

Thesis, Åland University of Applied Sciences, Degree Programme in
Information Technology

INVESTIGATION OF THE SOFTWARE PROFOUND UI FOR INTERFACE MODERNIZATION

Isak Jansson



39:2016

Publishing date: 08.12.2016
Supervisor: Agneta Eriksson-Granskog

DEGREE THESIS

Åland University of Applied Sciences

Study program:	Information Technology
Author:	Isak Jansson
Title:	Investigation of the Software Profound UI for Interface Modernization
Academic Supervisor:	Agneta Eriksson-Granskog
Technical Supervisor:	Crosskey Banking Solutions - Gunnar Löfström

Abstract
<p>The purpose of this thesis is to investigate whether the software for interface modernization Profound UI can be integrated and used for modernizing Crosskey's existing back-end system. Profound UI is a tool for interface modernizing of IBM i applications. The investigation will be focused on the modules Genie, Visual Designer, DDS-Conversion and RPG Open Access Handler of the Profound UI Suite.</p> <p>A number of test programs and proof of concepts have been made. By using the Profound UI to develop and convert existing applications the product and the results are evaluated. Additionally a brief explanation of the programming languages, techniques and the different programs are presented.</p>

Keywords
Profound Logic, Profound UI, Genie, Visual Designer, DDS-Conversion, Open Access Handler, IBM i, RPGLE, JavaScript, CSS, Web Application, Interface modernization

Serial number:	ISSN:	Language:	Number of pages:
39:2016	1458-1531	English	43 pages

Handed in:	Date of presentation:	Approved on:
08.12.2016	02.12.2016	08.12.2016

EXAMENSARBETE

Högskolan på Åland

Utbildningsprogram:	Informationsteknik
Författare:	Isak Jansson
Arbetets namn:	Undersökning av programvaran Profound UI för gränssnittsmodernisering
Handledare:	Agneta Eriksson-Granskog
Uppdragsgivare:	Crosskey Banking Solutions - Gunnar Löfström

Abstrakt

Syftet med denna avhandling är att undersöka om programvaran för gränssnittsmodernisering Profound UI kan integreras och användas för att modernisera Crosskeys befintliga backend-system. Profound UI är ett verktyg för gränssnittsmodernisering för IBM i produkter. Undersökningen fokuserar på modulerna Genie, Visual Designer, DDS-Conversion och RPG Open Access Handler ur Profound UI Suite.

Ett antal testprogram och konceptprogram görs genom att använda Profound UI för att utveckla och konvertera befintliga applikationer. Produkten och resultaten utvärderas dessutom. En kort förklaring av de programmeringsspråk och tekniker samt olika program som används ingår i arbetet.

Nyckelord (sökord)

Profound Logic, Profound UI, Genie, Visual Designer, DDS-Conversion, Open Access Handler, IBM i, RPGLE, JavaScript, CSS, webbapplikation, gränssnitt, modernisering

Högskolans serienummer:	ISSN:	Språk:	Sidantal:
39:2016	1458-1531	Engelska	43 sidor

Inlämningsdatum:	Presentationsdatum:	Datum för godkännande:
08.12.2016	02.12.2016	08.12.2016

TABLE OF CONTENTS

1. INTRODUCTION	5
1.1 Purpose	5
1.2 Methods	5
1.3 Limitations	6
2. BACKGROUND	7
2.1 Crosskey	7
2.2 Why change the system?	7
2.3 Languages and Frameworks	8
2.3.1 Hyper Text Markup Language	8
2.3.2 JavaScript	8
2.3.3 CSS	9
2.3.4 JSON	10
2.3.5 RPGLE	10
2.3.6 Display File DDS-source	11
2.3.7 IBM i	12
2.3.8 RDI	12
2.3.9 IFS	12
2.3.10 IBM i Access for windows	13
3. THE SOFTWARE	14
3.1 Profound Logic	14
3.2 Profound UI	14
3.3 The different modules	15
3.3.1 Genie	15
3.3.2 Visual Designer	15
3.3.3 DDS-Conversion	15
3.3.4 RPG Open Access Handler	16
3.3.5 Atrium	17
3.3.6 Jumpstart	17
3.3.7 Profound UI Mobile	18
4. SOFTWARE INVESTIGATION	19
4.1 Genie	19
4.1.1 Genie application	19
4.1.2 The Genie process	22
4.1.3 Development of the Skin	22
4.1.4 Design mode	23
4.1.5 Limitations of Genie	24

4.2 Visual Designer and DDS-Conversion	25
4.2.1 Old screens	26
4.2.2 Old program	26
4.2.3 Development	27
4.2.4 New screen	28
4.2.5 Changed Program	30
4.2.6 Running the new screen	30
5. SPECIFIC AREAS	32
5.1 Theme code	32
5.2 Skin code	33
5.3 CSS	34
5.4 Problems	36
5.4.1 Genie	36
5.4.2 Visual Designer and DDS-Conversion	36
5.4.3 Contact with Profound	37
5.5 Overall Security and Login	37
5.5.1 Security and underlying structure of Genie	38
6. CONCLUSION	39
6.1 Genie evaluation	39
6.2 Visual Designer evaluation	39
6.3 DDS-Conversion evaluation	40
6.4 User reviews	40
6.5 Overall evaluation	41
6.6 Final words	41
REFERENCES	42

1. INTRODUCTION

1.1 Purpose

The purpose of this thesis is to investigate whether the software for interface modernization Profound UI can be integrated and used for modernizing Crosskeys existing back-end systems consisting of IBM i servers with IBM i Access emulators.

An interface modernization of the current system is under investigation and a solution for the modernization is investigated. To avoid large costs and labour intensive work, different solutions to modernize the system are investigated. Crosskey have selected a product that could be of interest and need an investigation if the product in question is possible to integrate into existing systems.

Profound UI is a tool for interface modernizing IBM i applications, and should ease the changes needed to move away from IBM i Access 5250 emulators. Profound UI moves the IBM i applications from the emulators to the web and should give a more user friendly experience of the system, without changing the underlying applications. (Profound Logic Software, 2016a)

1.2 Methods

Studies in documentation of the software have been used for understanding the Profound UI product and developing of the applications. Documentation and discussions at Crosskey gave information about the current system and its use.

In the investigation of the Profound UI product, a number of test programs and concept programs have been made. By using the Profound UI to develop and convert existing applications the product and the results have been evaluated. By comparing old and new applications the usability and consistency was tested, this was done by a few product

specialists and the developer. The thesis concludes if the software Profound UI is suited for the demand from Crosskey.

1.3 Limitations

By only examining certain parts of the software package, work will be limited to those areas. Crosskey selected three parts of the software to be tested. With focus on Profound UI's software Genie, Visual Designer and DDS-Conversion the product will be evaluated. The other parts of the software will be shortly mentioned but this thesis will not go through them in detail. The installation of the software is already done and will not be brought up in the text.

A simple test program and a proof of concept (concept program) will be the main elements that are presented. Additionally a brief explanation of the programming languages, techniques and the different programs are going to be presented.

2. BACKGROUND

2.1 Crosskey

Crosskey Banking Solutions (CBS) is an IT company that delivers banking systems for banks in the Nordic and Baltic regions. CBS was founded in 2004 by separating the IT system of Ålandsbanken and developing to a separate business. Today CBS is a subsidiary to Ålandsbanken and has around 200 employees. (Crosskey, 2016)

CBS operates in the traditional banking areas as well as eBanking, Card, Mobile Payments and Capital markets. The company tries to operate dynamically with modern and flexible solutions with a personal touch and let the customers select precisely which functionality they need. (Crosskey, 2016)

2.2 Why change the system?

The back-end users from the banking areas are using the IBM i Access 5250 emulators to get access to the resources of the IBM i. The User Interface or UI is outdated and is in need of modernization. The typical reaction of a new client that sees the IBM i Access 5250 emulators is that it is old and outdated. The system works well and is robust but the look and feel could be improved. Comparing the emulator to new applications the UI is in need of modernization.

