual!

Figure B.1: Start from N73 desktop, and follow menu to application folders until reaching python.

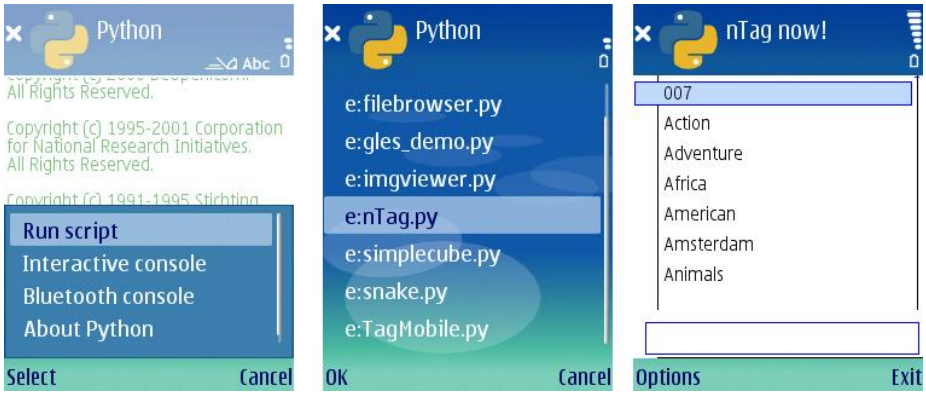

Figure B.2: Browse the script list and find out nTag, run it. Here you are, do as it says, nTag now!

Appendix C

A Useless User Manual: Start to Tag Images

Not it begins to tag. This part shows you how to read menus.

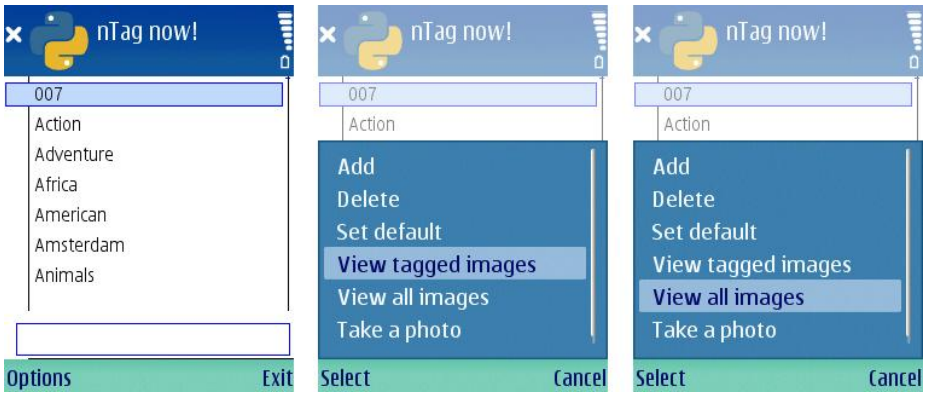

Figure C.1: Starting from the main interface – a tag list, there are two ways to get access to images. If choosing view tagged images, all images will be shown once no images mapping to this tag.

