TAMPERE POLYTECHNIC
Computer Systems Engineering, Software Engineering

Lasse Leiniö

TESTING OF S60 SEARCH APPLICATION

Final thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering.

Tampere, May 2nd, 2007

Supervisor:	Principal lecturer Matti Ilmonen
Instructor:	Site Manager Tero Jokinen

Author:	Lasse Leiniö
Title:	Testing of S60 Search application
Date:	2.5.2007
Pages:	26
Keywords:	software, testing, search, Series 60
Degree Programme:	Computer systems engineering
Specialization:	Software engineering
Supervisor:	Principal lecturer Matti Ilmonen
Instructor:	Site Manager Tero Jokinen

Competition between mobile phone manufacturers has been intense in recent years. In order not to lose market shares, smartphone vendors are delivering their products with new technologies at a faster pace. Decreasing delivery times have been adding extra challenges especially in quality assurance.

Testing is one of the best ways for ensuring the software quality. Appreciation of software testing has increased and software development projects have been investing more in testing services.

Features and services of the modern smartphones are close to the ones provided by laptop computers. Data transfer speeds have increased in wireless communication and the smartphone end users are provided with more data storage capacity. Increased data amounts have created a need for a way to search for the stored data. In-Device Search application gives one solution to finding locally stored data from the Series 60 smartphones.

This thesis focuses on the testing of the In-Device Search application, which was delivered on Series 60 platform. The graphical user interface and basic functions are described in detail. System level test planning and test execution are represented and test automation in the testing of In-Device Search is described.

Tekijä:	Lasse Leiniö
Työn nimi:	Testing of S60 Search application
Päivämäärä:	2.5.2007
Sivumäärä:	26
Hakusanat:	ohjelmistotestaus, hakusovellus, Series 60
Koulutusohjelma:	Tietotekniikka
Suuntautumisvaihtoehto:	Ohjelmistotekniikka
Työn valvoja:	Yliopettaja Matti Ilmonen
Työn ohjaaja:	Site Manager Tero Jokinen

Laitevalmistajien välinen voimakas kilpailu nopeasti kasvavilla matkapuhelinmarkkinoilla on johtanut siihen, että uusimpia teknologioita tukevat tuotteet on saatava yhä nopeammin markkinoille. Palvelujen monipuolistuminen ja toimitusaikojen lyhentyminen ovat tuoneet uusia haasteita laadunvarmistukseen. Tuotteen laatu onkin noussut yhdeksi tärkeimmistä myyntiartikkeleista erityisesti älypuhelinmarkkinoilla.

Yksi parhaimmista ohjelmiston laadunvarmistuksen menetelmistä on ohjelmistotestaus. Ohjelmistotestauksen arvostus onkin noussut viime vuosina ja ohjelmistokehitysprojektit ovat panostaneet testaukseen entistä enemmän.

Modernit älypuhelimet ovat toiminnoiltaan ja palveluiltaan jo lähes kannettavien tietokoneiden tasolla. Langattomien yhteyksien nopeutuminen ja matkapuhelinlaitteiden tallennuskapasiteetin nousu ovat luoneet tarpeen suurten tietomäärien hallinan helpottamiseen. Tätä tarvetta vastaan Series 60 alustalle on tehty hakusovellus tiedon löytämisen helpottamiseksi.

Tässä työssä kuvataan Series 60 alustalle toimitettavaa hakusovellusta ja erityisesti sen testausta. Työssä keskitytään järjestelmätason testauksen suunnittelun ja toteutuksen selvittämiseen. Lisäksi kerrotaan testausautomaatiosta ja siitä, kuinka testausautomaatiota käytettiin apuna hakusovelluksen testauksessa.

FOREWORD

This thesis has been done for Ixonos Plc during spring 2007. Site Manager Tero Jokinen from Ixonos Plc has been the instructor for the work. Principal lecturer Matti Ilmonen from Tampere Polytechnic has supervised the thesis.

I want to thank my supervisor Matti Ilmonen for support throughout the thesis writing process. I thank my instructor Tero Jokinen for giving the opportunity to do this work. I want to thank all my colleagues in the software testing team and other project personnel for their feedback. Also thanks belong to Gareth James for proof-reading the thesis.

Tampere, May 2007

Lasse Leiniö

ABBREVIATIONS

BAT	Basic Acceptance Testing
Black-box testing	Testing method where testing is planned and executed without knowledge of the actual software implementation
Error	Failure in software
Highlight	Way to indicate focus on screen of a mobile device
IMAP4	Internet Message Access Protocol 4
I/O	Input / Output
MMS	Multimedia Message Service
MTP	Master Test Plan
N/A	Not Available
OS	Operating System
Pane	Area on Series 60 graphical UI
POP3	Post Office Protocol 3
QC	Quality Center, a testing and test automation tool provided by Mercury Interactive
QTP	Quick Test Professional, software product used in test automation
S60	Series 60, a Symbian based software platform
SIS	Symbian installation file
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
Softkey	Software configurable key, which function is shown on screen of a mobile device
Symbian	A software licensing company, which provides operating system for mobile devices
UI	User Interface
USB	Universal Serial Bus, a connectivity method
White-box testing	Testing method where tests are based on the design or the structure of the software implementation.

1. INTRODUCTION

Mobile phone development and the technological evolution of the mobile business has been rapid in recent years. There is a need for a mobile device, which in addition to key mobile phone features, has the similar communication and data handling capabilities as a personal computer. This means, that mobile devices need to support more applications and different data types as well as means to hold large amount of data. Whit data storage capacities constantly increasing, users must have a way to organize and locate the information within their mobile device.

Symbian is one of the most popular mobile device operating system. The Symbian based Series 60 platform provides many smartphone applications, which are designed to be user friendly and to make multiple mobile business services available to the end users. One of the S60 applications is In-Device Search, which offers one solution for finding user data from a mobile device. The application gives an option for the end user to search for information from the stored data and to open it in an application associated with the data type.

The Mobile phone market has been continuing to grow and competition between mobile service providers has been intense. One of the key factors in this success has been the quality of a product and this has made smartphone vendors and mobile phone manufacturers to invest more in software testing services. As one of the best ways for ensuring the software quality, testing can define when the maturity of a product is good enough for market. Appreciation of software testing has increased in recent years and software development projects are investing more resources in test planning, design and execution.

This thesis focuses on high-level testing of the In-Device Search application. Low level testing is only mentioned in testing levels and test planning parts of the thesis. Series 60 platform and In-Device Search testing environment are briefly introduced. Manual and automated testing of the application is described in detail.