The modernization of the system is worked on with a more modern UI written in Java but the process is taking long time. The customer must now open two applications to access all the functionality, giving a clumsy way of working. To speed up the process of modernization the software product Profound UI is investigated as a solution.

2.3 Languages and Frameworks

2.3.1 Hyper Text Markup Language

Hyper Text Markup Language (HTML) is a language for text formatting of documents on the Internet (World Wide Web). Web pages are generally written in HTML and contain the structure, metadata and to some extent the design of the webpage. HTML allows inserting other types of information such as other files, applications, design instructions and links to other websites.

HTML documents consist of tags enclosing elements on the page. Usually there is a start and end tag for the elements. Attributes in the tags can give different characteristics of that particular element. The content of the tags is written in between the start tag and the end tag.

The page consists typically of an HTML tag, within the tag is a head tag that contains the non-visual information of the page such as metadata. The tag also contains a body tag that defines the content of the page. HTML is used in combination with other languages to get more functionality, usability and design, for example JavaScript and CSS. (W3schools, 2016)

2.3.2 JavaScript

JavaScript (JS) is a scripting language used for web pages. JavaScript enables the web pages to react to input without having to communicate with the server side application. JavaScript is an object oriented scripting language which is small and lightweight for the web. Linking objects with JavaScript gives programmatically control over the objects.

JavaScript have a standard library of most common functionality, but functionality can be coded in separated APIs (Application Programming Interface) for the web application. Client-side scripting provides control over the web browse for example responding to input from the page without refreshing the web page.

JavaScript is similar to Java, but they have major differences. JavaScript is a scripting language used primarily in client-side scripting, while Java is used for applications outside the web browser. (Mozilla Developer Network, 2016b)

2.3.3 CSS

Cascading Style Sheet (CSS) is a style sheet language that is used to describe the presentation of HTML documents or other documents written in markup languages. The HTML page contains tags and the content of the tags can be designed by using CSS to define fonts, color, layouts and more. CSS is designed to separate the document content and document presentation from the HTML pages.

One CSS file can be linked to many HTML pages and define the design of all HTML pages as long as the same structure is used across the different pages. It is easy to modify the CSS file and the effect is shown on all linked HTML pages. This speeds up the loading speed for the webpage, because the CSS file only needs to be loaded once.

The CSS file consists of different selectors that contain the design elements of the corresponding HTML tag. The selectors can be simple, class (.class) or id (#id) selector. The CSS file can contain different selectors for different screen sizes and vary for the same HTML elements without needing to change anything in the HTML.

The CSS describes a property scheme to determine which property that will be applied to which element, if the property matches many elements priorities are calculated and the right rule is assigned to the property. Meaning previous design rules are overridden. World Wide Web Consortium (W3C) maintains the CSS specifications and operates a validation of CSS documents. (Mozilla Developer Network, 2016a)

2.3.4 JSON

JavaScript Object Notation (JSON) is syntax for describing objects in text format that is easy to parse and generate. It is a subset of JavaScript syntax but is not the same, JSON is only a data-interchange format, that helps serializing and communicating data between website JavaScript and server side applications.

JSON objects are a set of values with a name, each name is followed by a colon and then the value, each pair of name:value is separated by comma. An object can contain an array of values, each value separated by a comma. In this way a JSON object can contain a large amount of data while keeping it structured and easy to read and parse. The similarities between JavaScript objects and JSON objects are many, making the conversion of JSON data to JavaScript objects easy. (Mozilla Developer Network, 2016)

2.3.5 RPGLE

RPGLE (RPG or RPG IV) is a programming language used for business applications on IBM i. It started as a Reporter Program Generator in 1959 to replicate punch card possessing originating from Cobol, but it has evolved to a more modern programming language by changing over the years. In 2001 RPG IV free-format coding was released and by 2013 full free format coding was introduced breaking the ties to punched cards.

RPG programs are often started by a control specification (H) that describes the control changes, then listing all files written, read or updated in a File specification (F) and then followed by a Data definition specification (D) that contains all variable, data structures and arrays. The Calculation specification (C) contains the executable code where all calculations and data handling takes place. Example code can be found in Figure 1 . The file can also contain a Procedure specification that defines procedures in the program (sub procedures). The procedures can be local sub procedures like functions or external procedures from other programs.

RPGLE is a popular programming language in the IBM i operating system. (IBM Corporation, b)

```
H DFTACTGRP(*NO)
FFILENAME CF WORKSTN infds(dspinf)
D text S 50
/free
if true;
text = 'Test text';
endif;
/end-free
```

Figure 1. Example of RPGLE code in free format.

2.3.6 Display File DDS-source

Data Description Specifications (DDS) are a way to describe data attributes in a file description. IBM i applications uses the description to show data and the application is external from the description.

A DDS-source can be different types of files. The Display file DDS is a file that specifies the design and output of a display screen in IBM i Access 5250 emulator environment. The file contains specifications of the Display file. The left side defines the common attributes like record formats, fields, names and length of the fields. The right side of the file is for DDS keywords. DDS keywords is defining the Display file look or defines a function that gives data. This format is a character based protocol for the presenting of applications on the IBM i Access 5250 emulators which is used for interaction against IBM i applications. (IBM Corporation, a)

2.3.7 IBM i

IBM i is an operating system run on IBM power systems, an evolution of previous named i5/OS and originally AS/400. The system is designed to be low maintenance when up and running and require no attention during normal operations. The system was the very first to be object based and to have a system for persistent objects.

IBM i has a built in DB2 system (database) that does not need to be installed separately. The DB2 system is integrated and therefore gives a unique performance and value to the system. The IBM DB2 has scalability and is a easy-to-secure environment for business applications. The main programming language used today is RPGLE for the applications on the IBM i. (IBM Corporation, 2016b)

2.3.8 RDI

Rational Developer for i (RDI) is the primary development environment for the IBM i systems. The RDI is an integrated development environment based on the Eclipse platform. IBM offers different plugins for the RDI, with the plugins the developer can edit the code, compile programs, browse the IFS, visual design Display files, access members, access objects and much more related to IBM i. All of these components gives the developer possibility to customize the development platform for more effective development. (IBM Corporation, 2016c)

2.3.9 IFS

Integrated File System (IFS) is an interface where users can access different types of files. This system is used in IBM i programs to access files. The system offers stream input, output and storage in a similar way as personal computers. The IFS is a part of the IBM i operating system. (Scott Klement, 2015)

2.3.10 IBM i Access for windows

IBM i Access terminal emulator package for 5250 screen applications gives the user access to IBM i. The package provides access to data and applications of the IBM i system. Through the emulator the user uses applications that is made of Display files that uses underlying RPG programs for the business logic. The Display file DDS-sources, contain the design information about the applications. Information is fixed size and only text based. The terminal emulator that emulates 5250 screens is also called Green-Screen applications. The product have had many names during the years but is now called IBM i Access emulator. The IBM i Access emulator is shown in Figure 2. (IBM Corporation, 2016a)