Figure C.2: In Image Grid View, using frame cursor to change focus

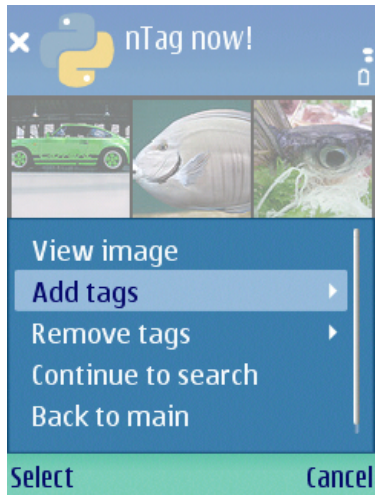

Figure C.3: Choose Add tags menu item

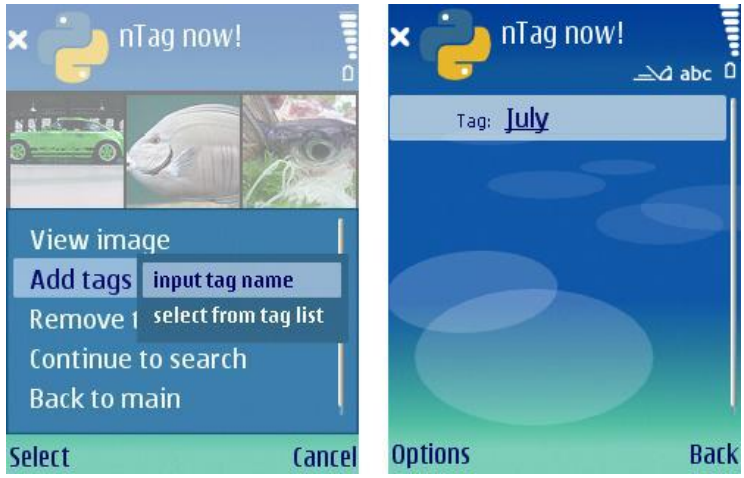

Figure C.4: If choose input tag name, a tag name is input by hand, only one at a time

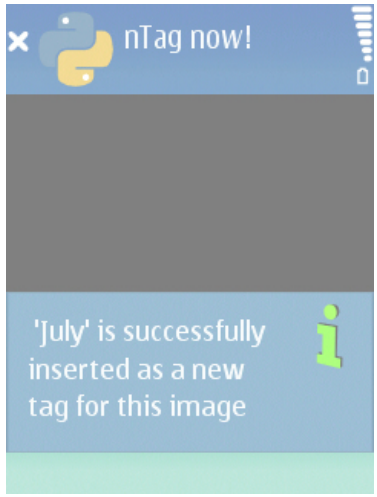

Figure C.5: A note will report the operation is successful

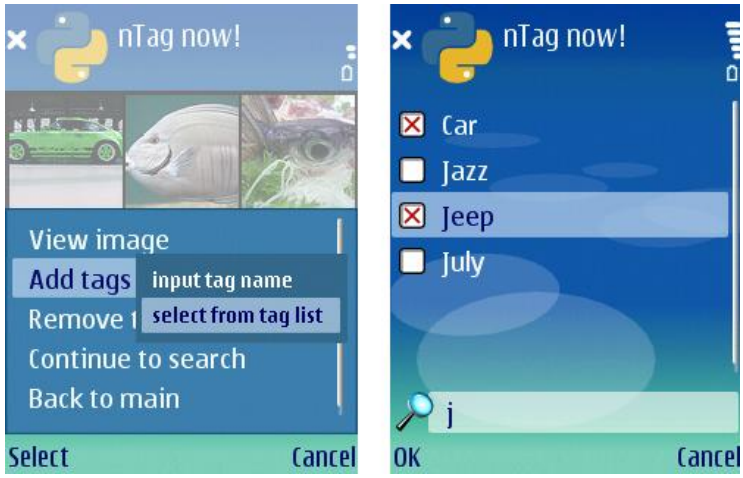

Figure C.6: If choose select from tag list to add tags, multiple tags can be(and only be) chosen from the whole tag list. It supports tag searching

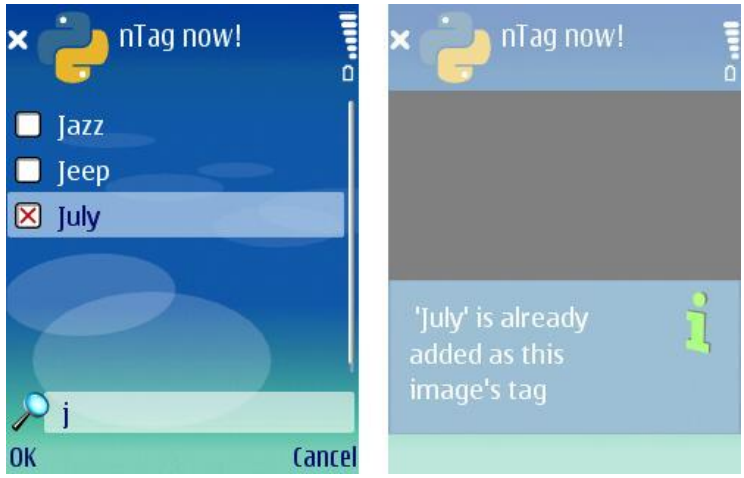

Figure C.7: If the same tag is added to the image again, the operation will be cancelled and a note will explain this to users