2. SERIES 60

Symbian OS is an open operating system, which is designed for mobile devices. As a mobile device operating system, Symbian OS has to provide services with limited resources such as memory restrictions and low power consumptions. Also mobility and continuous usage add more challenges for a mobile device operating system. /1/ /3/

Symbian provides resources for platforms, which need support for e.g. messaging, security, networking, telephony and other real-time communication. Multiple platforms are based on Symbian OS and more than 100 million Symbian smartphones were shipped in 2007. /1/

2.1 Series 60 platform

One of the platforms running on Symbian OS is Series 60. Series 60 platform provides multitasking support between applications and a UI for switching between the running applications. /3/

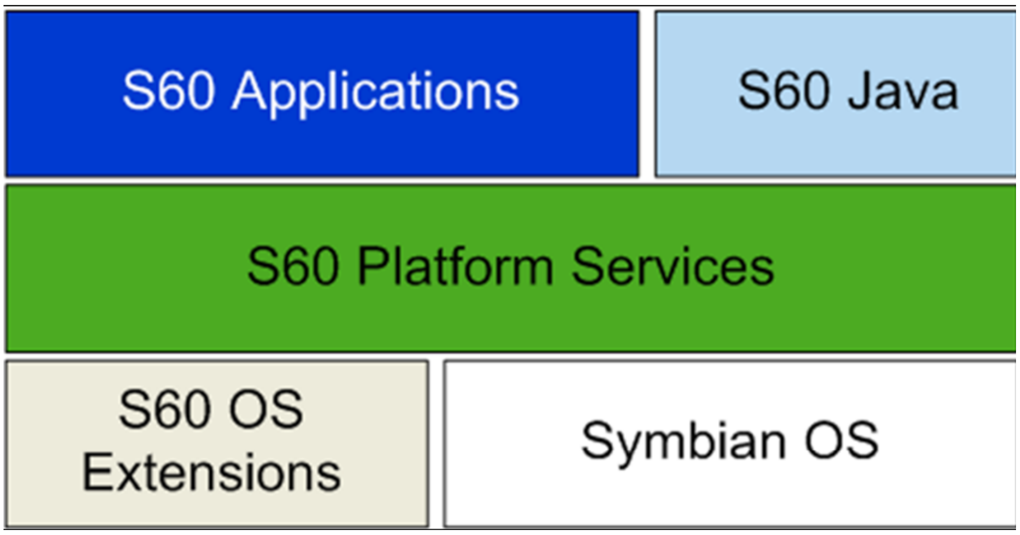


Figure 1: High level architecture of Series 60 /3/

S60 platform is based on Symbian OS and it provides services for S60 applications, S60 java and S60 OS extensions as depicted in figure 1. S60 application services are base for features, which can be utilized by applications. /2/ /3/

2.2 Series 60 Graphical User Interface

Graphical area, which displays the application on the screen, is called an application window. The application window is usually divided into panes. Standard panes in the application window are the main pane, status pane and control pane. Figure 2 shows an application window and the standard panes on Application menu. /4/

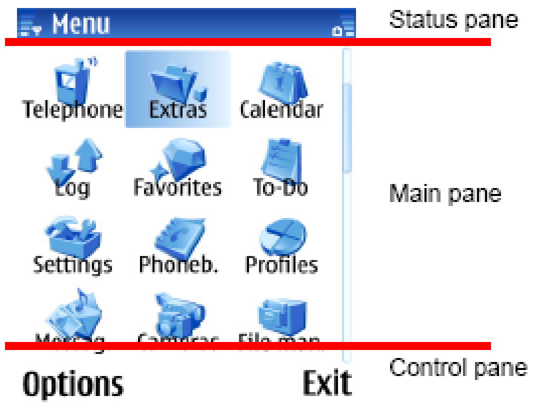


Figure 2: Standard panes in an application window are main pane, status pane and control pane. Application menu is shown in main pane. /4/

Standard panes are further divided into sub panes. Status pane has for example a context pane, title pane and a navi pane. Figure 3 shows the division of the status pane. /4/

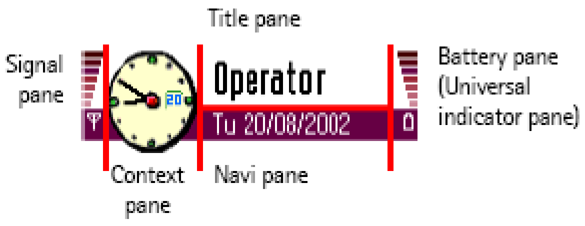


Figure 3: Status pane is divided into five sub panes. /4/

The control pane is shown in the bottom part of the application window. Softkey labels are displayed in the control pane. /4/

3. TESTING IN A SOFTWARE DEVELOPMENT PROJECT

The main target of software testing is to find errors in the software. Other important task of software testing is to give reliable information of the current state of the software. Testing can be used to measure and improve the quality of the software being tested. Some basic concepts in software testing are test case, test set and error. /5/

Test case is a single entity, which consists of preconditions, test steps and expected results. Preconditions define what kind of environmental setup must be prepared and in which state the software must be before the test execution can be started. For example needed test data is defined in the test case preconditions. Test steps are consecutive instructions made for a tester to complete the test. The actual results of a completed test step are checked against the expected results and test case status is set accordingly.

Test set is a collection of test cases. Test set is usually planned to cover some area of the software and the amount of test cases in a test set depend on how much information from the software is needed. Test cases can be inserted into a test set in a specific order, which makes the test execution more convenient and faster.

Errors are found during testing, when the test execution doesn’t produce the expected results. An error is said to be a deviation from a specification, which was used to create the test cases. /7/

It is good to understand, therefore, that there is no such thing as a piece of error free software. Testing can point out, that there are errors in software, but testing can not point out, that there are no errors in software. Software development projects have limited resources and time and testing has to be stopped at some point. With good test planning, all major errors can be found before the software product is released. /7/

3.1 Testing levels

Testing is usually performed on many levels according to V model of testing, which is shown in Figure 4.

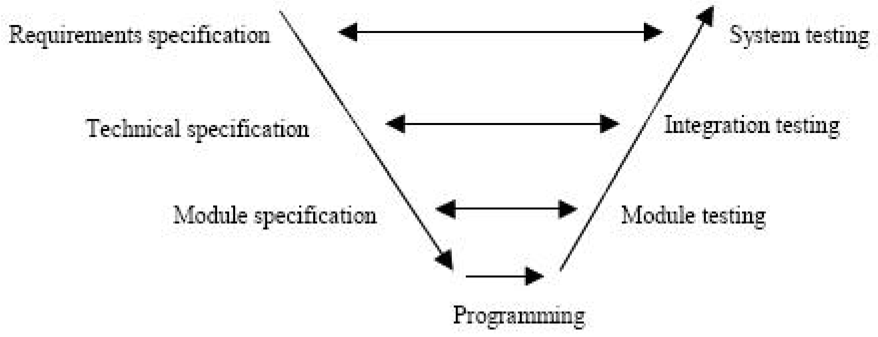


Figure 4: V model of software testing. /7/

The V model of testing is based upon a waterfall model of software development. By applying the V model of testing in test planning, the amount of overlapping testing is reduced and testing is better taken into account in the entire software development life cycle.