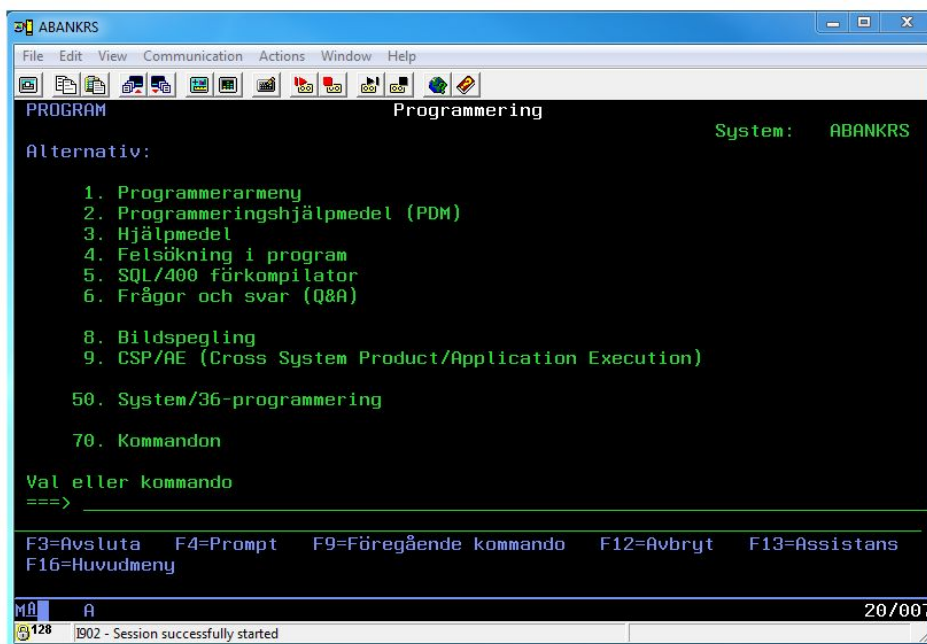


Figure 2. IBM i Access emulator.

3. THE SOFTWARE

3.1 Profound Logic

Profound logic software is a provider of modernization and development solutions for the IBM i platform. It was founded in 1999 and have provided software solutions for the IBM i platform since, with the first Visual Designer for RPG applications and the most mature RPG Open Access Handler. Their mission is to stay dedicated to the IBM i platform but ensure reaching the modernization goals for the customer. (Profound Logic, 2016)

The rest of the thesis information is supplied by the Profound Logic web pages and documents (Profound Logic, 2016) (Profound Logic Software, 2016b).

3.2 Profound UI

Profound UI is a modernization Suite for IBM i applications, that offers different solutions for modernizing RPG applications without changing the business logic or functionality. Profound UI offers a set of tools to transform existing applications into richer user interfaces still run on IBM i. The underlying application do not need to change to get the benefit of bringing the IBM i to the web.

Using web development like HTML, CSS and JavaScript the application gets a more modern feel and use. The benefit of using Profound UI is that no extensive knowledge of new programming languages is needed to be able to develop applications. As well as not changing from RPGLE or IBM i. With knowledge of new techniques own solutions can be coded and integrated into the Profound UI platform.

Profound UI offers a graphical interface with an easy way to transform old applications or develop new applications that still run on IBM i. In the Profound UI Suite there are many different modules, but in this thesis only a few of them will be extensively explained, but to give an overlook of the product a brief explanation of the different modules are made.

3.3 The different modules

3.3.1 Genie

Genie Converts traditional 5250 emulator screens on the fly and distributes them onto the web. Enabling screen emulation from the IBM i on the fly to a web application. The emulation of the application is done by different conversion rules that is specified in a so called Skin. Genie allows customization and changes to the screens by offering different Skins that can be modified and an online Designer mode that can design individual screens after the conversion.

3.3.2 Visual Designer

Visual Designer is a design tool that allows for new development of Rich Display files with a graphical interface. Designing web enabled screens without needing to learn extensively about web languages. The Visual Designer is a browser interface where designing screen applications are simply done by point-and-click, drag-and-drop manoeuvres. An RPG programmer can in this way easily create new screen applications without learning about new web languages.

3.3.3 DDS-Conversion

DDS-Conversion tool included in the Profound UI Suite, is a process that converts old Display file DDS-source files to new Rich Display file DDS-sources. The automated process gives the possibility to convert the old applications to fit the new web enabled platform and the screens can then be maintained through the Profound UI products.

The DDS-Conversion tool is integrated in the Visual Designer to give a direct view of the converted screen and give the possibility to modify and improve the original screens. The design is saved in Rich Display files DDS-sources that then is used to show the screen on a webpage. The process removes all ties to the old 5250 character-based protocol for presenting applications.

3.3.4 RPG Open Access Handler

Rational Open Access Handler is an Open Access Handler for opening up access for the RPG programs to other resources not only calling the operating system. It uses the RPG I/O model to access resources that are not directly supported by RPG. This allows writing a I/O handler to access other devices and resources such as browser, web services and much more.

The handler allows the RPG program to use standard code to access outside devices though a handle that takes care of the I/O operations for the program. In this way the RPG code can be used as previous but the I/O services can change.

Profound UI Handler for RPG Open Access provides a handler that uses its Web Services that then output the code to a website, without changing the RPG code, the changed process is shown in Figure 3. The RPG programs that operate with Display files are automatically understood and the output is rendered in a web browser. This gives the solution of using native IBM i for modernizing the interface of the applications without changing the RPG code. (IBM Corporation, 2010)

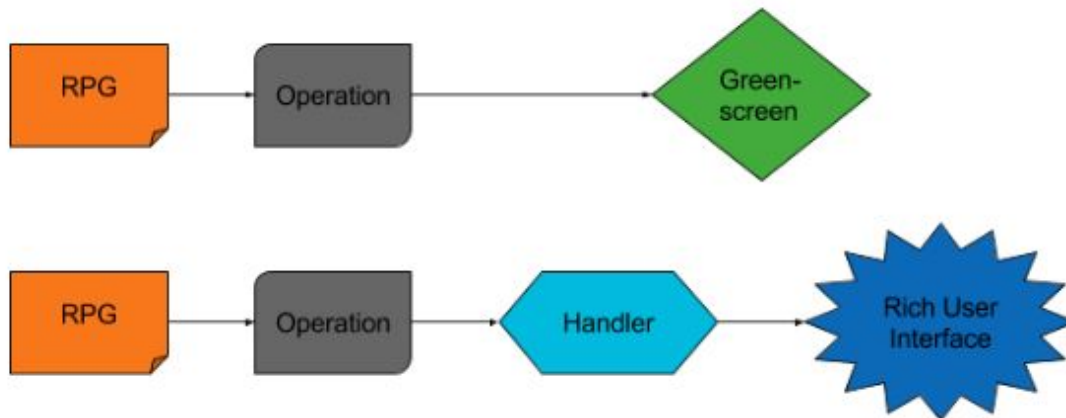


Figure 3. Old I/O operations vs the new Open Access Handler.

3.3.5 Atrium

Atrium is a navigation system that offers a way of organizing menu options and launching different applications. It ties the applications securely to the IBM i system, using menus in a web browser environment. Atrium can eliminate multi-level options screens with a menu that gives a direct path to the applications.

There are many different menu systems to choose from to link the applications. In Atrium it is possible to manage user groups and users to use special customized menus and screens, giving a granular and secure separation for users. This solution requires that all the screens and menus are converted to Rich Display format.

3.3.6 Jumpstart

JumpStart is a code generator for RPG programs that automatically generates user interfaces and application logic. Jumpstart enables developing without starting from scratch. Providing templates that give a program design for specific needs. With a template and some simple options a fully functioning programs and interfaces can automatically be created. It can create subfiles, data entry and lookup applications.

3.3.7 Profound UI Mobile

Profound Mobile is a mobile development tool that uses the other components of the Profound UI products to make IBM i available for mobile devices. Profound Mobile allows development of mobile products to add native capabilities on any mobile device and connect directly with native IBM i applications. Using integration with PhoneGap/Apache Cordova.

