User Centred Design Principles for IT Development Projects

Dr Michael Baron

About the Author

PhD (Curtin University), MEM (ECU), Grad. Certificate In Industrial Education & Training (RMIT). BA (University of Melbourne)

Dr Michael Baron was born in Moscow, Russia in 1976. He is the founder and Managing Director of Baron Consulting, a boutique Melbourne-based consulting firm. He has been involved in a number of large-scaled technology implementation projects in Australia, New Zealand, Singapore and China. Michael also lectures MBA and MIS students in some of the leading Australian and International Universities.

Preface

User-centred design (UCD) is a comprehensive framework for information system development in which usability goals, user characteristics, environment, tasks and user requirements are used as a guideline for the system development in order to achieve maximum level of customer satisfaction by tailoring the resulting output to the users' needs.

The very objective of any IT development, implementation or integration project is to achieve business goals of the stakeholders in an optimal way. In the end of the day, it is the level of user satisfaction and performance efficiency that makes the respective project a failure or a success.

This book has been written with the aim of equipping IT Business Analysts with the core User-Centred Design (UCD) skills. The target audience of this book is fresh University graduates who are on the one hand ''fully qualified'' Business Analysts (BAs) but on the other hand are unfortunately still likely to have limited understanding of how to apply the User-Centred Design Techniques and Practices in the ''real world''!

This is why this book aims to focus on practical side of the UCD rather than ''theory''. It attempts to provide ''real-life'' examples and looks into the ways UCD is carried out by private and public organisations in a range of industries. To make the learning more ''hands-

on'', each of the Modules incorporates Snapshots and Case Studies from a single industry. For example, *Module 1* looks into the Education industry, while *Module 2* considers use of UCD in the Banking sector.

I hope that you are not only going to enjoy working through the book but will also become empowered to make the transition from being a good student or a hopeful graduate to a successful IT professional!

Good luck,

Dr Michael Baron,

CEO

Baron Consulting

Table of Contents

Snapshot Directory

Module 1

User-Centered Design: Framework and Principles

1.1 Setting the Scene

User-Centred Design (UCD) is a comprehensive framework for information system development in which usability goals, user characteristics, environment, tasks and user requirements are used as a guideline for the system development in order achieve maximum level of customer satisfaction by tailoring the resulting output to the users' needs.

In other words, it is the process of gathering stakeholders' requirements and designing systems and applications that match these requirements best!

Let us consider the following scenario:

You are leading a team of E-Learning developers and you have been contacted by a highly successful college that is currently delivering its training programs exclusively via a traditional delivery mode (classroom/face-to-face delivery). The college however feels it is ready to tap into the lucrative marketplace of E-Learning. As of now – it has not moved its offerings online beyond having a simple billboard-style website.

The college wants to commission you to build the E-Learning Platform ''from scratch'' and to assist in moving all of the courses online. Unfortunately, as they are new to the very concept of e-learning, they are happy to provide you with the budget (hopefully appealing

enough for you to get on board), timeframe and scope for the project as well as with complete information about their current business environment (courses, students, resources, facilities etc.). And now…the ball is in your court!

You take a look at the college's offerings. You really like the courses they are delivering. By accepting to take charge of the project, you will not only earn well, but will also do your modest share in turning the world into a better place by making low-cost yet high quality training courses more accessible. And the good news is – you already know about E-Learning. You have great knowledge of the E-Learning platforms, sound understanding of the training and assessment processes, solid idea about the costs and operational requirements involved so with an easy heart – you reply with ''Yes''.

But…how shall you get started?

The very first step that you need to undertake is:

ENSURE THAT YOUR PROPOSED DESIGN MATCHES THE CLIENTS' REQUREMENTS

This can be done by the means of:

- Achieving explicit (rather than implicit) understanding of the users, business processes, tasks, requirements and challenges
- Involving the system users proactively throughout the process of design and development
- Adjusting the design based on the feedback collected
- Balancing between requirements of different User Groups

1.2 Achieving explicit (rather than implicit) understanding of the users, business processes, tasks, requirements and challenges

Gaining detailed understanding of the users' requirements is a laborious yet an essential task. Understanding of the business processes has to be acquired PRIOR to considering technologies to be used!

One of the greatest challenges of UCD is ensuring that the analysts have sufficient understanding of the industry the projects are aimed at. In our case (see the scenario above) –developers require understanding of the E-learning processes and tasks as well as the processes and trends in the Education industry in general. Without a solid industry-related knowledge base, no UCD design will be possible. Likewise, UCD designs for other industries (e.g. banking, retail, insurance etc.) require in-depth knowledge of these respective industries. It does limit ''business opportunities'' for the developers as they are competent to undertake only certain projects but it ensures that the developers can achieve business process optimisation. It would be hard to develop optimal business processes based on the documentation provided only as opposed to having consistent exposure to the environment these processes belong to.

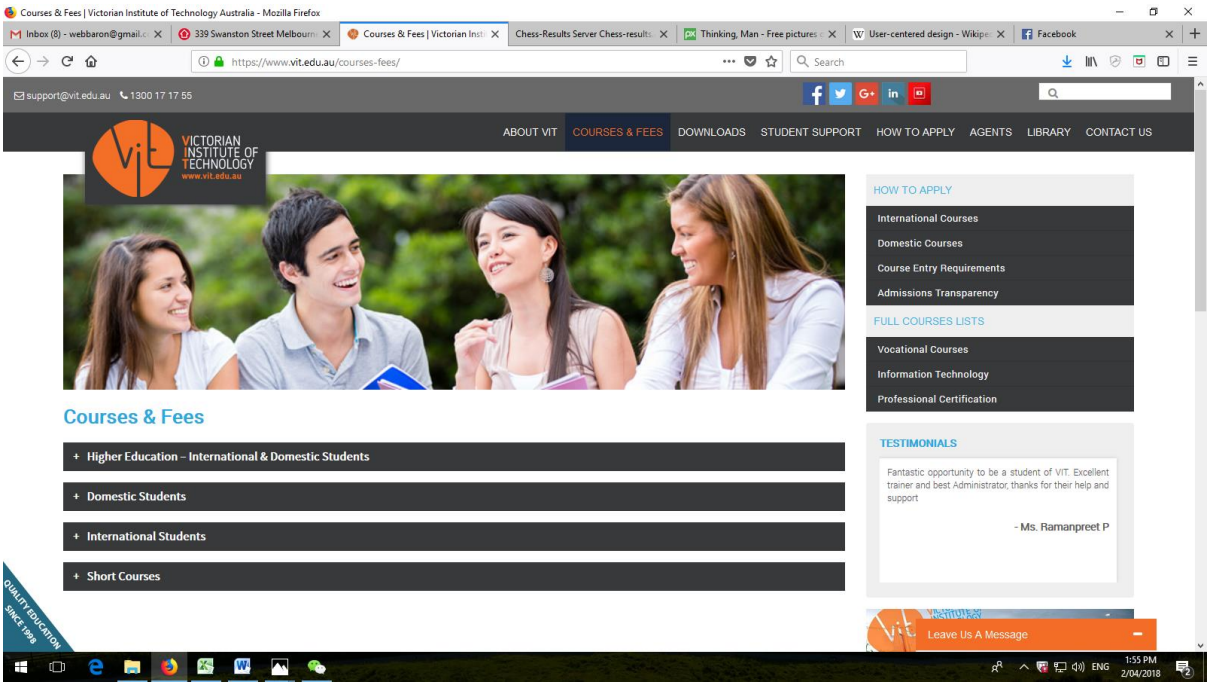
As well as in-depth understanding of the industry requirements, client-specific requirements should be considered. The requirements could be both *restrictive* and *innovative.*

The *restrictive* requirements involve addressing client-specific objectives as within any single industry – there are many different vendors who deliver their

services in a variety of ways. In our case, we are dealing with the Education Industry – one of the most diverse industries of all. Therefore, the service delivery requirements will vary accordingly.

Below are 2 examples of the *restrictive* requirements for the User Interface Design:

The first of the 2 **screenshots** is for the ‘’Courses and Fees’’ page of the Victorian Institute of Technology. The institute’s current niche market is international students and majority of the courses offered are TAFE-level courses. This results in ‘’Fees’’ becoming a critical piece of information that potential customers are looking up first. No wonder that ‘’Fees’’ related information (pricing, availability of instalments etc.) and services (bill payment) can be accessed via direct link from the college’s Homepage. Both informational content and functionality of the ‘’Fees’’ section are assigned primary roles within the service delivery infrastructure.