A single module is tested in module testing, which is also called as unit testing. A module is a program entity, whose size is usually 100 – 1000 lines of code. Module testing is usually performed by the software developer who has implemented the module. Module testing is based on module design specifications and it includes tests for basic I/O operations, functions, conditional statements, etc. /7/

Integration testing is based on technical specifications and it’s performed simultaneously with the module testing. Testing is focused on the interfaces of the modules and interactions between different modules and module groups. Testing is performed by software developer or specific integration tester. /7/

System tests are run on a system, which has all of the software modules integrated. Testing is done by software testers who have no knowledge of the actual implementation. Tests are designed against the requirements, so when software implementation changes, the same test cases can be used if requirements remain the same. The most common tasks for software testing organizations and software testing engineers are system level test planning and test execution. /7/

3.2 Testing techniques

In Black-box testing, the test cases are created according to specifications not implementation. Input is entered into a system under test and output is analyzed without knowledge of the actual process. Some parts of the software, especially unspecified functionalities, are most probably not tested with blackbox method. /6/

White-box testing is performed in the same way as black-box testing, but white-box testers have knowledge of the system, the design or the structure of the code. With white-box testing it is possible to see how the system works in addition for making sure that the system does what it is supposed to do. /6/

Testing is not biased by the knowledge of the implementation in black-box technique and white-box testing provides possibility to test all of the ways the system works. Testing effectiveness and quality are clearly increased, when both of the testing techniques are taken into use in software development project. /6/

3.3 Benefits of test automation

Usually tests are designed to be run manually, which means that tests are planned to be run by a person. This restricts the test design, because long lasting tests and repetitive tests are usually demotivating and hard to run by a human tester. Tests, which are difficult for a human, may be easy for a computer. In test automation, test cases are executed by a computer without human intervention.

Test automation can be used to automate manual test cases. This is beneficial, because automated tests can be run parallel with manual test and it frees manual testing resources to other tasks. More benefits of test automation will be gained, when automated tests are designed differently from manual tests.

Long period tests and repetitive tests are not reasonable to be run by a human tester, when they can be automated. This way, the same test can be repeated thousands of times or with thousands of different parameters. Test inputs can be placed into tables from which an automated test procedure can fetch the data and enter it into a program under test. Automated test can later verify that the expected results are produced by the program. Another benefit of test automation is that automated tests can be run overnight. /8/

4. IN-DEVICE SEARCH APPLICATION

The services available for mobile devices are not far from the ones, which can be used by modern laptop computers. Mobile phones can now utilize high volume memory cards and end users may create, download and receive data up to several gigabytes. With older mobile devices, it was relatively easy to find a stored contact from SIM card or from a sent short message, because memory storage was limited to dozens of entries. Now, only after a few months of normal business usage, phone may contain thousands of emails, calendar items, files etc. and it could be time consuming to try to find out some specific piece of information from the vast amount of data.

In-Device Search is an application, which provides the mobile device user a solution for finding locally stored data. User can search for textual strings from both the phone memory and memory card. In-Device Search has a simple UI, which allows user to limit the search to certain categories.

4.1 Searchable data

Selectable content classes are messages, emails, contacts, calendar events, notes, to-dos and files. User can choose one, several or all of the content classes to be included in search.

First content class is messages, which includes short messages (SMS) and multimedia messages (MMS). Short message is a text message, which has less than 160 characters. SMS messages are usually sent and received via mobile devices. Multimedia messages are similar to SMS, but they can also contain images, sounds and videos.

Messages are searched from specific folders in the Messaging centre application. The messaging centre has several folders, but search can only be performed on messages, which are to found in the inbox, my folders, drafts or the sent folder. When message is received, it is added to the inbox folder. My folders or user created folders can be used to save messages. Unfinished messages are stored in the drafts folder and sent messages in the sent folder.

The content of a message is included in the search, but textual strings can also be searched from both the receiver and sender fields of messages. Also the subject, if any, of a MMS can be searched.

The calendar content class includes meetings, memos and anniversaries, which are stored in Calendar application of a mobile device. Subject and location of a meeting event can be searched as well as subjects of memos and anniversaries.

Notepad notes can be searched by selecting notes content class. Saved text files and notes are listed in Notepad application.

Received emails can be searched from remote mailboxes, which are defined in the Messaging centre. Along with messages, emails can also be searched from inbox, my folders, drafts and sent folders. Mailboxes can be set up to use IMAP4 or POP3 protocols and search can be done on both mailbox types.

Text can be searched from both the email body and subject fields. Also sender and receiver can be searched. Receiver information can also be searched from circulated copy and blind circulated copy fields of a sent email.

The Phonebook application holds contact details and contact groups, which can be searched by selecting the contacts content class. Contact card has many information fields and search can be done for content of all textual fields. Most common fields defined for a contact are first name, last name, mobile phone number and email address. Also fields like company name, job title, fax number and postal address are quite common. There is no limit, other than memory constraints for adding information fields for a contact and most of the fields are searchable. Phone may contain contact data as business card files, but they can be found by selecting other files content class.

To-dos or tasks are stored in Calendar application and their subject can be searched. Not completed and completed to-dos as well as to-dos with all priorities can be searched.

All files in user area, which are not in system or hidden folders, can be searched by their filename. Folder names are not included in search.

4.2 Graphical user interface of In-Device Search application

In-Device Search application has three different views. These are Main view, grouped results view and single results view. All three views of In-Device Search application are described in the following chapters. Application specific help is available for most of the Series 60 applications and In-Device Search has help topics for each of the three views. Also content specific viewers and editors can be opened from search results.

4.2.1 In-Device Search main view

In main view, user is shown a selection list of content classes. These content classes are the areas, from which the data can be searched. The topmost item on the list can be used to select or deselect all of the content classes. One or several items can be selected from the selection list. When the In-Device Search application is opened, all content classes are selected by default. An input field is shown in the bottom of the main view and it’s used for entering the search string. Search string can contain many keywords, which can be partial words as well. When keywords are separated by a white space, the Boolean AND operation is applied between each keyword.