4. SOFTWARE INVESTIGATION

4.1 Genie

The investigation and proof of concept of the Genie software have been focusing on the development and possibilities of the Genie Skin, for customization of the emulation of the 5250 sessions. The emulation and conversion rules are specified in a so called Skin. By looking on how the Genie application work and developing a Skin for the Genie application, the product was evaluated.

4.1.1 Genie application

Genie is the emulator equivalent of the IBM i Access 5250 emulator that is in use today. Genie moves the emulator to the web and can be accessed from any web browser capable of running modern JavaScript, an example is shown in Figure 4. The Genie application analyses and converts the 5250 stream of the IBM i. This means the underlying code does not change. When analysing the stream the application sends the data with help of JSON to the web application that builds the screens as specified in the Skin. In Genie the concept of defining screen design and conversion is called Skin.

The Skin handles the data and applies settings and code for the look and function of the application. There are many different Skins available from the start and the Skins can be customized for different conversion rules and looks. By customizing a Skin the web application can change design as wanted by the customer. The Skins can easily be switched between and in this way change the emulation behaviour.

Genie has an administrator page that is used for creating new Skins and copying existing Skins as well as for editing the Skins. If a more granular conversion or detecting of special property is wanted, modification of the predefined settings or JavaScript code in the Skin can be coded.

Genie has a Designer mode when running the application, which can be turned on in the administration page. The Designer mode is explained further in the chapter 4.1.4 Design mode.

The Skins have configuration settings for easy configuration of how a Skin should work and what a Skin need to detect when converting. The setting file is a JavaScript file that should only contain predefined properties as the documentation describes. By adding a property and giving it a value the settings will change. This can be done in the administrator page or by coding in the setting file.

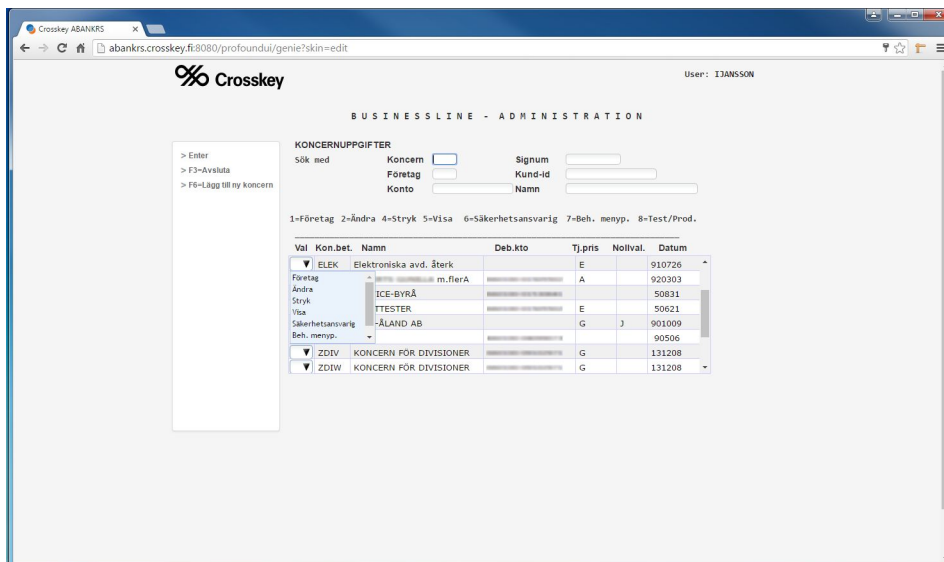
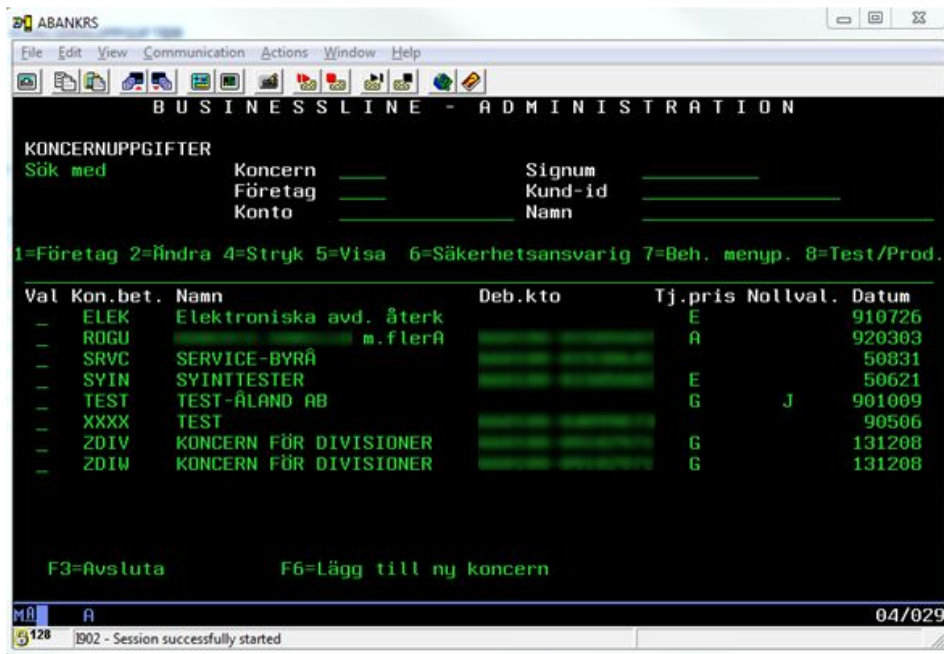


Figure 4. IBM i Access emulator vs Genie.

4.1.2 The Genie process

When Running Genie the emulator process is handled by JavaScript that parses and builds the web page in JSON data format. JavaScript builds the web page, the CSS handles the design and the HTML connects all files. The data stream from the IBM i servers is parsed, the data is put in JSON format, the data is sent to the web application and JavaScript runs the page. The parsing is done by underlying programs that recognizes the different information and data fields in the 5250 data stream. All the requests and interactions are handled by JavaScript that sends the requests to an underlying program and then to the RPG application.

The process of building the web page is handled by JavaScript. A part of this script can be modified by using a so called custom JavaScript file in the Skin. In the custom file the JavaScript builds the screens by looking for properties and giving them a different design or value. This file can be used for special customization of properties, changing the look of the emulation. JavaScript functions can be written for example to let button actions change a property on the screen.

4.1.3 Development of the Skin

The development of the Skin mainly focused on changing the JavaScript to find and handle all necessary properties, as well as coding the CSS for the look of the Skin. Looking at different Skins, a good example Skin was selected as a basis for the customization. The selected Skin was modified for fitting the specifications. The JavaScript was changed and the CSS was rewritten, for example adding the company logo to the screen and much more.

Small problems with the Skin not converting or detecting a property correct were repaired by the developer, or using the Designer mode for repairing problems that could not be repaired through the coding of the JavaScript or CSS.

The code of the Skin is further explained in the section 5.2 Skin code.

4.1.4 Design mode

In Genie there is a mode called Design mode that can be enabled from the administration page configuration settings. The Design mode enables the user or administrator to design a specific screen in the emulation process. When the Design mode is enabled a floating toolbar will appear on the screen as shown in Figure 5. The user can use the different property tools to design a screen. This is done by drag and drop tools or editing the properties in the screen tool window.

Designing the screen in the Designer mode, a specific screen identifier that is unique for that screen must be chosen before the design can be saved. Using the Skin and returning to the screen the changed elements will appear. The changed elements are stored in a scn-file with the changed and modified properties in JSON format. On the administration page the changed screen's names and identifiers can be seen.

Returning to the designed screen Genie finds the identifier, reads the file and applies the changed properties to the screen. The Design mode makes no changes to the original source code, meaning changing the screens on the fly.