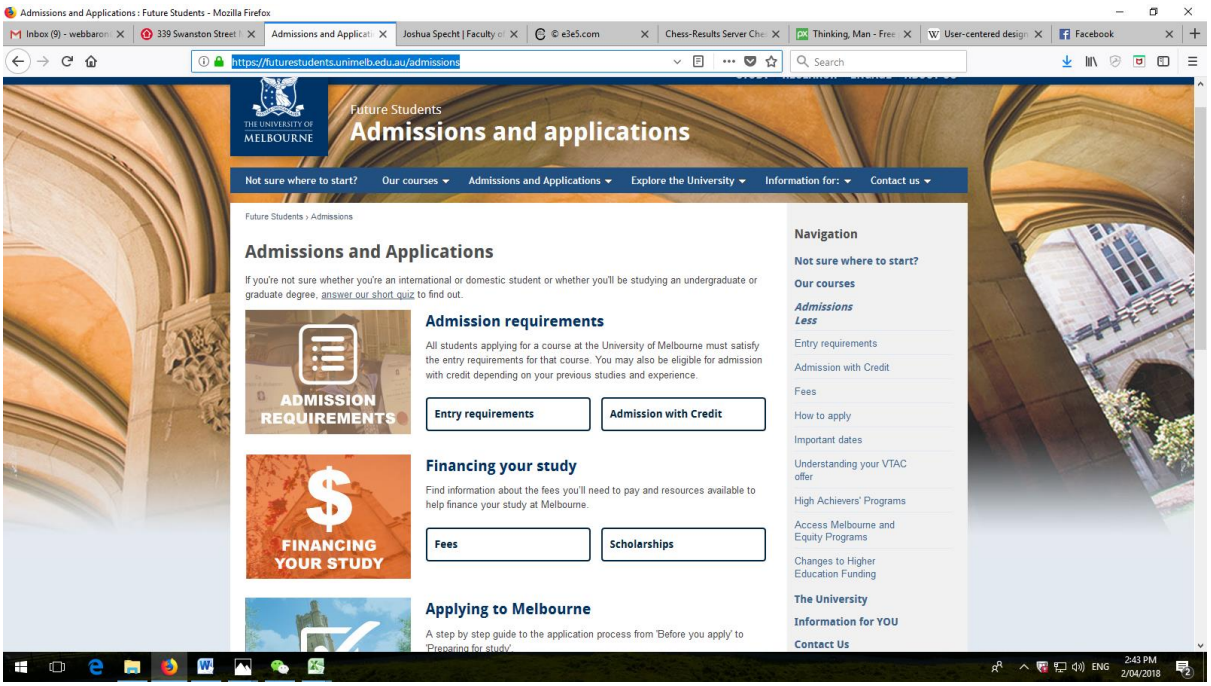


1.1VIT Courses and Fees Page Screenshot, made on April 2nd, 2018.

Source: https://www.vit.edu.au/courses-fees/

On the other hand, while all of the training providers need to share (and usually do share) their ‘’Fees’’ information online, in cases of the majority of larger and more established providers (such as leading Universities), the fee-related content is not always of primary importance and can consequently be demoted from the main pages of their websites.

The screenshot **1.2** below is for the Admissions page of the University of Melbourne website. The website does not provide ‘’Fees’’- related information and processes directly from its Main page. Instead, link to the ‘’Fees’’ can be accessed only indirectly from the ‘’Admissions’’ section. Even within that section, it is just one of the many links available and is not highlighted in any way to stand out. In the very same column as ‘’Fees’’, links that are traditionally perceived to be of lesser significance such as for instance ‘’Access Melbourne and Equity Programs’’ can also be seen. It is a clear indication of an important informational content element being either underestimated or displaced.



1.2 University of Melbourne Admissions Screenshot, made on April 2nd, 2018.

Source: https://futurestudents.unimelb.edu.au/admissions

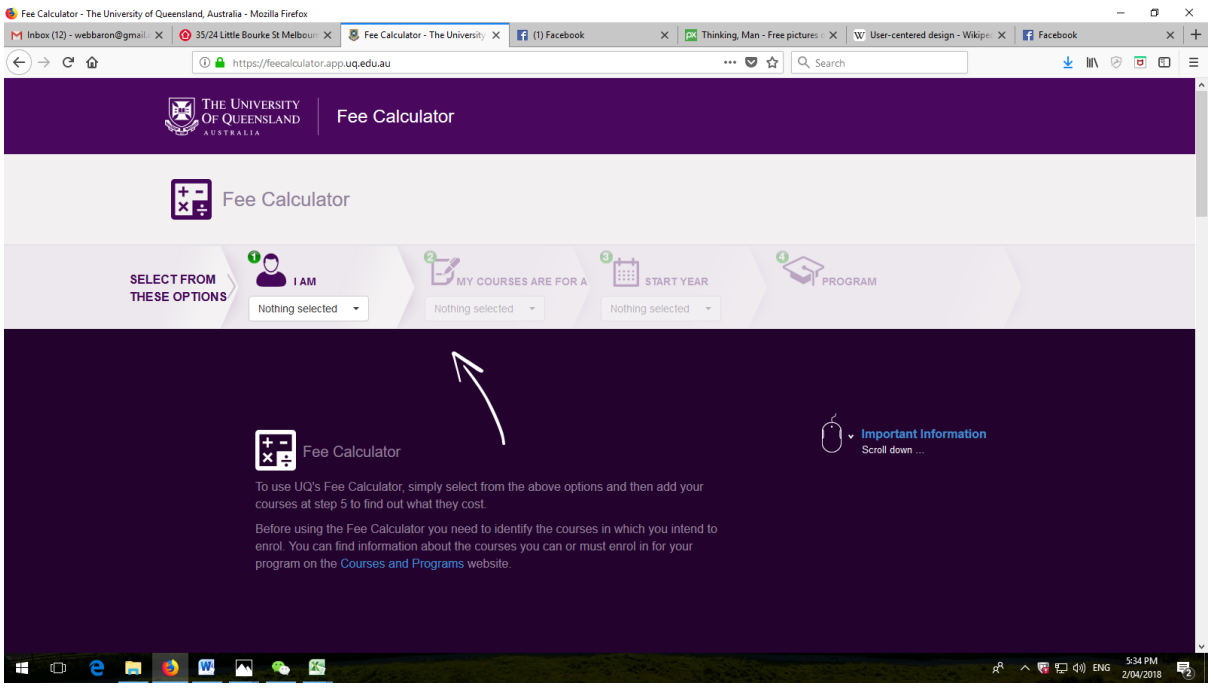
While the *restrictive* requirements ensure that essentialities emphasised by the clients are incorporated into the UCD projects accurately, we should be proactive as well as reactive. The proactive approach involves identifying the so-called *innovative* requirements.

Ability to identify and implement *Innovative* requirements is central to making the project outstanding! *Innovative* requirements refer to identifying features and processes that will empower the system by the means of making it unique. It is innovation that often helps organisations to utilize technology to get ahead of competition.

Within every industry, system users (irrespective of whether they are back-end or front-end users) have a tendency to compare different vendors with one another. UCD should be looking into technology-driven solutions to develop systems that not only satisfy basic requirements but also ‘’deliver what other systems do not’’.

In the example below, The University of Queensland is using *Innovative* approach to management of the ‘’Fees’’. Instead of making students browse from different courses to find out how much they have to pay for each of the respective study programs, the UQ website offers a Web Tool called ‘’Fee Calculator’’.

In order to use UQ's Fee Calculator, visitors to the website simply select from the list of options provided (e.g. domestic/international, level of study, faculty etc.) and then add their courses to find out what they cost. This is a step-by-step process that is easy for the users to follow. Furthermore, in order to make the tool easier to use for the newcomers, the tool is linked to the Courses and Programs website that includes Program names, subject and program codes etc. Having such a tool provides UQ with an *innovative* edge over other training providers!