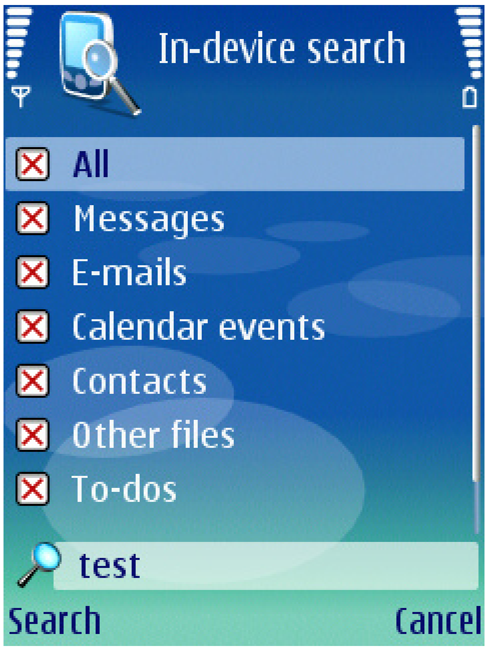


Figure 5: Content classes can be selected and keywords entered in In-Device Search main view.

If the user wants to perform search with keywords without using AND operation, an exact search can be used. This is done by entering the search string within quotations. An exact search produces results, which items contain the whole search string including the white spaces.

Also commonly used wildcards can be used in search string. Several characters can be replaced by an asterisk (*) and a single character can be replaced by a question mark (?).

4.2.2 Grouped results view

Grouped results view shows the amount of matches, which have been found in each chosen content class. The search string used to generate the results is shown in the title pane and the total amount of matches are shown in the navi pane. The amount of matches in each content class is shown in brackets after the content class name.

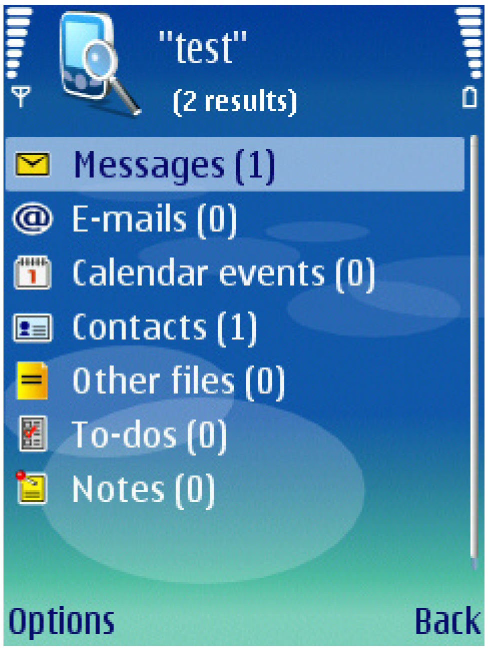


Figure 6: Grouped results view shows amounts of matches in each content class.

After search is started in Main view, the grouped results view is opened and search animation is shown in the context pane. Navi pane indicates when a search is ongoing and an ellipsis is shown in brackets after each content class. The ellipsis is replaced with the match amounts, after the search has been completed for the content class.

Focus can be moved over one of the content class. More detailed, content class specific search information is available, when one of the content classes is selected.

4.2.3 Single results view

Single results view is opened, when search is performed for only one content class or one of the content classes is selected in Grouped results view. Title pane shows the used search string and navi pane displays the opened content class. Navi pane shows also the total amount matches in the content class and position number of the highlighted item.

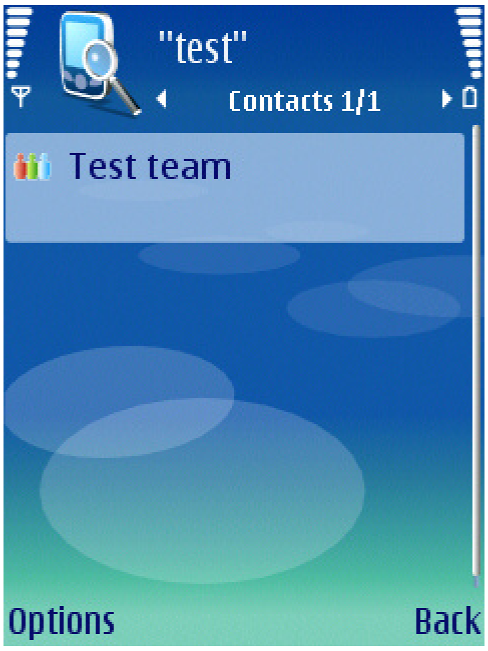


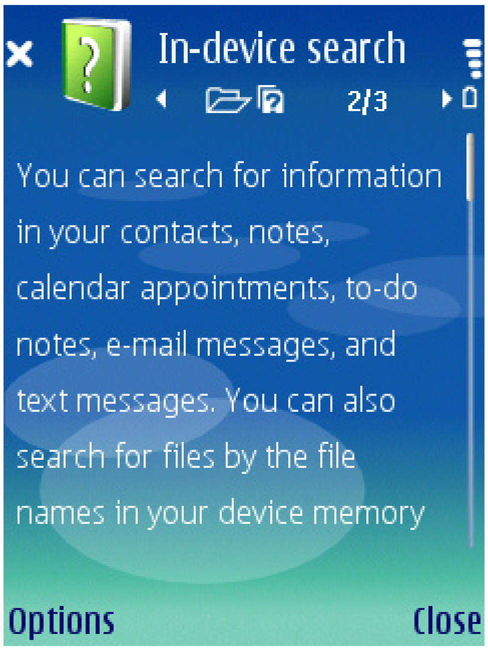
Figure 7: Single results view shows items, which content matches with the search string.

If search was performed with more than one content class selected, it is possible to see the results of other content classes by pressing left or right arrow key. This way user doesn’t have to go back to the Grouped results view to see the results from other content classes.

When item is selected in Single results view, the viewer or editor of the content class specific application is opened. For example, when found contact is selected in Single results view, Phonebook application is opened and details of found contact is shown.

4.2.4 In-Device Search help

It is common in Series 60 devices, that application specific help topics are available. Help is available for end user in all In-Device Search views. Figure 8 shows help topic opened from the main view.



You can search for information in your contacts, notes, calendar appointments, to-do notes, e-mail messages, and text messages. You can also search for files by the file names in your device memory

Figure 8: Help is available for end user in all In-Device Search views.

Help application contains information about each of the In-Device Search views. Some of the functions are described and advice on using the application is provided by Help application.

5. TESTING ENVIRONMENT

In-Device Search tests are run on Nokia E50 smartphone and the test cases are read from Quality Center. Quality Center is a software product of Mercury Interactive Corporation and it's used for creating, maintaining and running tests. Quality Center, which is abbreviated as QC, is used via web browser interface. Test cases are designed in QC and they contain test description, preconditions and test execution steps. /5/

Before test cases are used in any test set, they are exported into Word documents, which are used in test specification inspections. After test specifications are approved, the test cases can be selected into test sets. Test sets are created for every software release of In-Device Search application and test results are gathered and exported into test results documents from QC.

Usually In-Device Search software is installed on S60 device by the use of a Symbian installer file. Symbian installer files are provided by In-Device Search application developers.