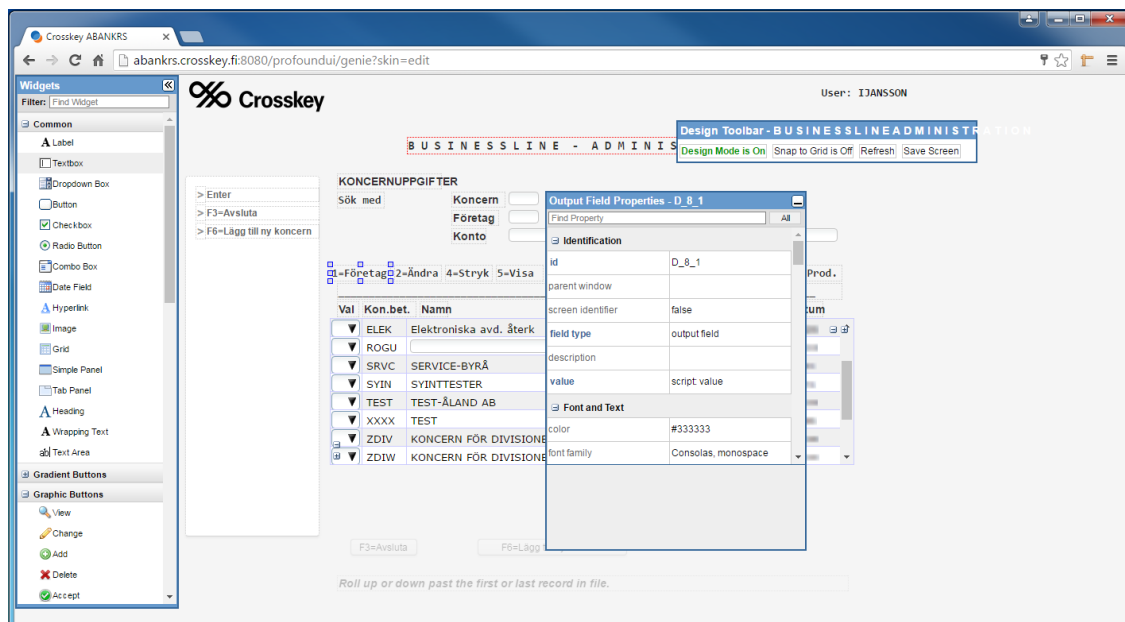


Figure 5. The Design mode enabled, showing the floating tool bars.

4.1.5 Limitations of Genie

Small design imperfections in the screens can be resolved within a few minutes easily by changing the properties in the Design mode. If the screen needs many changes it will create a large design file (scn-file) and which take time. When making large design changes and requiring functionality changes the Visual Designer and the DDS-Conversion could be a better option. Note that the Genie application can't change code and you are limited to the already existing functionality of the application. The screen designed in the Visual Designer can still be used through Genie after some settings changes.

4.2 Visual Designer and DDS-Conversion

In this section a comparison between the old IBM i Access 5250 emulator and using the Visual Designer to show the RPG applications on the web is done.

The Visual Designer application is a design tool for developing new applications for the web using RPG programs in the background. The Visual Designer saves the screens into Rich Display files that contains the design information for the application. By using the DDS-Conversion tool in the Visual Designer, old 5250 screens can be converted to the new Rich Display format and be displayed as a web application. The difference in the process is shown in Figure 6. This technique uses the Open Access Handler for enabling I/O operations from the RPG programs to the web applications.

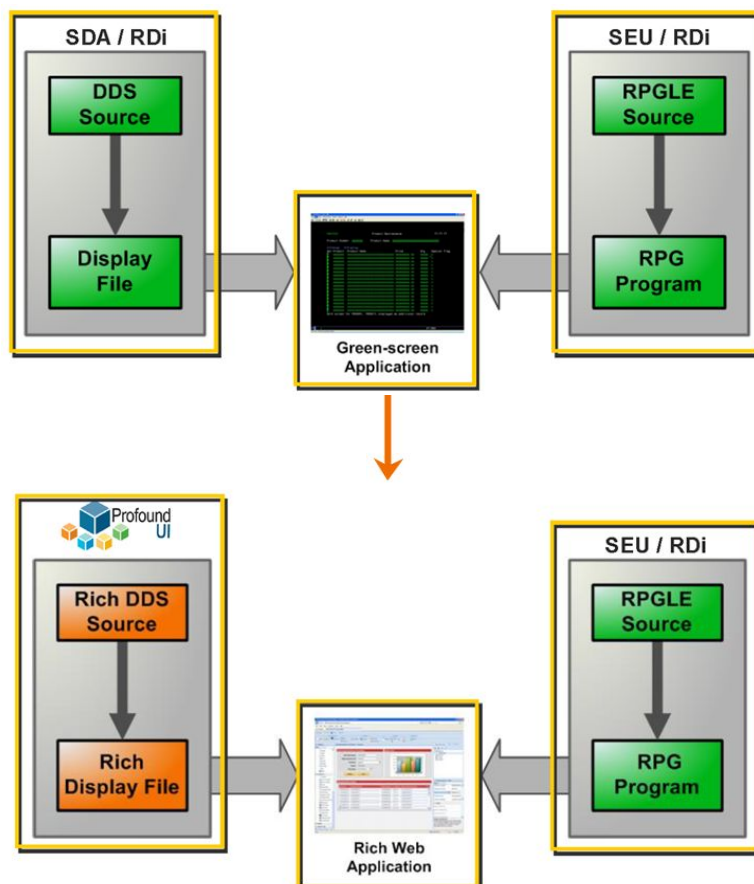


Figure 6. The change from using Display format to Rich Display format and the Open Access Handler for Web application. (Alex Roytman & Profound Logic, 2015)

4.2.1 Old screens

The old screens are written in Display file format DDS-sources. The development of the screens are often done in the RDI. The screens contain different input areas and record formats that have definitions for how they look and how they behave. A brief explanation of this was done under the explanation of the 2.3.6 Display file DDS-source section in the beginning of the thesis.

The DDS-sources basically function as a template for the RPG program. The DDS-sources is where all fields and text are defined. All properties are position defined and fixed to a specific position on the screen. An example of a field is an input field where the user can put in data and the RPG program processes the data. The keywords specify different behaviours for the fields, for example message constants. The message constants come from a message library containing text constants for different languages.

4.2.2 Old program

The old RPG program is the handler for the Display file, handling all the output and input as well as processing the data. The RPG file contains a file specification where a Display file is specified for out and input processes. The RPG program fills the Display file with data and shows the data by calling the record format from the program using a write command. All the information that is set in the Display file variables are pushed to the Display file and shown to the user. The user sees the information and puts in the data. The program handles the input. This is often done in a loop that contains the program logic for the Display file distributing the data to other programs or services.

4.2.3 Development

In the Visual Designer and DDS-Conversion the development was mainly focusing in creating a Theme for the conversion that would fit the company specifications. The conversion Theme is a script for telling how the conversion of old Display files is done and how the converted screens should look. The Theme contains rules for the DDS-Conversion.

The Theme is written in JavaScript, mainly in JavaScript objects that handle different properties during the conversion process. By defining the properties of the conversion, the program handles the different screen properties and layout differently. More about the code of the Theme is covered in 5.1 Theme code section.

The process of converting after the Theme is written is simply a matter of choosing convert and check the conversion results that are displayed in the end of the conversion. The conversion results contain information about the conversion and information of possible errors or unhandled properties. By looking at the result the developer can correct or know what needs to be corrected for the screen to function properly.

After the conversion the result will appear in the Visual Designer for editing possible problems and continue designing the screen. By designing the screen even more after conversion a number of improvements to both functionality and design can be added. By simply dragging and dropping widgets, editing properties and even adding JavaScript to the screen.