1.3 UQ Fee Calculator Screenshot, made on April 2nd, 2018

Source: https://feecalculator.app.uq.edu.au/

1.3 Involving the System Users Proactively Throughout the Process of Design and Development

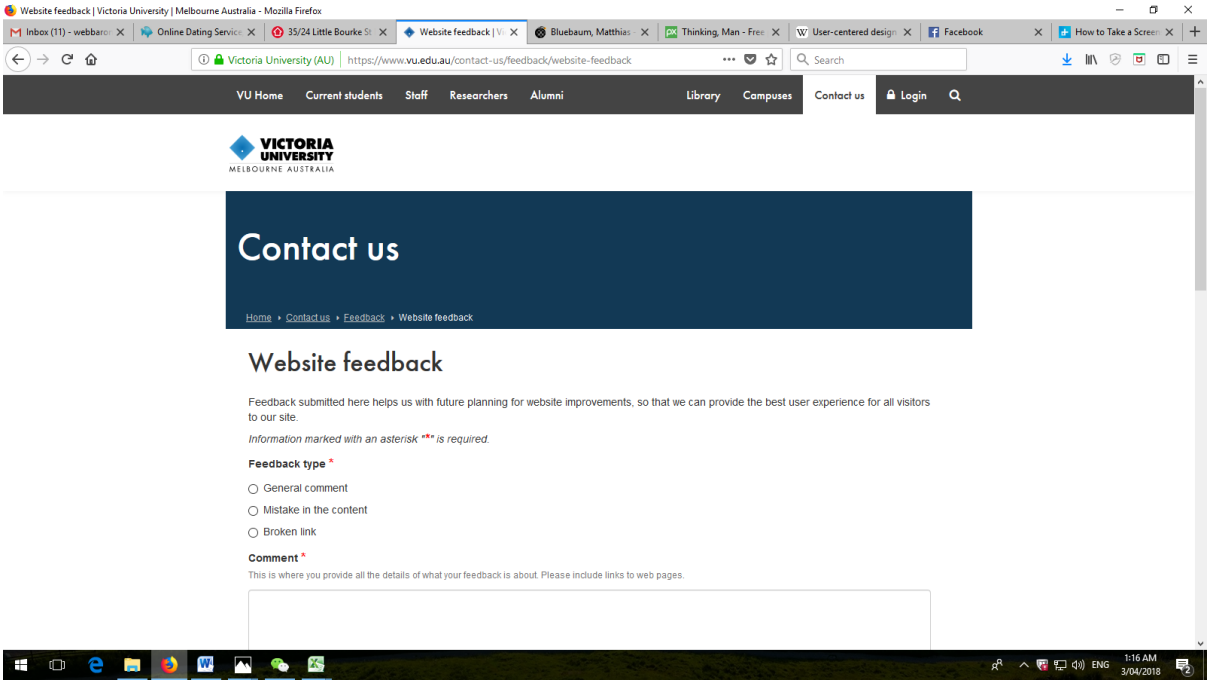
The very concept of UCD is to create a system that is best-tailored to the users' needs. This would not be possible without ongoing proactive involvement of the users. Their feedback is instrumental for achieving the UCD goals.

One typical mistake is to limit the users' involvement to the requirement gathering stage and from there onwards – failing to acknowledge their ''existence''. The more information we can keep collecting from the users and the more regularly the feedback is going to come, the more likely we are to sustain high level of the user satisfaction. Also, as time goes by, the users may also become better-versed in the project and quality of the feedback collected (relevance, clarity of expression etc.) is also going to improve considerably.

As UCD requires involvement of both *back-end* and *front-end* users, developers need to identify a workable strategy to keep all of the users proactive. The main challenge is keeping the *front-end* users involved. Needless to say, this should be done without making the complete project details available to them as they are not the ones who are commissioning the projects (they are the future ''customers'' the systems will be servicing) so the task should be approached rather delicately.

In case of the Education industry, the *back-end* users are educators, e-learning support officers and university/college admin staff, while the *front-end* users are potential and/or current students. Consistent involvement of the back-end users can be achieved by making a schedule for contact group meetings with the stakeholders. It is only logical that stakeholders you are developing the system for will be more than happy to contribute. In fact one of the most common complaints about the development teams is that they do not ‘’listen’’ to remarks and suggestions coming from the users. Well, the UCD approach will resolve this potential source of the stakeholders’ dissatisfaction amicably.

Involving *front-end* is going to be much easier if the UCD project targets those who are already involved with the parent organisation (e.g. current students of a college). They can be requested to attend contact groups, fill surveys or even to submit proposals on what additional processes/improvements to the current processes should be carried out for improving the user experience further! It is likely that at least some of these *front-end* users will get involved particularly subject to some kind of small incentives provided.



1.4 VU Website Feedback Snapshot, made on April 2nd, 2018

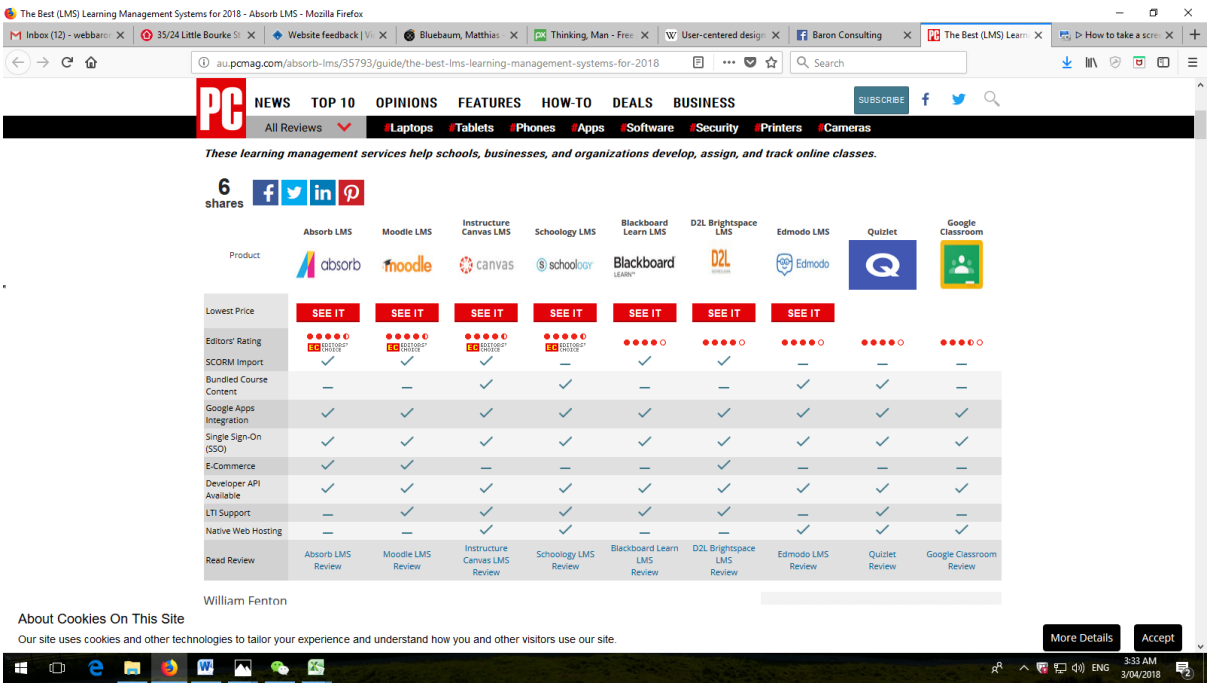
Source: https://www.vu.edu.au/contact-us/feedback/website-feedback

The snapshot **1.4** is a good example of collecting feedback from current users of a system. In this particular instance, the feedback is collected in a well-structured manner that enables the UCD developers not only to identify nature of the users' dissatisfaction but also to distinguish between the user groups.

When submitting Feedback, the users are asked to indicate whether they are staff (aka *back-end*) or students (*front-end*) users of the system. This enables the developers to pinpoint areas where improvements/further development is needed. Some of the problems experienced by the front-end users may remain unnoticed by the back-end service providers if not for the Feedback. Many of the Learning Management Systems (LMS) have significant differences between the Staff and the Students' virtual homes. The student' interfaces incorporate study activities, materials

and assessments while the staff ones need to also include administration and content management functions.

UCD can also collect user feedback via ‘’*observation*’’. This can be done by examining users’ activities when logged into the LMS: website pages they access, learning exercises they undertake, functions and tools used etc. All of the mainstream E-Learning platforms have capacity to monitor user’s activities and consolidate these activities into a report format. Such reports can be customized based on the activity information the developers are after.



1.5 The Best LMS for the Year 2018 Screenshot, made on April 2nd, 2018.

Source: http://au.pcmag.com/absorb-lms/35793/guide/the-best-lms-learning-management-systems-for-2018

The screenshot **1.5** shows the ranking list for the Top 10 Learning Management Systems (LMS) available for e-learning development/online training delivery in the

year 2018. As evident from the table captured by the screenshot, all 10 of the e-learning platforms have at least some capacity for customization of the content and processes required for the online delivery of training programs. Some of these platforms (such as Moodle or Blackboard) also provide a range of features for integration of other tools into the platform. In other words, the platforms are flexible enough for the developers not only to implement UCD during the initial development/reengineering of the system but also to continue to do so on an ongoing basis.