5.1 Testing tools

Test data is added into device used in tests by specific script files. Test data scripts are created in PC environment and they are later sent to mobile phone. Mobile phone has a client application which fills the phone with data specified in the script file. Some other Series 60 tools are used for testing e.g. memory consumption of the application.

6. TEST AUTOMATION IN TESTING OF SEARCH APPLICATION

In-Device Search has simplified UI, which allows end user to access search results more quickly. Simple UI and an existing test automation environment were main drivers for the decision to use test automation in In-Device Search testing. Test automation was planned to be used in typing text in input field, making key presses and reading text from the screen.

6.1 Mercury QuickTest Professional

Mercury QuickTest Professional, which is abbreviated as QTP is a test automation and testing tool made by Mercury Interactive. The test automation program used in testing of In-Device Search application was created with VBScript on QTP. VBScript is scripting language, which syntax is a variation of Microsoft Visual Basic programming language. /9/

QTP supports creation of multiple test iterations and data handling between databases, spreadsheets and text files. Program parameters for the test automation program were fetched from Microsoft Excel sheet and the results were written into a text file. QTP reporting functions were used to indicate status of test run in QC and in case of a failure, more detailed explanations of the failure were reported. /9/

6.2 Test automation environment

Test automation environment consists of mobile device, which is connected via USB cable to PC, which runs the actual test automation program. Test automation program was created with VBScript and it is run by QTP. Test automation program for example sends key event actions to mobile device and mobile device responds by sending textual data from the device UI. Automation specific client software is running on mobile device handling the communication towards PC.

Test case made for launching the test automation program was created in QC. Test execution was started and statuses of the test runs were reported in QC as well.

6.3 Test automation program

The basic idea of the test automation program is to enter randomly made, but predefined keywords to In-Device Search application and to check, that correct amount of matches are shown in Grouped results view. This means, that keywords must be created from the same data, which is in the mobile device. Test data is collected to specific script files, which are later sent to the mobile device. Another Symbian based tool is used to fill the mobile device with the data from the script files.

First task in preparing the test automation setup is to create search strings from the test data file and then count the amount of matches for each string. The amounts counted at this phase are stored for the post processing, which is done after the test automation program is run.

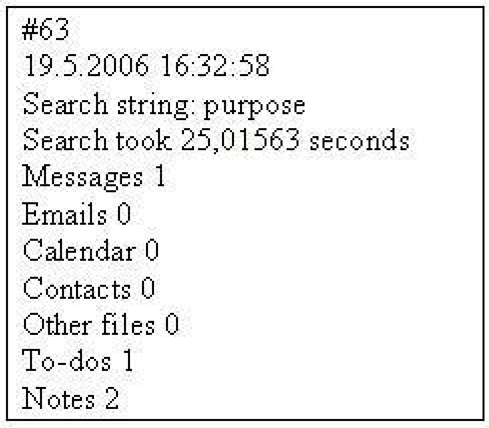
The search strings are stored into a search string array, which is sent as a parameter to the test automation program. The program takes two more parameters, which are the total amount of search strings and a file name of the results file.

The test automation program starts by launching In-Device Search application. After In-Device Search is opened, the program verifies that the correct application is launched and it is opened in the correct state. Verification is done by checking the title and the softkeys of the main state.

In the main state the first search string in the search array is entered into the input field. Search is started and timer value is read for search duration calculation. Search string is written in the results file together with current date and time information. After this the program waits until all results are printed on the screen and the search is completed.

After search is completed, the search duration is calculated and it is stored into the result file. If search takes too much time, it is also reported in the results. Amount of matches are shown in brackets in the grouped results view and they are stored into the results file next to the corresponding content class.

Table 1: Result of one search iteration, which has been produced by the test automation program.



```
#63
19.5.2006 16:32:58
Search string: purpose
Search took 25,01563 seconds
Messages 1
Emails 0
Calendar 0
Contacts 0
Other files 0
To-dos 1
Notes 2
```

Table 1 shows one example of search results, which are stored into a result file. First row shows the number of the iteration, second row show the time stamp and third row the entered search string. Fourth row shows how much time the search has taken. Next seven rows show match amounts in each content class.

To make sure, that correct search string was used, the title pane of grouped results view is read and its content is compared with the entered search string. If the title is different than the search string, it is reported to the results.

After results are read, program goes back to the main view and verifies the state from title and softkeys. The next search string is fetched from the search string array and it is entered into the input field. After search is started the results reading procedure is repeated for all search strings in the array.

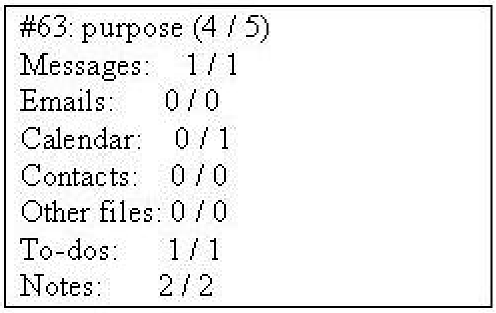
6.4 Analyzing the test automation results

After search match results are produced by the automation system, the results are compared with the ones calculated from the test data file. If there are any differences in the match amounts, it’s prompted in post processing. The keyword, which caused the difference in the results, is checked again manually.

Test results analysis produce an analysis file, which contains results of all iterations, which caused unexpected result amounts. One entry of results analysis file is shown in table 2. First row shows the number of the iteration, the search string and total amount of matches found by the test automation program and total amount of expected matches. Next rows show the results in each content class.

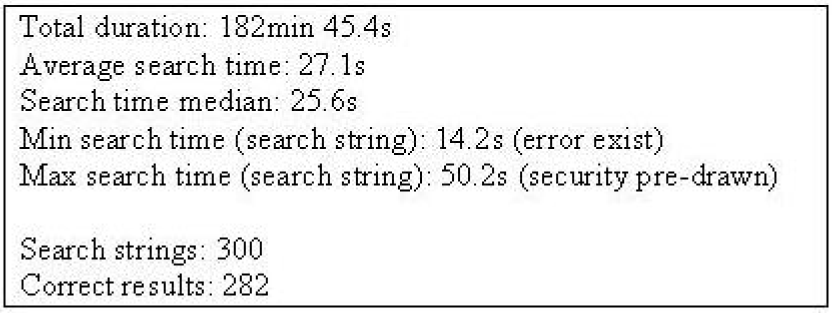
In example results shown in table 2, search string “purpose” is 63rd iteration and it has been found 4 times by the test automation program. Expected amount of matches has been 5 and due to the difference, this entry has been reported in results analysis. For some reason the test automation program has not found any items containing the search string from calendar events. After this, the word “purpose” will be manually searched from the calendar events and reason for the unexpected result will be investigated.