To get full use of the web interface that the Rich Display files are shown in, a change in program logic is often needed. Today the layout of the menus and screens only uses the limited area of the 5250 emulator. In the web interface the capabilities of the program are increased, meaning a change in the RPG program can give a better use of the web interface.

4.2.4 New screen

The new Rich Display format files are still stored in DDS-sources but containing different data from simple Display format files. Still containing the input fields defined as the old screens and the same definitions of the record formats. The difference are the keyword definitions. They are replaced by HTML tags instead. These HTML tags contain data about the different output fields or JSON data that define the layout and structure of the page. For every new record format there are a new input definition and an output definition as well as several HTML keywords with JSON data. The DDS-source is still the same but contains different data structures and code, the compilation process remains as of before. The difference between old Display format and Rich Display format in DDS-sources are shown in Figure 7.

```

A      MARKÖR      3S 0H      SFLRCDNBR
A      BANKNAMN    30A      1 2DSPATR(HI)
A                                     1 33MSGCON(030 XXXXXXX XXXX)
A                                     DSPATR(HI)
A                                     1 70DATE(*JOB *YY)
A                                     EDTCDE(Y)
A                                     DSPATR(HI)
A      JOBAID      10A      2 2DSPATR(HI)
A                                     2 31MSGCON(030 XXXXXXX XXXX)
A                                     DSPATR(HI)
A                                     2 72TIME
A                                     DSPATR(HI)
A                                     4 3MSGCON(047 XXXXXXX XXXX)
A      S1SETL      3A  B  5 5DSPATR(UL)
A      R  BCMD

```



```

A      BANKNAMN    30A  H
A      FIELD0001  10A  H
A      JOBAID      10A  H
A      MARKÖR      3S 0H
A      MSGCON1     30A  H
A      MSGCON2     30A  H
A      MSGCON3     47A  H
A      MSGCON4     14A  H
A      MSGCON5     25A  H
A      S1SETL      3A  H
A      R  BCMD

```

```

1 2HTML('QPUIREC4      ROVERLAY  0      -
RROWS      2      C2      21C2      22 FM-
SGCON4      MSGID      5      C1      0C0 -
C7      XXXXXXXXXX      XXXXC0      FMS-
GCON5      MSGID      5      C1      0C0 -
C7      XXXXXXXXXX      XXXXC0      ')
1 2HTML({'screen':{'record format name": "BCMD", "overlay": "true", "overlay-
range": "21-22"}, "items": [{"id": "XX-
XXXXXX_XXXX_HeadingPanel", "field ty-
pe": "output field", "left": "5px", "to-
p": "0px", "width": "940px", "height": "-
44px", "z index": "8", "locked in plac-
e": "true", "css class": "hybrid-headin-
gC", "css class 2": "stationary"}, {"-
id": "XXXXXXXX_XXXX_ActionsPanel", "f-
ield type": "output field", "left": "5-
px", "top": "52px", "height": "454px", "-
width": "165px", "z index": "8", "locke-
d in place": "true", "css class": " ac-
tions-panel", "css class 2": "station-

```

Figure 7. Example of old Display format and Rich Display format in DDS-source.

4.2.5 Changed Program

The changes to the RPG program aren't big if only a simple conversion is done without further development. The only thing that should change is adding the Profound UI Open Access Handler to the program and set the right activation group if it's not previously set. By adding the Profound UI Handler in the file specification the I/O operations go through the Open Access layer and the new Open Access Handler takes care of the operations, as shown in Figure 8. This replaces the old write calls with a new functionality. Giving the program possibility to access other forms of I/O operations than the old Display file or file operations of IBM i. The Open Access Handler is designed to replace the I/O calls and give a more versatile use of the operations, in this case communicating with the web application.

As said before in the 4.2.3 Development section, the more use of the web interface can be obtained by changing the program logic. When changing the program further to adapt to the web interface the only specific thing to add to the program is the Open Access Handler. The other changes are done as programming an RPG program in the old fashion way.

```
FFILENAME CF  E          WORKSTN HANDLER('PROFOUNDUI(HANDLER)')
F                                     sfile(bs01:rrÄ)
```

Figure 8. The RPG Open Access Handler in RPG code.

4.2.6 Running the new screen

Running the new screen is done by simply using a web browser that can handle JavaScript and a link to the program that will be run. The link starts the page and shows the program running in a web interface. In most cases the program that is called needs the user to have the right library list, the right authority and the right in parameters. This means that there is a need to create a start program that gives the user the right library list, the right authority and in parameters. Further the start program should take into account that it must be able to call different applications in different libraries. When the program is called everything will

function as normally as in the IBM i Access emulator but with enhanced design, usability and on the web, without using an emulator. This is shown in Figure 9.

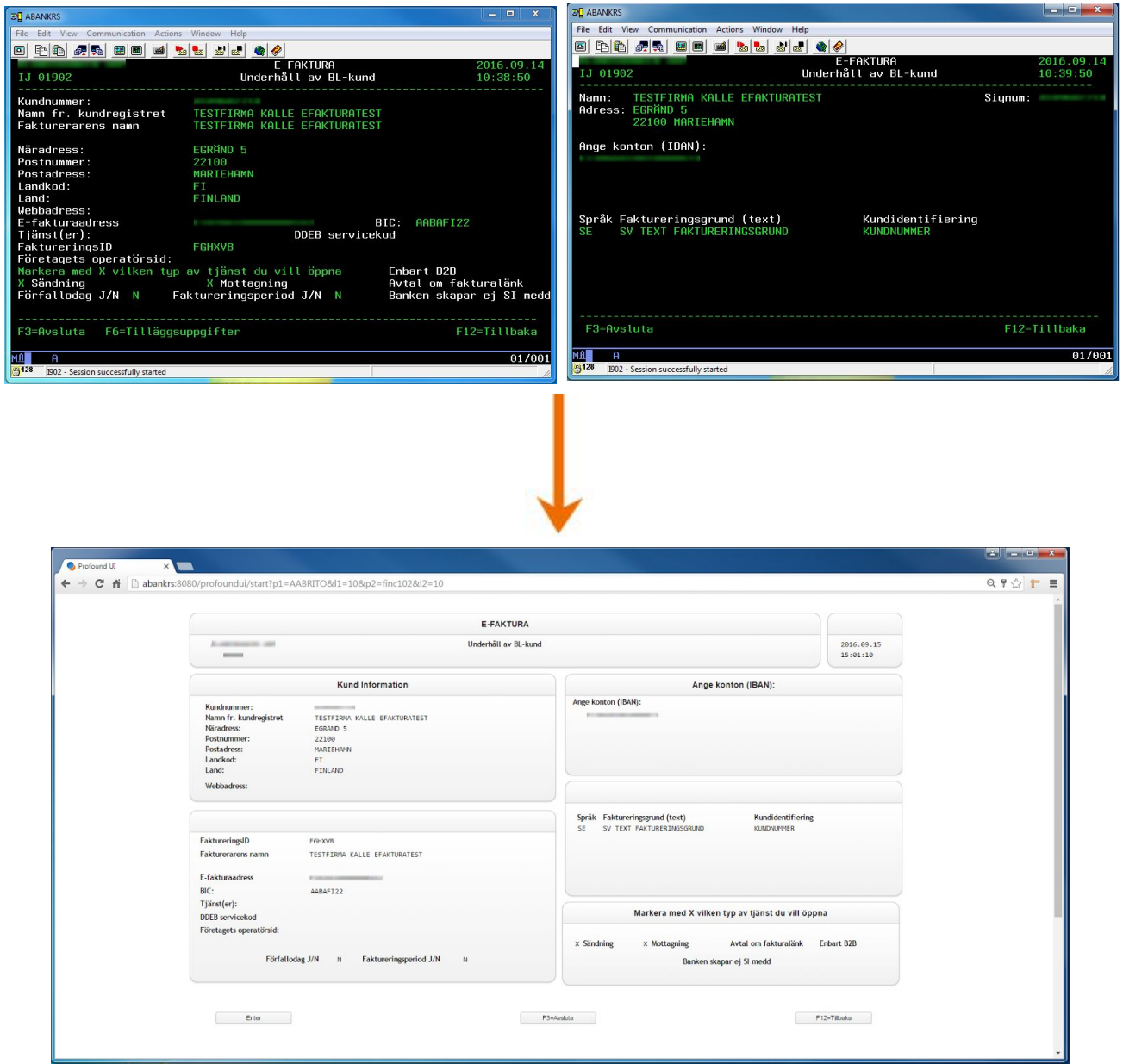


Figure 9. Example of Screens that was Converted and merged into one web application. The RPG code was changed after the DDS-Conversion and the screen was Designed further in the Visual Designer.