The concept of UCD is one of the central features of E-Learning development. User preferences in the education industry vary significantly so without UCD, it is hard to capture needs of the target audience.

Right now, Moodle is regarded by majority of the training providers as the best and most user-friendly e-learning platform of all. It is adopted by many of the Australian and international training providers (Universities, TAFE Colleges, Private Training Organisations, Learning and Development Units of Large Companies etc.) as the main platform for delivery and management of online courses.

Features of Moodle (shown on the screenshot 1.6 below) that could be of particular interest to the UCD-driven developers are:

- Moodle Analytics: Moodle Analytics allows UCD developers to track how the platform is being used. Based on the usage data collected, the users' preferences and needs can be determined and the platforms' setting adjusted accordingly.

- Extendible and Customizable: Moodle learning site is extensible and customisable with over 1400 public and free plugins created by global community of the Moodle users and developers. There is also a Moodle plugins directory so the users' requirements can be matched quickly.

- True Open Source: Open Source Platforms enable not only access to development resources but also keep costs of the developments (including ongoing upgrades, based on the user's feedback) down.

1.6 Moodle Screenshot, made on April 4th, 2018.

Source: https://moodle.com/

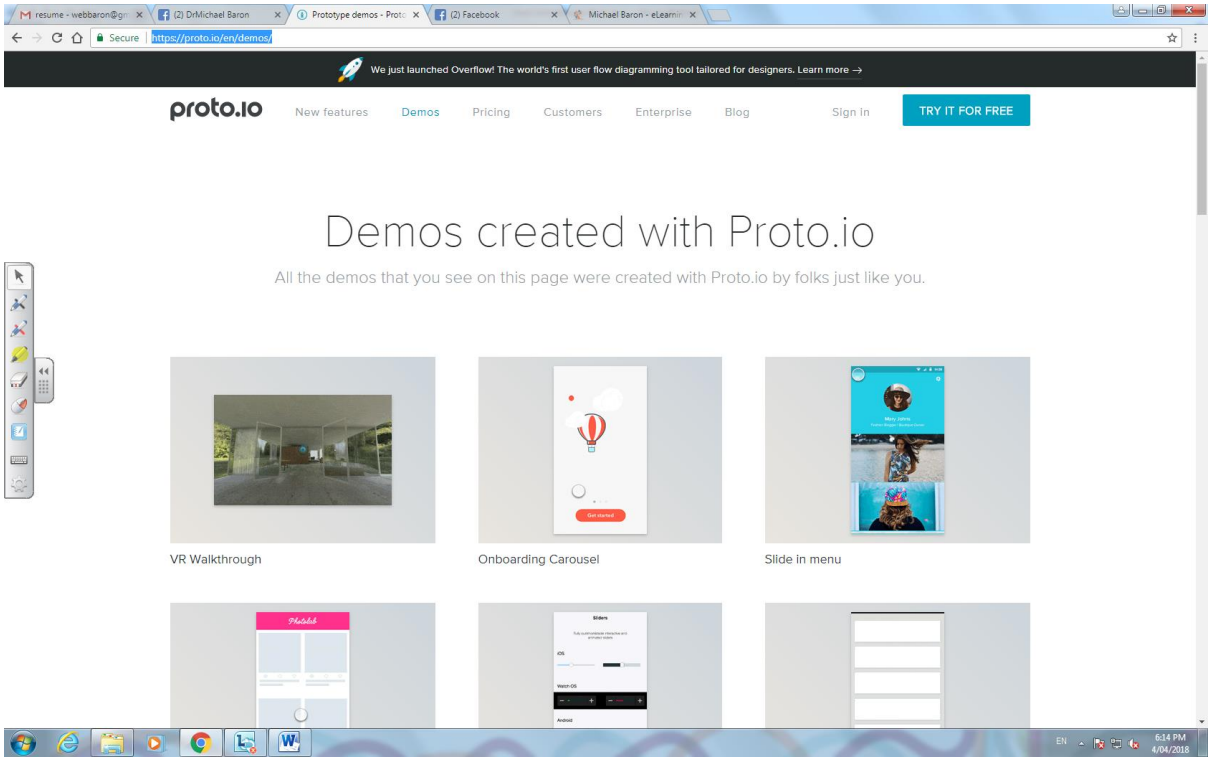
As Moodle is free to try and use, it is a perfect platform for the ‘’UCD developers in making’’ to polish their skills by registering a free account and examining the development and customization options available. Consider different requests, complaints and suggestions that could be coming from different user groups: How would you address those?

1.4 Adjusting the Design Based on the Feedback Collected

As we already discussed above, UCD would be impossible without all of the user groups contributing to design of the systems proactively. In order to be able to integrate their feedback into the UCD development process, we can consider the following options:

- *Micro-Prototypes* –

A working prototype can serve as a starting point for experimenting with the users' feedback. Providing users with access to the prototype is one of the best ways of letting the users ''have their say'' about the systems (and this is what UCD is all about). The advantage of using ''Micro-Prototypes'' as opposed to fully-scaled ones is the ease of usage. The system users will be able to focus on specific functions/system development requirements that are of greater concern to them. The prototype mode will enable developers to test the users' suggestions quickly and to see if they do improve the system's performance, user-friendliness and the user satisfaction level.



1.7 Proto Prototyping Tool Screenshot, made on April 4th, 2018.

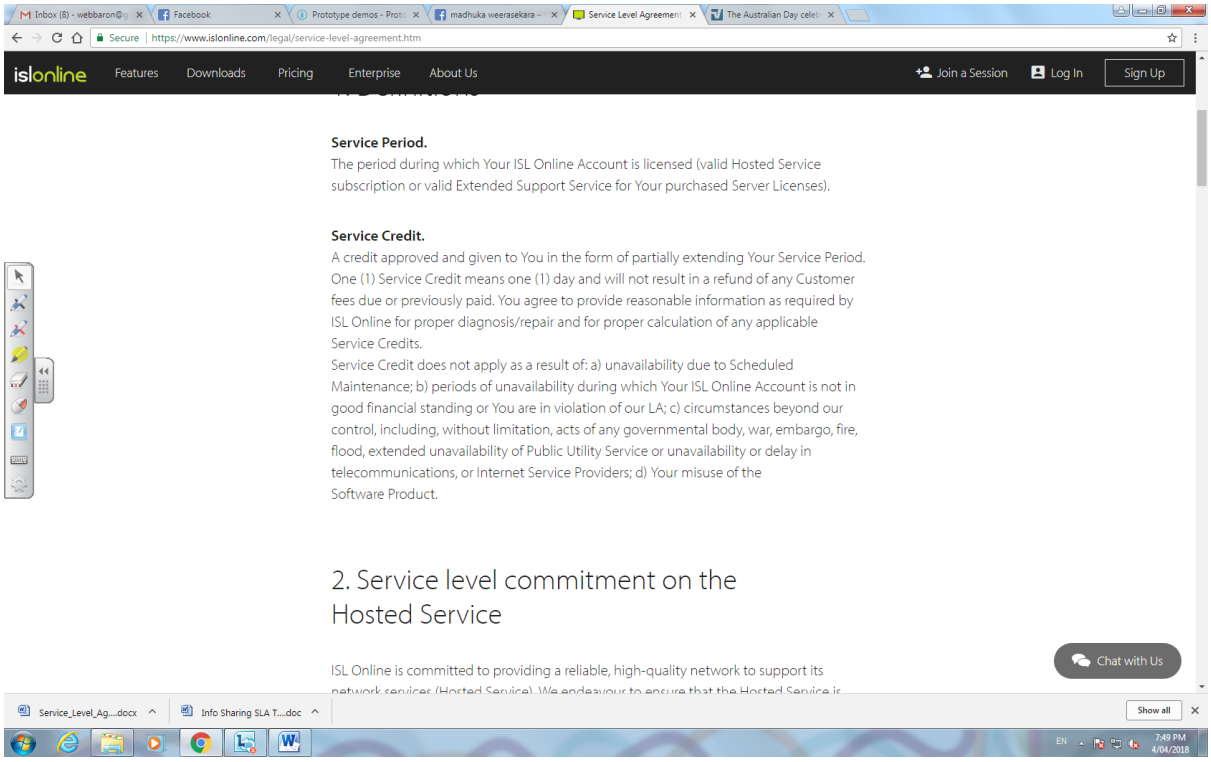
Source: https://proto.io/en/demos/

- *User-End Customization Tools and Options –*

Nothing empowers the system users more than ability to carry out ‘’do-it-yourself’’ customization. Giving users more customization options will lead the way to improving UCD’s linkage to the actual users’ requirements. If a developer notes that users opt for particular customization patterns, these patterns can be utilized in the overall design of the system. The focus on ‘’patterns’’ as opposed to specific changes is particularly important. Concrete customizations reflect individual users’ requirements while ‘’patterns’’ reflect requirements of user groups. For example, if many different users opt for the same pattern (e.g. Creating Virtual Note Pads) – it means that a note-taking function should be incorporated into the UCD.