Table 2: Different match amounts have been found in analysis. Calendar event contains a textual string which was not found by the test automation program.


```
#63: purpose (4 / 5)
Messages:    1 / 1
Emails:      0 / 0
Calendar:    0 / 1
Contacts:    0 / 0
Other files: 0 / 0
To-dos:      1 / 1
Notes:       2 / 2
```

After all iterations have been analyzed, some overall statistic is collected. The first row in the beginning of the analysis file shows the total duration of the automated search run. An example of the overall statistics is shown in table 3. The second row shows an average value and the third row shows a median value of the times of all search iterations. The fourth row shows the minimum and the fifth row the maximum search time of a single iteration. Search strings of the minimum and maximum times are shown in brackets after the search times. Also total amount of search iterations and total amount of correct results are shown.

Table 3: Test automation analysis provides overall information about the test run



```
Total duration: 182min 45.4s
Average search time: 27.1s
Search time median: 25.6s
Min search time (search string): 14.2s (error exist)
Max search time (search string): 50.2s (security pre-drawn)

Search strings: 300
Correct results: 282
```

After test results of automated test run have been analyzed, the search strings which caused unexpected result amounts are checked again manually. In most of the cases the results are correct after manual retesting. Sometimes there might be some unexpected events during a test automation run, which may cause for example a missed key press and this can be seen as an error in the test results. Programs may also crash during a long lasting test run especially, when software is in early phase of software development. This stops the test run and makes QTP to take a screenshot, which usually clarifies the cause of the failure. After this, the error case is reproduced manually and the error will be reported as an error in automation environment or as an error in software under test.

7. PLANNING TESTS FOR IN-DEVICE SEARCH APPLICATION

Test planning is as important for testing as project planning is for a project. Planning is started by creating a Master Test Plan. Test planning follows the V model of software testing and it starts from system-level test specifications down to the module test plan.

7.1 Master Test Plan

First step in test planning is to create the Master Test Plan. When requirements specifications and project plan are being developed, the Master Test Plan should be started. This way testing is taken into account in the early phase of the project and it may affect on content of the project plan as well. /6/

Master Test Plan deals with important testing related issues such as resource utilization, risk analysis, schedules, responsibilities, test method definitions, entry and exit criteria, references, testing strategy, test environment and tools, approvals, suspension criteria and resumption requirements. Sections in Master Test Plan may vary depending on areas, which are considered to be relevant for testing in the current project. /6/

After Master Test Plan is created and requirements are complete, detailed test planning can be started.

7.2 Test specification

In-Device Search application system level test specifications are created against the UI specification and feature requirements. UI specification defines all application states, their functionalities and UI design. All UI elements, graphics and textual strings of the application are defined in UI specification. Requirements are provided as a list of application requirements, features and sub features.

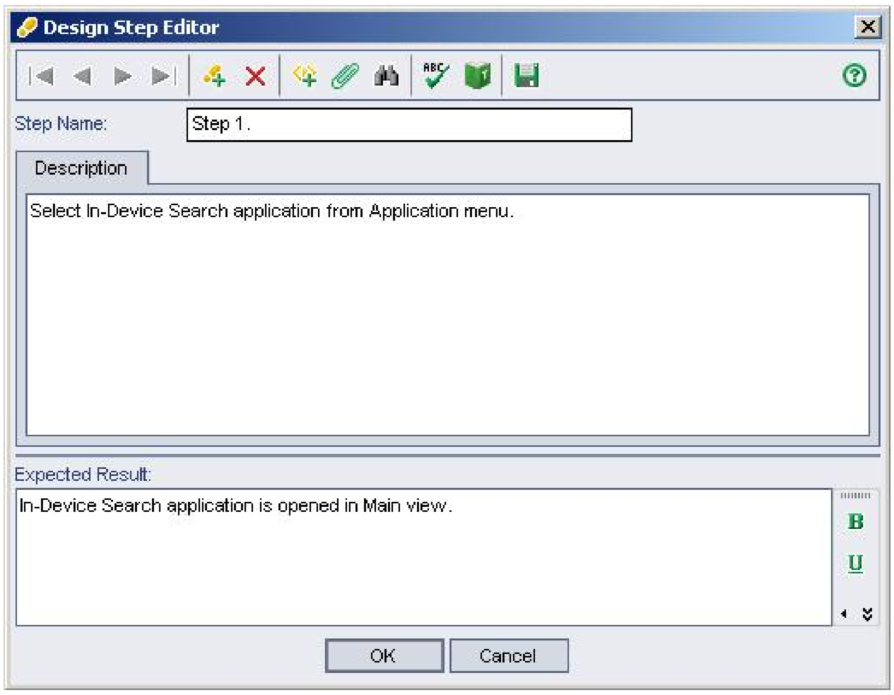
Used test environment is explained in the beginning of each test specification. Also testing tools, testing equipment and possible data needs are defined in the test specification. Usually persons responsible of testing and persons responsible of the application design are mentioned. Also possible open items are added in test specification in design phase.

First test specification is created for basic acceptance testing, which is abbreviated as BAT. BAT test cases are designed so, that all areas of the application are covered. The main idea of BAT is to test, that all areas work to some extent and fixes made after previous release have not caused any regression in other areas of the application.

In-Device Search functional test specifications were created for giving detailed instructions how to test the application. Seven functional test specifications were created for each content class, which were contacts, calendar, notes, to-dos, messages, emails and files. An additional test specification was made to test how In-Device Search functioned with memory card. This included file search from memory card and searching for messages and emails, when message storage was set to be on memory card.

There was also a separate functional test specification for testing general UI of the In-Device Search application. This test specification included test cases for checking graphical elements of the application as well as softkeys, title and options list. Textual strings, which were planned to be tested in localization testing, were also part of UI functional test specification. Usually localized tests are part of all functional test specifications, but because In-Device Search had so few localization strings, all tests for them were planned in the same test specification.

Some non-functional test cases were also created and they were included in functional test specifications. These tests had cases for testing large amounts of data, memory consumption and feature interaction with other applications.



Step Name: Step 1.

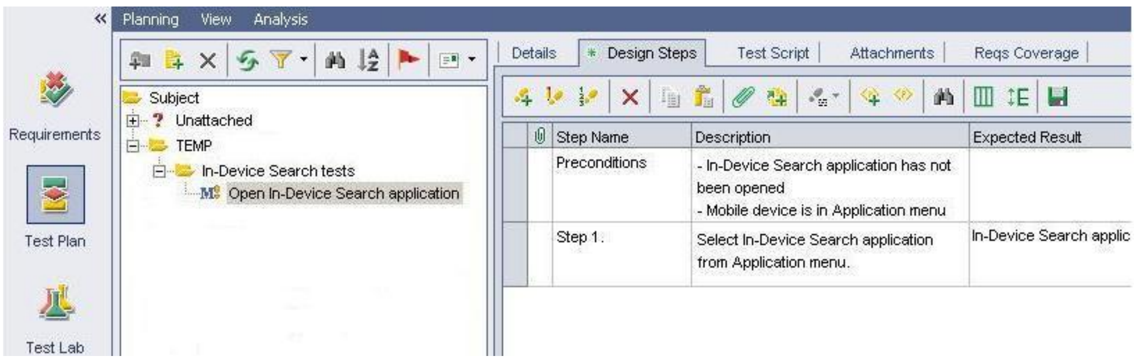
Select In-Device Search application from Application menu.