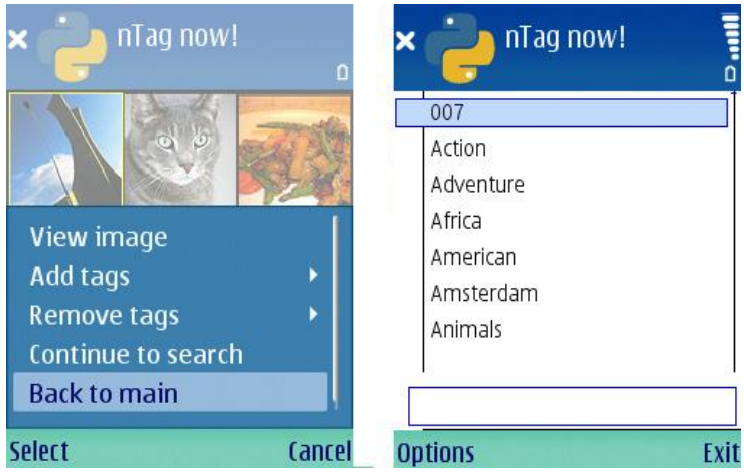

Figure C.8: Use menu to return back to the main interface Tag Text View

Appendix D

A Useless User Manual: Learn to Use the Search Engine

The tag searching method is shown here, and also image exclusive searching strategy – AND-search function.

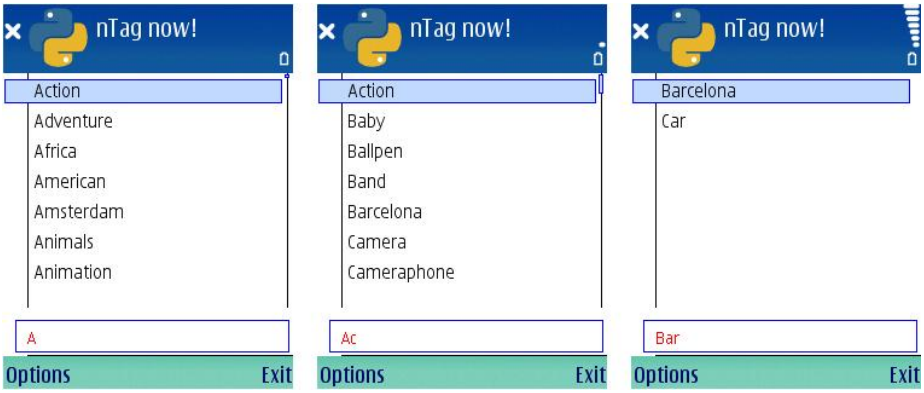

Figure D.1: Start from the main interface, press a key to input the query. In order to look for a tag called "Car", just press numeric key in sequence "227". Query field shows the beginning of the first possible targets.

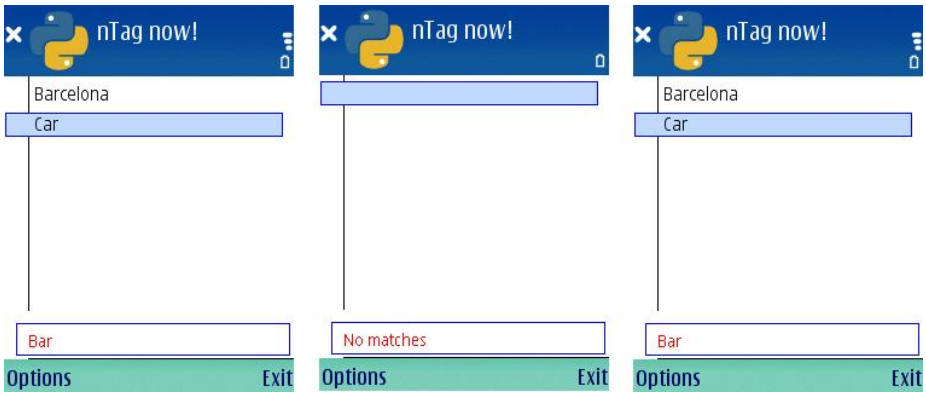

Figure D.2: Use cursor to help choose target; warning will be shown in the query field if the query returns no results, e.g. press key "5" at the left screenshot situation; press BackspaceKey, the search engine will return back to the last valid status.