- *Long-term SLAs –*

As UCD should ideally be managed as an ongoing project (to keep adjusting and improving the systems based on users' requirements) – it is good to be able to get engaged into continuous development even after the system is already in full use. Furthermore, from a commercial perspective, it is impossible to continue working on the system unless there is a long-term SLA between the UCD developers and their customers.



1.8 IslOnline SLA Screenshot, made on April 4th, 2018.

Source: https://www.islonline.com/legal/service-level-agreement.htm

A Service Level Agreement (SLA) - is a contract between a service provider (in our case – UCD developer) and a client. The contract defines particular aspects of the service such as quality, availability and responsibilities as agreed between a service provider and a service user. In a nutshell, the key objective of a SLA is that the services should be provided to the

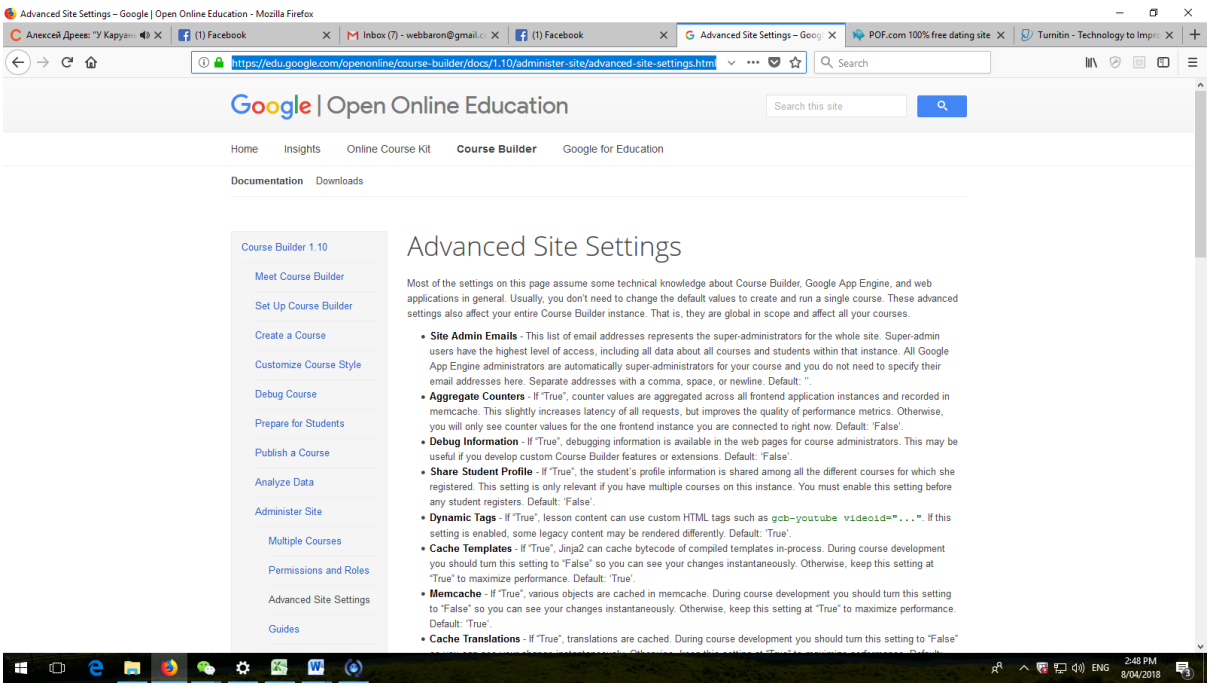
client in accordance with the contract that both sides (the service provider and the client) need to honour. The agreement will also include a time frame for the service delivery (e.g. 5 years).

Having an accurate and detailed SLA also makes the relationship between the client and the UCD developer more dynamic. It puts both parties into the same boat for years to come and consequently makes them more responsible as well as better-organized as a team. In a standard scenario (and this is where UCD-driven projects are different from other types of projects/other methodologies), a SLA for a UCD project will include not only tasks that the service provider is to accomplish and benchmarks for these tasks but also, descriptions of the supporting tasks that the client will be contributing throughout. Even if indirectly, clients (aka users) are going to find themselves being valuable members of the development team!

1.5 Balancing Between Requirements of Different User Groups

UCD requires addressing needs of all of the user groups. Obviously needs and preferences of the user groups vary. In case of the E-learning platform users (just like with any other service delivery portal) – the balancing should be carried out between requirements of the service providers (educators/trainers/administrators) and the service recipients (learners).

The differences start with system access rights. Learner's access rights are far more limited. The same limitations have to apply to the customization options and the critical challenge for UCD developers is to ensure that *customization at the learner end does not impact functionality and user-friendliness of the users on the Admin end and vice-versa.*



1.9 Google Education. Admin Settings Screenshot, made on April 7th, 2018.

Source: https://edu.google.com/openonline/course-builder/docs/1.10/administer-site/advanced-site-settings.html

The screenshot **1.9** above shows Advanced Site Settings Options version of the Google Open Online Education Learning Platform. The options provided incorporate virtually all of the service delivery functions and the inevitable question is, how will each and every of the system adjustments are going to impact the learners?

The important aspect of the balancing is not to assign %ages and relative values to the requirements (e.g. matching 70% of admin requirements and 30% of learner requirements) but instead to try to identify those UCD developments that are *improving system performance/user satisfaction of one of the user groups at the ''expense'' of other group(s) as this can be counterproductive.*

Ideally, UCD should involve some activities that enable developers to bring all of the user groups together to provide consolidated feedback on the systems designed but from a practical perspective, it is often very difficult and has to be done with great care. For example, in case of the e-learning, ''introducing'' students to the assessment administration options may increase the risks of these students trying to (eventually) gain unauthorised access to the back-end of the system to change their grade, grant themselves a fake assignment extension etc.

1.6 Module 1 Summary

In this Module, we have used the Case of E-learning to learn:

- User Centred Design (UCD) Definition
- Getting Started with the UCD
- UCD Administration
- Collecting and Incorporating User Feedback into the Development Process
- Core Challenges to the UCD

1.7 Module Review Questions

1. What is UCD?
2. What are the benefits of UCD?
3. Why do UCD developers require good knowledge of the industry they are designing for along with the IT business analysis/development skills?
4. How do we incorporate user requirements into the UCD?
5. How do we collect user feedback?
6. What are the challenges in implementing user feedback into the UCD?
7. How to balance user requirements of the different user groups?

Module 2

User Centered Design Elements

2.1 Identifying User Centred Design Elements

UCD projects are multi-facet ones. The designs need to span across 4 dimensions:

- Visibility
- Accessibility
- Legibility
- Language

It is important to note that all 4 of the dimensions are *NOT technical* and should therefore be considered *irrespective of the technologies used in the project.*

Accurate treatment of the design elements increases platform/application efficiency and eliminates trivial discrepancies such as use of language that some of the user groups may find difficult to comprehend. Failure to understand the content shared will consequently result in the users' lack of acceptance and/or ineffective usage of the system.

Let's Consider the UCD Design Elements in the light of the following Scenario:

Your UCD consulting firm has commenced working with a new client. The client is a boutique financial planning firm that has been assisting its customers with loan acquisition, all kinds of business investments, financial planning and property investment since 1997.

The firm has been developing its practice successfully by the means by establishing long-term working relationships with its customers as well as planning, implementation, and regular reviews of the services it provides. It is happy to service both corporate clients and

individuals who are in need of assistance in managing financial affairs and it values all of its customers.

As of now, the firm uses its website exclusively for Communication functions and does so very effectively. However, it wants to develop its online presence further and to make all of the services available online. This will involve radical re-development of the website to integrate a large number of applications and tools into the current portal.