Expected Result: In-Device Search application is opened in Main view.

Figure 9: Test case design is done in QC.

Actual test cases are designed in QC. Test case design starts by defining a descriptive title for the test case. More detailed information about the test case is written in description and design steps part of the test case. First field in design steps part defines the preconditions, which must be met before test step execution can be started. Test steps are planned to give precise description of actions, which are supposed to be executed one after the other in order to complete the test case.

Figure 9 shows one test step in design step editor. The step under design is the first step after the preconditions. In test step, there is a description part, which defines the action, which tester must perform. After the test step has been performed, tester checks the actual result of the action and compares it to the expected result. If the action produces the expected results, the step is passed. Otherwise the step and the whole test case will be marked as failed.



Subject
Unattached
TEMP
In-Device Search tests
Open In-Device Search application

Step Name	Description	Expected Result
Preconditions	- In-Device Search application has not been opened - Mobile device is in Application menu	
Step 1.	Select In-Device Search application from Application menu.	In-Device Search applic

Figure 10: QC Test Plan view shows the designed test cases.

Test cases are added in a feature specific folder in QC. Figure 10 shows a highlighted test case in QC Test Plan view. Column on the left side shows, that view is in Test Plan, column in the middle shows the folder structure and column on the right show the preconditions and steps of the highlighted test case.

7.3 Selecting test data

Test data is selected to cover all searchable content in the device. Types of test data should also cover all special cases, which may arise in normal and heavy phone usage.

Contact search is performed on contact cards in Phonebook application. Test data for contact search is selected so, that there are many contacts with varying fields in the Phonebook. There is also a contact, which has all possible characters in a single data field and one contact, which have some information entered in all the different data fields.

Search can be performed on all type of calendar events, which are meeting, memo and anniversary. Test data is selected so, that lot of calendar events are created in the Calendar application. All types of events are created and they are distributed to fill many days in couple of months. Also one day is filled with many events. Searchable field in Calendar application are meeting subject, meeting location, memo subject and anniversary occasion.

Task subject can be included in search and different kinds of tasks are created as test data. Tasks with different priorities, due to dates and completion statuses are created as test data.

Text can be search from Notepad notes. Hundreds of notes with different content are created. Also a large note in size of several hundred kilobytes and a note with all possible characters are used for testing.

In-Device Search can also find different kinds of messages, which are sent, received or stored in the device. Search needs to find characters from all supported message types, which are SMS, MMS and email. Messages can be found from Messaging centre inbox, sent, drafts and user created folders.

Search criteria can be set to meet different fields of messages. Sender, receiver and message content fields can be searched. Testing is also done for mailboxes, which use different protocols in receiving mail. IMAP4 and POP3 mailboxes are tested as well as mail, which are fetched, not fetched or contain attachments. Message storage can be set to be on memory card, so searching messages from memory card is also tested.

All files, which are stored in the user area of the mobile device, can be searched by their file names. Files of all supported types and some not supported types are used in testing. Opening files from search results is not a requirement for In-Device Search application, but this is tested for checking that interaction with other application works correctly.

7.4 Testing schedules

Need for testing depends on project phases. In the early phases of the project, maturity of the application largely defines the testing need. Testing need is highest at the point, where application is about to be released into production. Software releasing schedule was defined Master Test Plan and it happened mainly in weekly or bi-weekly basis. Some unofficial releases were delivered for testing between official releases.

BAT was run for all releases. Functional tests were run for almost all bi-weekly releases and performance tests every now and then, at least once in a month. Reasons for unofficial releases was to help software developers in finding root causes of certain defects or to check that some fix didn’t cause any regression compared to any older software release.

8. TEST EXECUTION AND REPORTING THE RESULTS

After test specifications, testing tools, test environment and test data are ready, the actual testing can be started. Testing is performed according to testing schedule, which is defined in Master Test Plan. Test cases are executed as they are defined in test specifications and found errors are reported as planned.

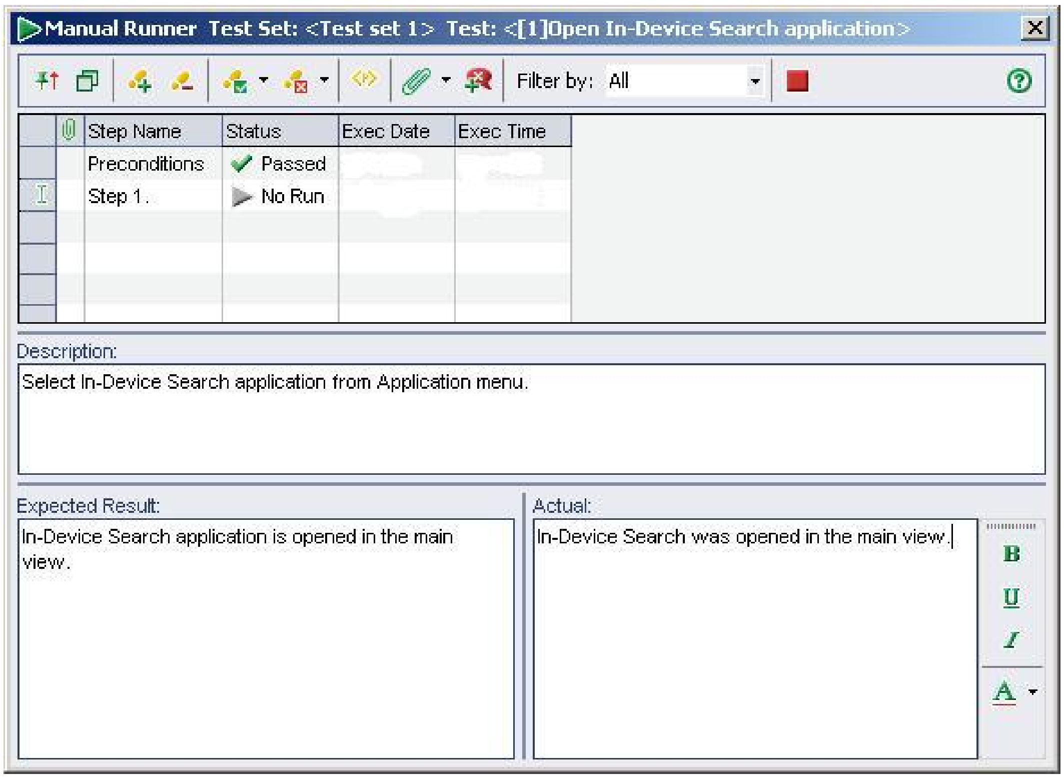
8.1 Running tests

System level tests were usually run on S60 device, which had 3.0 or later S60 version. Usually tests were run on Nokia E50 smartphone, but all supported devices were tested.