Figure D.3: Use menu View tagged images to get all images mapped to the chosen tag, so here three cars are shown

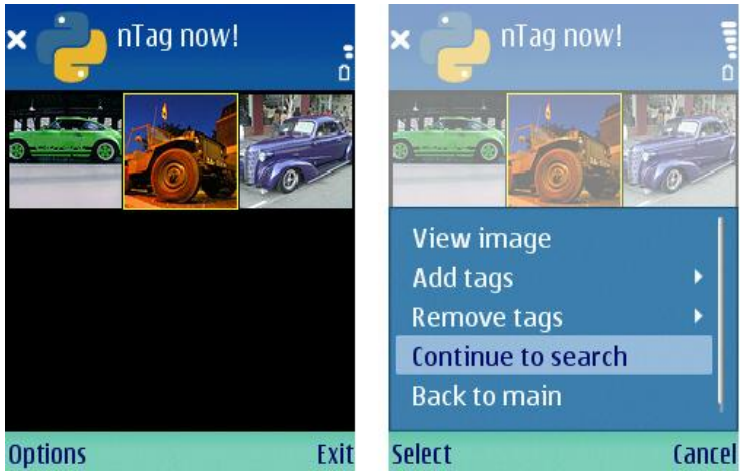

Figure D.4: If you want to get the "jeep" not by browsing as the left screenshot does, you go to menu and choose Continue to search as the right screenshot does

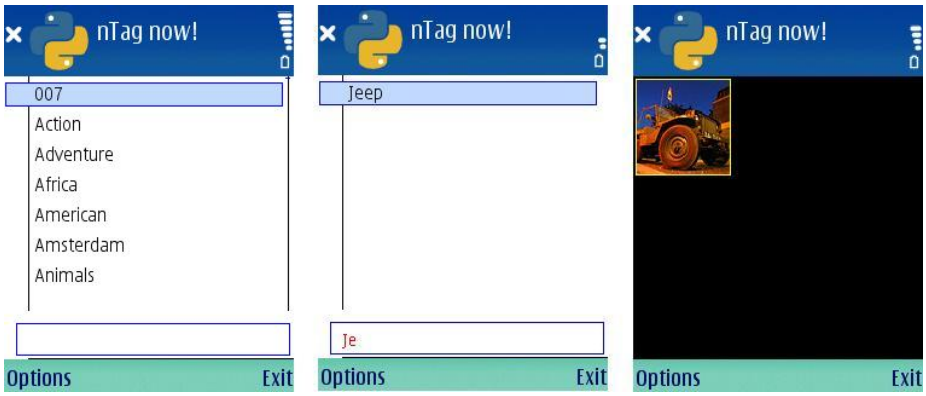

Figure D.5: The GUI goes back to the tag list, you can select a tag using the same method explained in Figure D.1 and use the menu item View tagged images to check the result, similar to Figure D.3. However, this time other cars are removed out of the result. It is because the left image has two tags "car" and "jeep", while the others only contain the "car".

Appendix E

A Useless User Manual: Remove and Delete Tags

Remove here indicates to break a link between a tag and an image. *Delete* means deleting a tag from tag repository. As it is explained many times, if a tag doesn't exist, the links with images disappear at the same moment.

Figure E.1: There are two choices to remove tags from an image, let's try to remove a tag by inputting the tag name. Only one tag can be removed at a time.

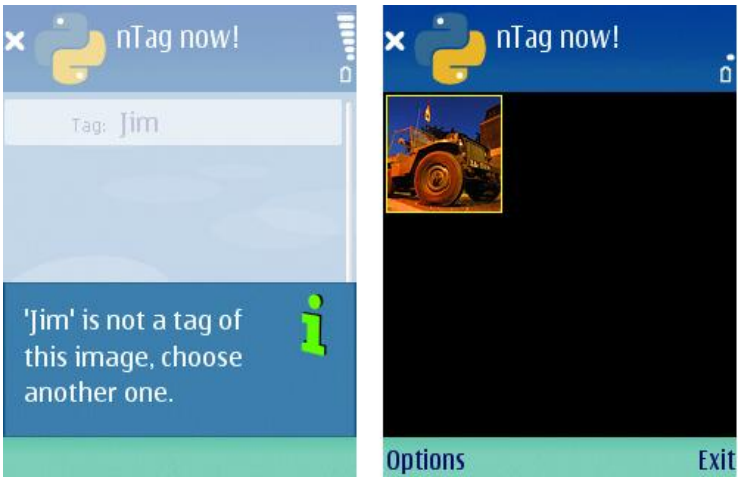

Figure E.2: If input a tag name, which has not been mapped to this image, a message is shown to inform that the operation is cancelled. GUI switches back to the Image Grid View mode.