5. SPECIFIC AREAS

5.1 Theme code

The Theme is the DDS-Conversion rules that define how the DDS-Conversion should handle the screen properties. The rules are simple JavaScript objects that the conversion code takes in account when performing a DDS-Conversion.

Defining different properties and designs for the screen properties are done by assigning JavaScript objects values. Depending on the value the DDS-Conversion will detect the screen properties differently. The Theme is also built so that JavaScript functions can be written in the Theme, limited to some fields that can turn into function calls. These functions have in parameters that contain information about the screen. The Profound UI provides a API to get information about the properties and set property values of the properties. By using the in parameters and the API new kinds of buttons, widgets and much more can be created with JavaScript code.

The design rules of the web application after conversion is handled by a CSS file provided by Profound. Some of the objects in the Theme are assigned a CSS class that will change the design of a certain property. A CSS file can be created to change the design of all Theme properties, this is covered under the 5.3 CSS section. A example of the Theme and linking of CSS classes are shown in Figure 10.

```

"show fkey name": true,
"button type": "hyperlink",
"button css class": "link",

"items": [
  {
    "id": "ActionsPanel",
    "field type": "output field",
    "value": "",
    "left": "5px",
    "top": "52px",
    "height": "454px",
    "width": "165px",
    "alt_height": "720px",
    "z index": "8",
    "locked in place": "true",
    "css class": " actions-panel",
    "css class 2": "stationary"
  },
],

```

Figure 10. Theme consisting of JavaScript objects that contains the design rules.

5.2 Skin code

A Skin consists of three files, a JavaScript for configuration and customization, a CSS file for style and a HTML file to link them together. All the files define the settings and the way Genie handles the emulation for that Skin. These files can be edited from the administration page or using a code editor that can access the IFS folders. There is a config JavaScript file but it should only be edited using the Genie administrator tool.

The CSS contains customized stylesheet for that Skin and with the global CSS defines the look. Read more about the CSS files in the 5.3 CSS section.

The custom JavaScript file contains design and property specific JavaScript for the Skin. This file can be coded to change the conversion process for the desired look. In the JavaScript the different screen properties can be detected and changed. For example finding special function keys that are not supported by the default Skin settings.

The Profound API supplies functions for detecting properties, changing values and much more. The API is used to get data about the properties for coding solutions for handling them. An example for the coding is a special case of function keys that the Skin settings couldn't detect. By using the API the data was collected and code for adding the function keys was made, see Figure 11.

```
/* Replaces F8/F9= with two buttons
* */
var fields = getOutputFields();
var pattern = /(F)\d(\w)(F)\d(=)/g;
for (var i = 0; i < fields.length; i++) {
    var field = fields[i];
    if (field.className == "A22" && pattern.test(String(getElementValue(field.id)))) {
        hideElement(field.id);
        var splitarray = String(getElementValue(field.id)).split("=");
        var fkeys = splitarray[0].split("/");
        buttons.push({
            fkey: fkeys[0],
            text: fkeys[0] + "=" + splitarray[1]
        });
        buttons.push({
            fkey: fkeys[1],
            text: fkeys[1] + "=" + splitarray[1]
        });
    }
}
```

Figure 11. JavaScript for a special case of the function key detection.

5.3 CSS

CSS controls the look of the web application. Genie has its own CSS for each Skin, but the built in Profound CSS has a big part of the look of Genie. The Visual Designer CSS is also provided originally by the built in Profound CSS but an own CSS can be created that will be loaded when the applications starts.

The Genie Skin CSS contains properties that are special for that Skin and can be directly modified in the Skin. Some of the properties are however controlled by the Profound CSS or by JavaScript. If the properties are controlled by the Profound CSS it's advised to make a CSS file that will contain the override CSS code and link the file in the HTML.

The DDS-Conversion and Visual Designer use the Profound CSS. To change the properties as needed, a CSS file needs to be made. The CSS file will contain the new design of the conversion. The classes can either be copied from the Profound CSS or made from scratch.

If a similar design of the DDS-Conversion and Genie Skin are needed, the global CSS can contain both codes and the classes can be linked. By linking the global CSS in the Genie HTML both the Genie application and the other Profound applications can have the same design without having multiple CSS files. In Figure 12 example code for an action panel is shown. This is done in the global CSS file for same design of multiple action panels in the different products.

```
15 @.actions-panel {  
16     background: white;  
17     color: #474747;  
18     border-radius: 4px;  
19     border: 1px solid #d5d5d5;  
20     -webkit-box-shadow: 0 2px 3px rgba(0, 0, 0, 0.05);  
21     box-shadow: 0 2px 3px rgba(0, 0, 0, 0.05);  
22     overflow: visible !important;  
23 }
```

Figure 12. Example of a CSS class containing Design Properties for an Action panel.

5.4 Problems

5.4.1 Genie

A few problems were encountered during the development of the Skin. The first problem was the inconsistencies in the screens. Writing general code for detecting properties and assigning values to them, problems will occur if the screens contain many special cases of properties or inconsistencies. This was a common occurrence in the current system due to the large number of developers and the different styles of development over the years. To take into account every special case through general code cannot be done. Some of the frequently appearing problems were resolved by coding general solutions but the rest of the special cases needs to be resolved through the Designer mode in Genie or fixing the original applications.

Converting on the fly, the underlying code in Genie detects function keys and more. The settings for function key detection can be changed for example by writing regular expressions (regex). When writing the regex I found that the whole function key detection stopped working properly if the regex was changed, example of the regex is shown in Figure 13. After some testing I reached out to the Profound company and after discussions they provided a patch that resolved the issue. The same patch included improved menu detection options also done with regex, which facilitated the detection of different kinds of menus.

```
2  
3 pui["function key pattern2"] = "(F|CA|CF|CK|CMD|CM|Enter) ?([0-9]{1,2}) ?([=: -]) ?(\\S+)";  
4
```

Figure 13. The Settings for the function key detection that broke the detection functionality.

5.4.2 Visual Designer and DDS-Conversion

Writing the Theme and testing the different pre-installed Themes some issues arose. The biggest problem in the DDS-Conversion process was that the text to all function keys wasn't detected. After some investigation and discussion with the suppliers of the Profound UI product the problem was found to be the way the function key text was written to the screens.

To language customize the screens Crosskey uses message constant fields on the screen. The message constants are supplied from a library with different text for different languages. Depending on which library the user have in their library list the text change accordingly to the language of the user. The conversion doesn't detect the message constants but only the text that is in the constant. The text will appear after the conversion on the function keys but the message constant will not be linked, meaning the text will not change according to the language. This can be fixed manually by linking the values to the right function keys, but on large menus this is quite a lot of work and time consuming. Further discussions were made with the Profound company and a possible future functionality upgrade to take this problem in account was discussed.