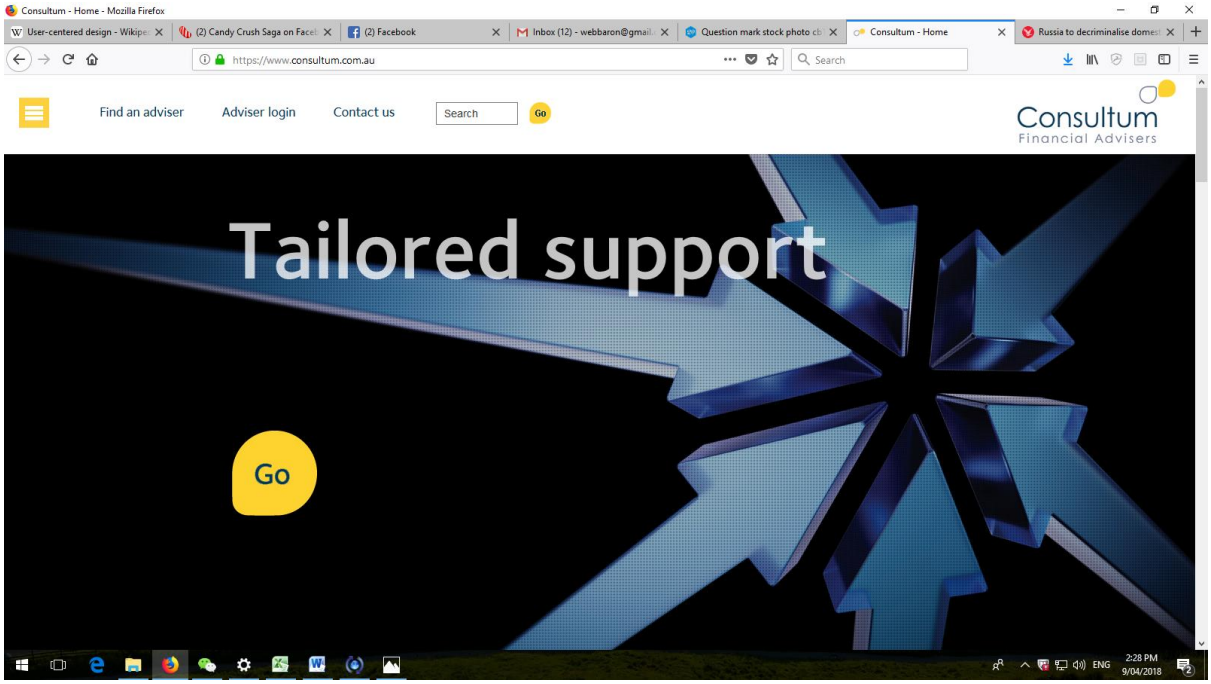
The ''new additions'' to the website will include:

- *A Loan Calculator*
- *Interactive Loan Application Form*
- *Financial Planning, Stock Market Analysis and Real Estate Market Analysis Apps*

The client wants links to all of these new additions to be available directly from the main Home Page and would like to see the entire re-development process completed within a fortnight.

So once again, lets' see if you are up to the challenge…Now that the E-Learning project (See *Module 1*) has been completed, you can apply your newly acquired skills in a very different industry setting (Finance) and see how usage of the UCD is going to differ depending on the change of the host industry. Also, the project requirements involve usage of tools and applications that some of the users (both on the front-end and the back-end) of the system are not fully familiar with and we need to ensure that by optimizing the 4 UCD elements (*Visibility, Accessibility, Legibility* and *Language*), we ease their path to smooth adoption of the system.

The Screenshot **2.1** is for a fairly standard Home Page for a financial advisory:



2.1 Consultium Home Page Screenshot, made on April 9th 2018.

Source: https://www.consultum.com.au/

2.2 Optimizing Design Visibility

Visibility analysis helps developers to construct a mental model of users' needs. Such models help to predict impact of the development on the users' actions and optimize this impact. Important elements of the website/system/application/tool (such as elements that aid easy navigation) should be emphatic. If visibility is optimised successfully, the users will be able to determine quickly, what capabilities the systems do and do not offer.

UCD is the only objective method of achieving optimal visibility as it is hard to empower the users…without the users getting involved and as discussed in *1.5* section of the book, balancing between different user groups is also a tricky task.

Our client is a financial planner that offers many different types of investment. For instance, some of his customers want to invest into a Superfund while others may want assistance with investing into stocks or bonds. There are also investors that do not want to ''put all of their eggs into one basket''. Such investors desire to be able to access complete range of the investment/financial planning options available and work with multiple tools and apps.

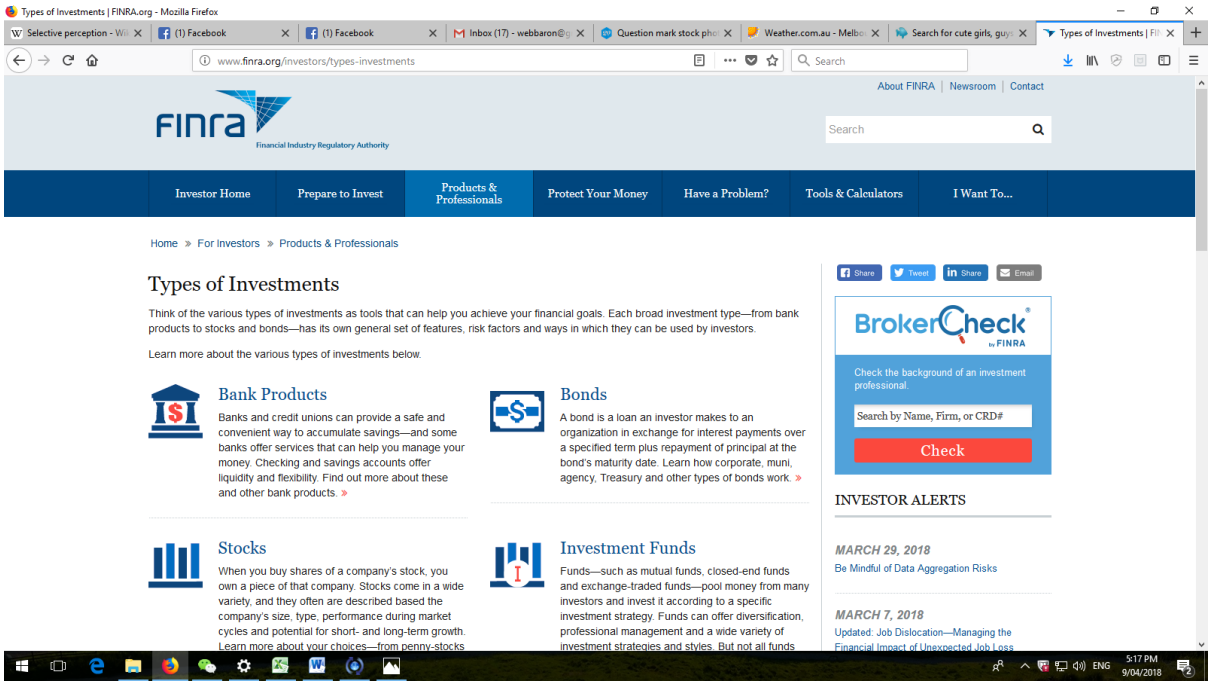
To get started with the *Visibility Optimisation* we need to consider both:

The Parent System

And

The User Feed-Back/Requirements

The Parent System – in case of our client, the Parent System is the starting point of our project. The client already has a fully-functional website that is serving its current clientele. While the current system is in obvious need of re-design, we cannot discard the fact that some of the customers are already used to settings it offers and appreciate a number of the systems' features. We need to note such features to ensure that they are incorporated into our UCD-driven project. Below is an example of a kind of *Parent System* we may have to get our project started with:



2.2 FINRA Investment Page Snapshot, made on April 9th, 2018.

Source: http://www.finra.org/investors/types-investments

The Snapshot **2.2** is an example of how links and descriptions of different investment products can be integrated into a single Web Page. Just like any page/system requiring a re-design, it is far from perfect and contains a number of visibility-related shortcomings. Informational content is randomised throughout the page and the tools/options/services

available to the customers are scattered all of over the page. However, it is nevertheless far better than a ‘’hypothetical mind map’’ that would be based purely on the developers’ perceptions of what to and what not to include.