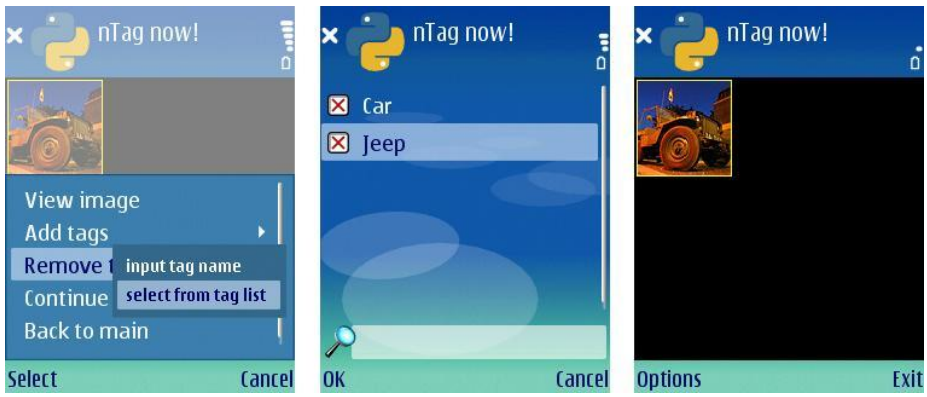

Figure E.3: A better way to do this removing task is using select from tag list. It will show all the tags mapped with this image and this function also supports tag searching and multiple selection.

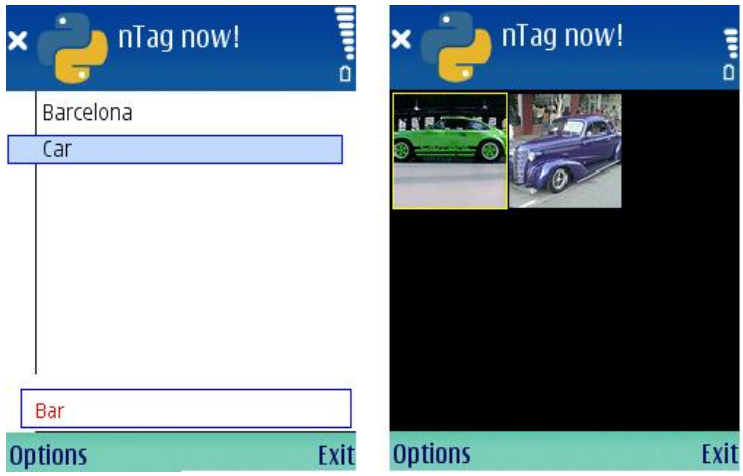

Figure E.4: Once a tag has been removed from an image, the link between them doesn't exist any longer. The right screenshot shows that you cannot find out "jeep" among "car"s.

Figure E.5: Another case is to delete a tag. Here, these three screenshots demonstrate a whole procedure of deleting a tag – "Car". A query is promoted before the deleting operation is executed. After this, "Car" no longer exist in the tag list.

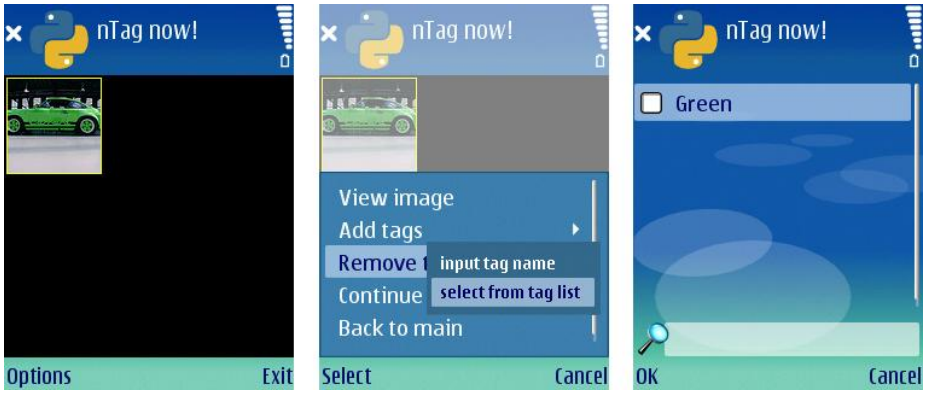

Figure E.6: As expected, as long as the tag is deleted, links between the tag and images vanish. Here, the image of green car only remains a color to describe it.