In-Device Search software releases, which were about to be tested were delivered within flash images or as Symbian installation files (sis). In both cases, the latest software version of the S60 product was first flashed on the device. If software was delivered as a sis file, it was installed on the device. Specific installation test cases were planned and their purpose was to test installation on different products and environments. These test cases also covered for example installation over existing In-Device Search application and installation on memory card.

Before the actual application testing can be started, the device must be filled with the test data. There was a separate program, which was used to fill the device with predefined test data.

After device has all the needed software and test data, the testing can be started. Testing starts by opening the correct test set in QC. Test set contains the test cases and each case defines the description, preconditions, steps and expected results for the test.


Manual Runner Test Set: <Test set 1> Test: <[1]Open In-Device Search application>
Filter by: All
B
U
I
A

Step Name	Status	Exec Date	Exec Time
Preconditions	✓ Passed		
Step 1.	No Run		

Description:
Select In-Device Search application from Application menu.

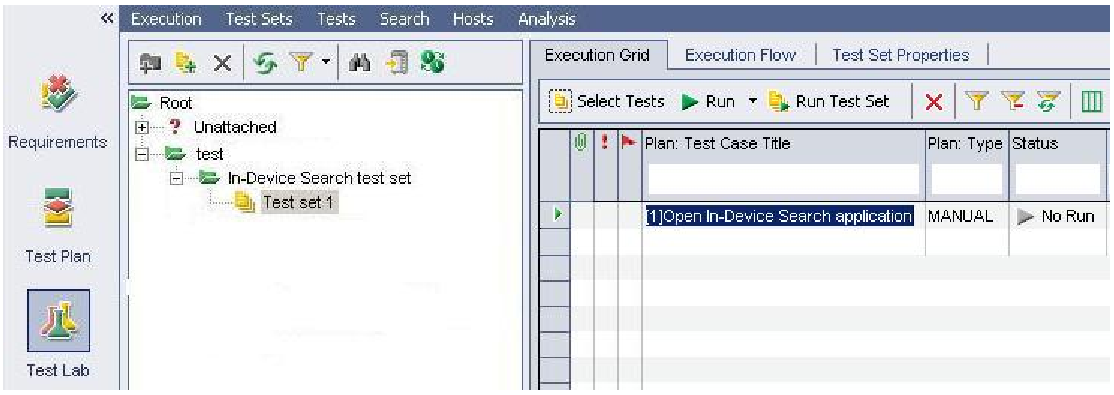
Expected Result:
In-Device Search application is opened in the main view.

Actual:
In-Device Search was opened in the main view.

Figure 11: Test case is in run in QC.

Figure 11 shows on test case, which is under execution in QC. Test case name is “Open In-Device Search application” and it has only one test step. Preconditions for running the test steps have been met, which can be seen as passed status after the preconditions field. Test description and expected results are shown and tester can type in the actual results of the action that was performed. As the actual result is the same as the expected, the test step and the whole test case can be marked as passed.

Tests are run one by one so, that the transition between the cases is as convenient as possible. This means, that the end condition of a test case should be close to the precondition of the next one. This reduces the total time of test set execution and more test cases can be added into the test set.



Plan: Test Case Title	Plan: Type	Status
[1]Open In-Device Search application	MANUAL	No Run

Figure 12: Test set is opened in QC Test Lab view.

An example test of In-Device Search tests can be seen in Figure 12. Left column shows, that view is in Test Lab, middle column shows the folder structure of test set folders and right column shows test cases in the test set. Before the test case is run, it is shown in “No run” status. Test set execution is started by pressing “Run” button and after test case is completed, the status of the run is updated into the test set view. If the actual functionality is the same as expected results, the test case status will be passed. If it doesn’t meet the expected results the status will be failed. Sometimes test case cannot be run at all and the status is marked as N/A.

Test execution round will end after all test cases in a test set are run. Error reports will be raised of the failed cases and all cases and their statuses are collected into a test results report.

8.2 Error reporting and fix verification

When error is found during testing, it will be reported into errors database. This database is used for storing all error reports with different statuses. Found error is first reported with detected status. When error is taken into further investigation, status of the error report will be set to be in progress. Software developers will investigate the problem and after the root cause for the failure is found it will be fixed. When error is fixed, status of the error report will be corrected. When the error fix is taken into some specific software release, the error status will be set to released.

After error is reported to be released, the error fix will be verified by testing team. This means, that testing engineer tests that the fix really corrects the error and no new error is raised due to the fix. If the error is fixed the error report status will be marked as verified. If error fix doesn’t correct the problem or it is not sufficient, the error will be put back to detected state.

After error fix is verified, project management can see that the error has been correctly processed and the error report will be closed.

9. CONCLUSIONS

Purpose of this thesis was to describe how system level testing was done for In-Device Search application in a software development project. Some background of a need for this kind of application was introduced. Series 60 environment was described and UI of the application was depicted in detail.

Some software testing theories were referred and test environment of In-Device Search application was described. The test automation program and automation result handling process was introduced. Test planning, test design and the actual test execution were represented in the final chapters of the thesis.

Theories of software testing were actively implemented in the In-Device Search testing project. Also test automation was used to give more information of the maturity of the system. Time needs for specific testing tasks became clearer during the project, which made resource allocation easier. Testing was performed within a planned schedule and project management received the test results and error reports in time.

To conclude, the testing of the In-Device Search application was performed as planned. Good project planning and well organized test planning and test design were the key factors in the successful completion of the In-Device Search software testing project.

10. REFERENCES

1. Symbian operating system homepage www.symbian.com

2. Series 60 homepage www.s60.com

3. S60 Platform Basics, 2006.

4. S60 UI Style Guide, 2006.

5. Mercury Quality Center http://www.mercury.com/us/products/quality-center/

6. Craig and Jaskiel, Systematic Software Testing. Artech House Publishers. USA 2002.

7. Haikala – Märijärvi, Ohjelmistotuotanto, 7th revised edition. RT Print Oy. Pieksämäki 2000.

8. Cem Kaner, James Bach, Bret Pettichord, Lessons Learned in Software Testing: A Context Driven Approach. Wiley Computer Publishing. USA 2001.

9. Automated Testing – Mercury QuickTest Professional http://www.mercury.com/us/products/quality-center/functional-testing/quicktest-professional/