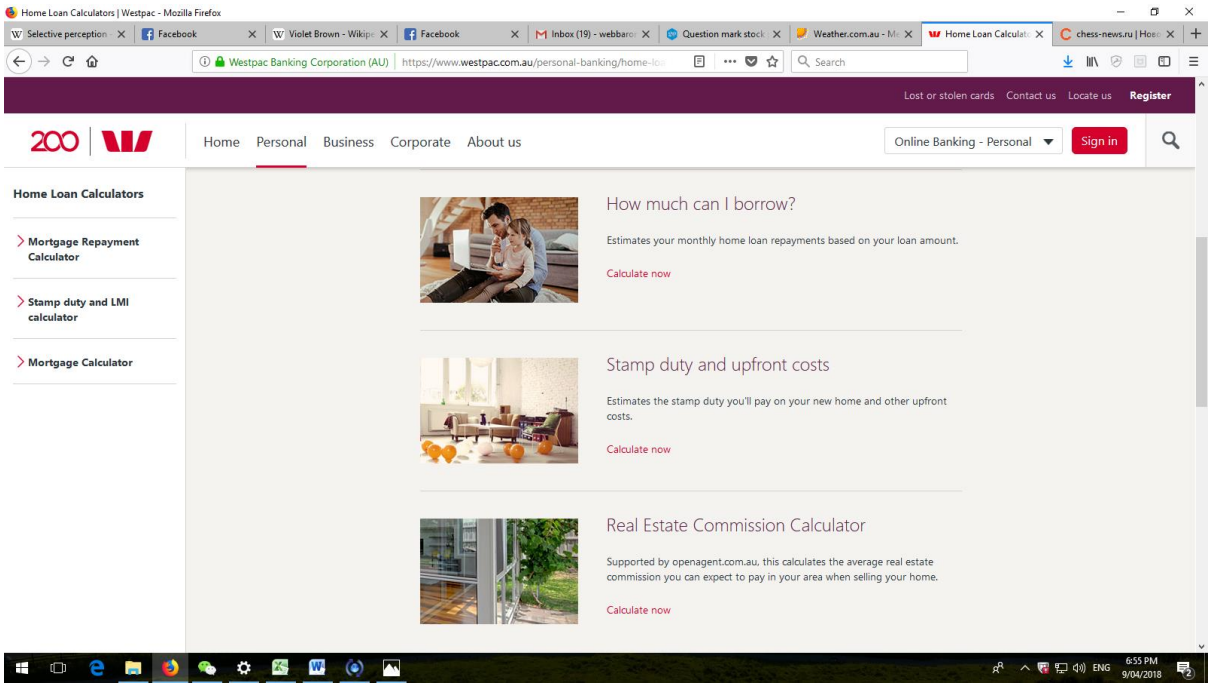
The next step of the Visibility Optimization is to start translating the *User Feed-Back/Requirements* into practical improvements to the Page and the business processes it powers. This task is going to involve reviewing both informational content and functionality of the page.

While informational content (text and images) is relatively is relatively easy to play with and can be adjusted based on the UCD requirements, adjusting functionality can become a significantly more challenging task. With many tools and apps available from a single page, it is important to make sure that they are all integrated with one another where applicable and functionality is not affected by the ongoing customization-driven changes.

Inter-relationships between the tools and apps are also critical to understand. For example, one of the most common tools used by providers of financial services is an online Loan Calculator Tool and/or a Loan Calculation App. In case of our client, the loan calculation function is relevant to some of the services it has to offer (e.g. mortgages) but is of little relevance to some of the other services provided. Therefore, the Loan Calculator needs to be integrated into the overall framework of the development accordingly.

The Screenshot **2.3** below shows a range of Loan Calculation Tools available. Note that in the case of Westpac (the Screenshot is from the Westpac website) all of them are ‘’placed’’ on a single page rather than linked to various elements of the websites’ informational content. What do you think about this arrangement? Can you think of a better way of integrating the

Loan Calculation Tools into the website's content or would you regard the current arrangement as an example of successful UCD Visibility Optimization?



2.3 Westpac Loan Calculators Screenshot, made on April 9th, 2018.

Source: https://www.westpac.com.au/personal-banking/home-loans/calculator/

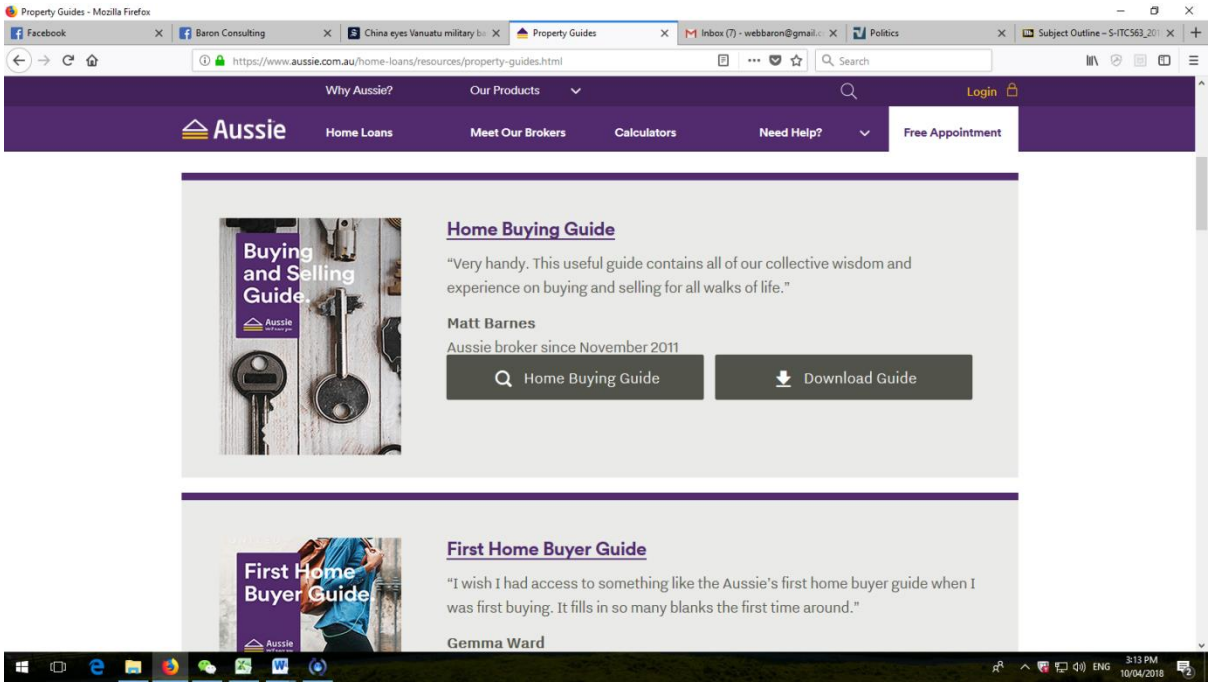
2.3 Addressing Design Accessibility Issues

Efficient accessibility design enables users to find information and services they are after quickly and to navigate the system efficiently, regardless of how large it is and how many different functions it incorporates. This can be done through a variety of UCD-powered patterns (such as clear-cut system maps (e.g. website maps), availability of the system search functions, system directories, clearly identifiable headings, subheadings and section names, linkage between relevant sections, tools and processes etc.).

If we bring *Accessibility* requirements in line with the consumers' vision of the financial service provision expectations, our clients' customers should be able to locate information or a service they are after easily even if they visit the website for the first time. Large providers of financial services have a lot of content to share so the online systems need to be structured very accurately not to allow any informational or functional elements to fall out. Also, navigational elements should be consistent with the overall style of the content and service deliveries. Likewise, this style should be consistent throughout the delivery. It will assist the users in familiarizing themselves with the website faster and developing a sense of ''understanding how it is organized''.

Accessibility can be improved further through the practice of ''Chunking''. *Chunking* is a content management pattern that involves breaking informational content into small pieces that can be organized into a clearly structured order or hierarchy. The ability to skim through these ''chunks'' allows system users to find information and tools they are after by scanning through key features of the informational content rather than browsing through the entire data set.

Chunking can be done in a variety of ways. Content-wise, it can include presenting headings and subheadings in significantly larger fonts and making them stand out by highlighting the key words. One example of simple easy-to-implement Chunking strategy is putting the headings/keywords in **Bold** or *Italic*. Likewise, the key words/links can be presented in CAPITAL LETTERS.



2.4 Aussie Home Loans Property Guide Screenshot, made on April 9th, 2018

Source: https://www.aussie.com.au/home-loans/resources/property-guides.html

The screenshot **2.4** above is of Aussie Home Loans Property Guide Page. As evident from the layout, the company attempted to optimize *Accessibility* of the page sections that are (according to AHL, as it is based on their perception of the user needs) particularly important/most likely to be searched for by the target audience.