Appendix F

A Useless User Manual: Take Photos and Organize Now

This is the last part of the manual. It shows how to take pictures under nTag environment. The advantage to do so is that nTag provides a chance to organize the pictures even before you create them. And also after the photos are taken, you still can organize it immediately. But the most valuable part of this section, as well as the whole thesis, is that you have got a chance to see the author himself!

Thanks for reading this useless user manual. Thanks for your patience. What inside this manual have all been well written in the menus and warning messages of nTag, as well as your own image organizing habits.

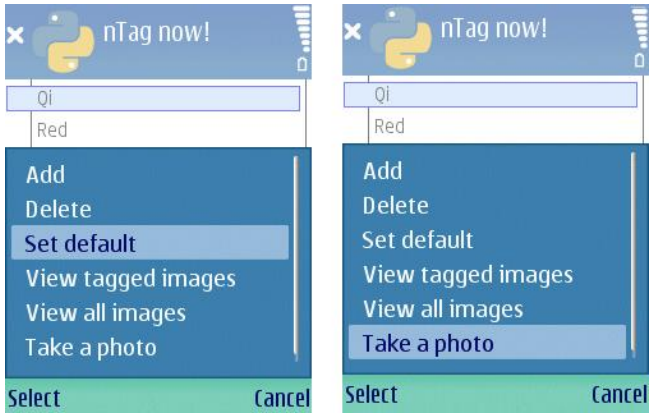

Figure F.1: Before go into the Camera Mode, you can choose to set a default tag to be attached to the photos, which will be taken in the future

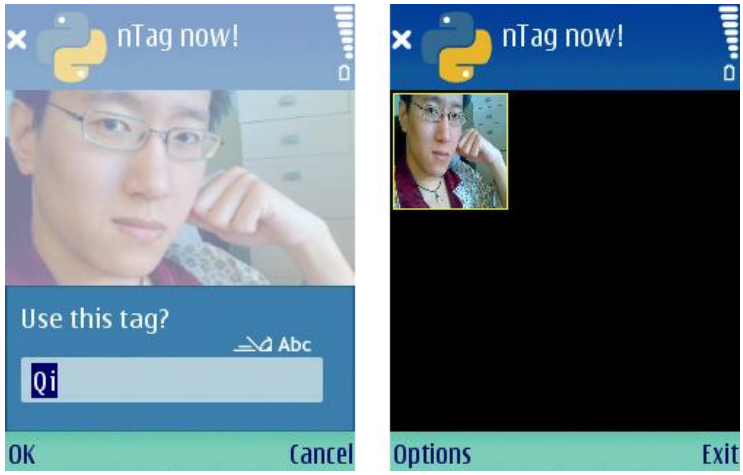

Figure F.2: After a picture is generated, a query jumps out to ask your confirmation on the default tag. The newly taken picture will be put into Image Grid View for your further processing.

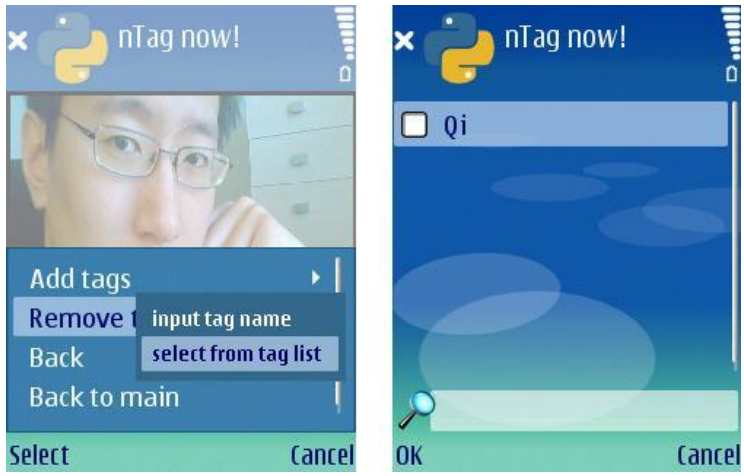

Figure F.3: Choose menu item select from tag list to remove tags, in which way the attached default tag "Qi", my first name, can be seen.