5.4.3 Contact with Profound

As mentioned previously in this chapter a discussion and email exchange with the Profound Logic Company was made. The issues were either resolved with patches or possible solutions were discussed. The company was helpful even if the licence of the product was only a test license.

5.5 Overall Security and Login

Profound UI provides a few ways of authentication. The default identification is using the IBM i profile. By using the IBM i user profile for authentication the program functions as if the user was using the IBM i Access emulator, giving the right user profile authorities and session behaviour. The session behaviour can be adjusted but it will timeout after a certain time and logout if the browser is closed as on the IBM i Access emulator.

Another option is using Kerberos for login, which is a network authentication protocol that uses tickets for authentication. This will enable single sign on access if it's needed, which could be useful if a sign on is not required every time accessing a IBM i application through Profound UI, like when using a link from another web page with already signed in user.

5.5.1 Security and underlying structure of Genie

The Underlying structure of Genie is a IBM/Apache server that is configured to function as a IBM i 5250 session with the same session and security as before. The only new concern for security with the Genie application is the functionality to use SQL for retrieving data for fields while running the app. It can be handled by using correct IBM i profiles or disabling the SQL functionality if it's not needed.

The data sent to the Genie web application are sent in plain text as in previous 5250 applications. If Genie is to be used outside a private network the data need to be encrypted, for example by using SSL. The IBM i OS provides the software for running SSL and Genie supports it.

6. CONCLUSION

The tests, evaluations of the programs, the proof of concept were done by comparing the old system and the new system with the Profound UI products. When comparing the new and old system the most important thing to investigate was the usability and consistency so it worked as it was supposed to and that no functionality was lost. Continuous testing of the functionality was done during the development of both products. The result was given to a product specialist for further input on how the new system worked.

6.1 Genie evaluation

The Genie product makes the IBM i system available on the web with a few steps and only a small amount of development. To get a good and fully functioning system can be more challenging. Genie is based on an on the fly conversion so to get the conversion perfect, many small imperfections must be resolved. The problem with old systems like the IBM i at Crosskey is that there are many inconsistencies and a general Skin cannot be used as well as it is intended. The inconsistencies in the original code come from many generations of developers and different development techniques. However, most of the small inconsistencies can be resolved using the Designer mode for editing. The downside is the workload of fixing each special case can be large.

6.2 Visual Designer evaluation

The Visual Designer is a good tool to create easily designed screens. The controls are simple and easy to understand. The move from designing screens in code or in the RDI graphical designer takes some time but when learned it is easy to make a simple screen from scratch. The tools and the options for designing new screens give the possibilities to fully use the capabilities of the web technologies. When making new programs and designing a new screen the full advantage of the web can be used but the capabilities of RPG programs still restrict this in some ways.

6.3 DDS-Conversion evaluation

The DDS-Conversion converts all necessary data from the old files, but gives a quite basic design. The conversion process is handled by the settings in the Theme and the ability to affect the conversion process is quite limited. To get more out of the Theme, JavaScript can be coded but the information provided is insufficient.

The DDS-Conversion does the job and leaves a basic screen for further designing, but it is not a completely automated process. The screens can be easily modified and designed. But a more versatile conversion would mean less work. To get full use of the web technologies the RPG programs can be changed. The business logic in the programs can now change to take advantage of for example the size of the screens. Changing the underlying RPG code can be difficult but the results are worth it.

6.4 User reviews

The overall expressions by the product specialists were that the programs worked with minor changes to the working process. The experience was good, but small problems were found. In the Genie product many small inconsistencies were found and a few of them were resolved using the Designer mode. As said earlier in the text the inconsistencies often originate from the conversion process not recognizing the different inconsistencies in the original screens. The programs done in the Visual Designer functioned as before but now with improved design and in some cases with better overview of the information. The webification of the applications had a positive outcome but the new way of doing things will take time to learn. (Product Specialist, 2016)

6.5 Overall evaluation

Overall the Profound products provide a good step towards modernizing IBM i, but the products need to do it better to compete with native solutions for the web. The RPG programs still restrict the full use of modern web or application development for the web. As a middle step between completely moving to new technologies or remaining as it is, the Profound products are good step for moving the IBM i to the web and modernization of the interface.

6.6 Final words

This thesis presented how the software product Profound UI works and how simple concept programs of each part of the product have been made. The thesis introduces a brief explanation of the programming languages, techniques and the different Profound UI programs. The test programs and proof of concepts that were made will hopefully serve as a base for the decision to the pursuit of a test of modernization of a whole product area. A report and a technical guide document was handed in to the supervisor at Crosskey.

The work of this thesis was done during vacation times and limited the focus on one product area, instead many small areas were tested. Focus was on the technical parts and the conversion of existing applications. More focus on design could have led to a better visualization of the product capabilities and a more thorough user study could have been made to get more opinions on how the product changed the workflow. The limitations were followed but some other parts of the software needed to be included more in the work, like the Open Access Handler.

For the future, an improvement could be a better design proposal by a professional designer and a conversion of all programs in a product area. This could not be done by this investigation due to limited resources. Focus was at concluding if the software Profound UI is at all suited for the demands by Crosskey.

REFERENCES

- Alex Roytman, & Profound Logic. (2015). Perform modernization magic with profound logic software. Retrieved from http://www-03.ibm.com/systems/data/flash/fr/resources/modernisation_i_2015/S19_-_Offrez_une_interface_Web_et_mobile_a_vos_applications_IBM_i_avec_Profound_UI_et_RPG_Open_Access.pdf
- Crosskey. (2016). About crosskey. Retrieved from <https://www.crosskey.fi/our-story/>
- IBM Corporation. (a). DDS for display files. Retrieved from http://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/rzakc/kickoff.htm
- IBM Corporation. (b). RPG. Retrieved from http://www.ibm.com/support/knowledgecenter/ssw_ibm_i_72/rzahg/rzahgrpgcode.htm
- IBM Corporation. (2010). Rational open access: RPG edition. Retrieved from http://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/books/rzasm.pdf
- IBM Corporation. (2016a). IBM - personal communications. Retrieved from <http://www-03.ibm.com/software/products/en/pcomm>
- IBM Corporation. (2016b). IBM i—An efficient, resilient platform for business. Retrieved from <http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=PM&subtype=BR&appname=pob03001usen&htmlfid=POB03001USEN&attachment=POB03001USEN.PDF>
- IBM Corporation. (2016c). IBM rational developer for i integrated tools for application development on IBM i. Retrieved from http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=SP&infotype=PM&appname=SWGE_RA_ZK_USEN&htmlfid=RAD14118USEN&attachment=RAD14118USEN.PDF

#loaded

Mozilla Developer Network. (2016a). CSS. Retrieved from

<https://developer.mozilla.org/en-US/docs/Web/CSS>

Mozilla Developer Network. (2016b). JavaScript. Retrieved from

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Product Specialist. (2016). *Email communication with two Product Specialists at Banking at*

Crosskey 18.7.2016

Profound Logic. (2016). Profound UI documentation. Retrieved from

<http://www.profoundlogic.com/docs/display/PUI/ProfoundUI+Documentation>

Profound Logic Software. (2016a). The 2016 state of IBM i modernization white paper.

Retrieved from <http://info.profoundlogic.com/2016-state-ibm-i-modernization>

Profound Logic Software. (2016b). Profound logic. Retrieved from

<http://www.profoundlogic.com/>

Scott Klement. (2015). RPG and the IFS. Retrieved from

<https://www.scottklement.com/presentations/RPG%20and%20the%20IFS.pdf>

W3schools. (2016). HTML; Retrieved from <http://www.w3schools.com/html/>