Bibliography

[1] Canalys, symbian: Apple iphone already leads windows mobile in us market share, q3 2007. http://www.roughlydrafted.com, retrieved on 2008-3-24.

[2] http://del.icio.us/help/tags. retrieved 2008-3-30.

[3] http://en.wikipedia.org/wiki/smartphone. retrieved 2008-3-30.

[4] http://en.wikipedia.org/wiki/windowsmobile. Retrieved on 2008-01-20.

[5] http://mobilemaster.wordpress.com/. retrieved on 2008-4-1.

[6] https://www.symbiansigned.com/app/page. retrieved on 2008-4-01.

[7] http://www.dependability.org/. retrieved 2008-3-30.

[8] http://www.microsoft.com/windowsmobile/default.mspx. Retrieved on 2008-01-20.

[9] *http://www.s60.com/life*. Nokia Corporation. Retrieved on 2008-01-20.

[10] http://www.sonyericsson.com/cws/products/mobilephones/overview/p1i. retrieved on 2008-4-1.

[11] http://www.t9.com/. retrieved on 2008-4-1.

[12] http://www.useit.com/papers/responsetime.html. retrieved 2008-3-30.

[13] Symbian fast facts q4 2007. http://www.symbian.com/about/fastfacts/fastfacts.html, retrieved on 2008-3-24.

[14] http://www.flickr.com/about/. 2008. retrieved 2008-3-20.

[15] Mihai Alexandru. Has iphone started a touch screen mobile phone war? Feb. 2007. http://www.playfuls.com/, Retrieved on 2008-02-20.

[16] Larry Bernstein. Taking software requirements creation from folklore to analysis. *ACM Ubiquity*, 7, Feb. February.

[17] Dines Bjørner. *Software Engineering 3: Domains, Requirements, and Software Design*, volume 3 of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, 10 2006.

[18] LiMo Foundation. Introduction, overview and market positioning. February 2008. Updated for Mobile World Congress.

[19] Linux Foundation. Linux online - about the linux operating system. http://www.linux.org, retrieved on 2008-01-20.

[20] Apple Inc. http://www.apple.com/iphone/. retrieved on 2008-4-1.

[21] Apple Inc. Apple announces iphone 2.0 software beta. March 2008.

[22] Sun Microsystems Inc. Java me at a glance. *http://java.sun.com/javame/index.jsp*. Retrieved on 2008-02-20.

[23] InfoTrends. Mobile imaging study results. 2006.

[24] Søren Svane Jensen. mobtag – mobile tagging. Master's thesis, Technical University of Denmark, Kgs. Lyngby, 2007.

[25] Jakob Eg Larse. *NEXUS – a unified approach to personal information management in interactive systems*. PhD thesis, Technical University of Denmark, 2005.

[26] Kevin Maney. Baby's arrival inspires birth of cellphone camera - and societal evolution. 2007.

[27] R. B. Miller. Response time in man-computer conversational transactions. 33:267–277, 1968.

[28] Inc. Motorola. http://www.motorola.com/shop.jsp. retrieved 2008-3-308.

[29] Jakob Nielsen. Why you only need to test with 5 users. *http://www.useit.com/alertbox/20000319.html*. retrieved on 2008-3-24.

[30] Nokia. http://www.nseries.com/. retrieved on 2008-4-1.

[31] Emily Raymond. Camera phones increase resolution, broaden appeal. October 2004. http://www.digitalcamerainfo.com/content/Camera-Phones-Increase-Resolution.htm.

[32] Suzanne Robertson. An early start to testing: How to test requirements. presented at EuroSTAR '96, Amsterdam, December 1996.

[33] B. Scifo. The domestication of camera-phone and mms commnunication. 2004.

[34] Bernado A. Hubman Scott A. Golder. The structure of collaborative tagging systems.

[35] Symbian. *Symbian OS v9.5 product sheet.* Symbian Software LTD., http://www.symbian.com. P95-001-C-2007.

[36] www.abiresearch.com. By 2013 one in every three phones sold will be a smartphone. March 2008. retrieved on 2008-3-25.