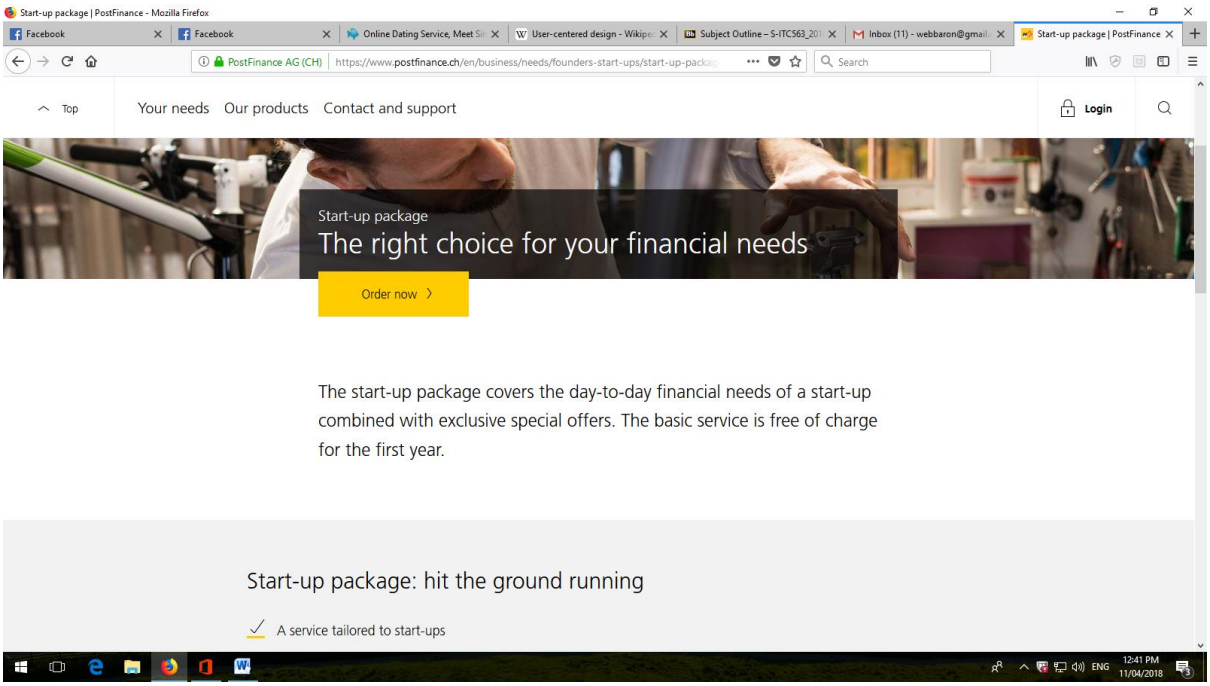
It can be noted that the page layout is a ‘’compromise’’ between functionality and design features. In order to make the page easy to navigate and the content chunks to stand out, the page has been designed in an over-simplistic style. It does not include any fancy animation, content-driven images etc. It focuses exclusively on delivering ‘’critical content’’ to the target audience!

2.4 Legibility and Content Management

Legibility is yet another dimension of the UCD. It focuses on presenting informational content in a ‘’readable’’ format. Specific dimensions of *Legibility* to be considered by UCD developers are: usage of fonts, text size, integration of text and images, highlighting text and links, use of headings and subheadings and establishing amount/degree of detail for the content to be provided.

While UCD requirements vary depending on the target audience, there are some general *Legibility* principles that could be considered and implemented accordingly:

- Text should be easy to read: this should be achieved by the means of finding an appropriate font style, spacing arrangements and margins. For instance, ornamental fonts and text in all capital letters are generally hard to read, but *italics* and **bolding** can be effective when used selectively throughout the webpages. If the text is too large or too small - it is also hard to read. (Screen size of 10-12 pixel sans serif and 12-16 pixel serif tends to be the commonly accepted standard).
- Use of contrasting colours and shades: High figure-ground contrast between text and background often increases *Legibility* by making it easier for the target audience to focus on the text. Another example of using the contrasting successfully is presenting key informational content in dark colours against a light background.



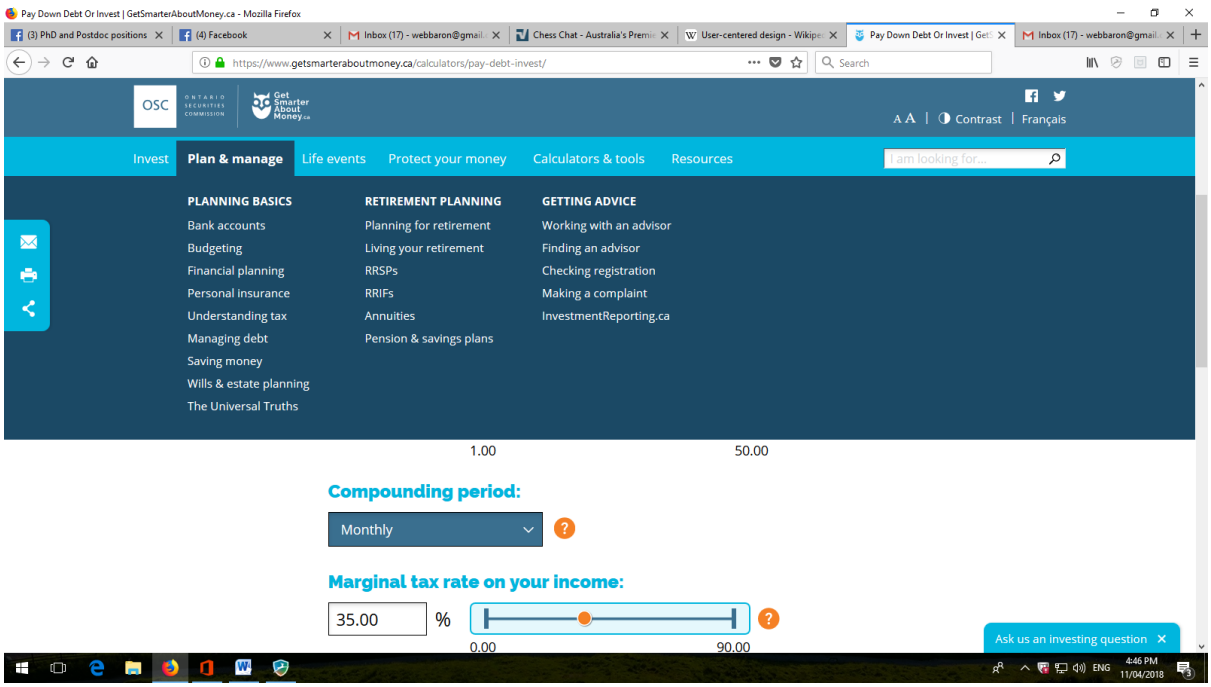
2.5 PostFinance Special Offer Page Snapshot, made on April 11th, 2018.

Source: https://www.postfinance.ch/en/business/needs/founders-start-ups/start-up-package.html

The Snapshot **2.5** above is for a ‘’Special Offer’’ by PostFinance – Swiss company specializing in financial brokerage and money management. As evident from the snapshot, the page does fulfil some of the *Legibility* requirements but not all of them. The special offers do stand out and it increases legibility of the key informational content components. On the other hand, the page does not incorporate an appropriate background, using blank space instead. It certainly does not improve quality of the developers’ work from the UCD perspective. Getting PostFinance’s customers to focus on the content would be easier if a matching background is used instead. For instance, if the text of the offer comes in **Bold** the background could have been more ‘’flashy’’ such as Red or Orange. Adding some relevant images could also be considered. Images capture users’ attention faster than big chunks of text as they are more legible.

Legibility has traditionally been attributed to informational content only (text). However, it can also be applied to improve performance/increase usage of the applications and tools. Finance industry relies on many different tools and apps for service provision and how these tools are presented does matter! Furthermore, in cases of the tools and apps, *Legibility* could be even more critical than with text as some of the customers have little idea about the tools available prior to accessing the system. Once the do get to the tools and apps, they may need clear guidance on the usage and benefits. UCD developers need to understand standard familiarization/usage scenarios and identify strategies for making usage and benefits of these system elements more transparent.

The snapshot **2.6** is for a fairly complex tool: ‘’Pay-Down Debt or Invest Calculator’’ offered by the Ontario (Canada) Securities Commission website. The tool enables clients to analyse their financial situation and obtain a personalized feedback on whether extra money should go into debt repayment or portfolio investments.



2.6 Pay-Down Debt or Invest Calculator Snapshot, made on April 11th, 2018.

Source: https://www.getsmarteraboutmoney.ca/calculators/pay-debt-invest/

Based on the UCD *Legibility* requirements it can be said that the Calculator has been integrated into the overall content of the page/system rather poorly. A number of shortcomings can be identified, namely:

- Such a complex tool needs to be explained and preferably - explained in plain language
- Application of the tool and significance/context of the calculations it produces are not incorporated into the overall content provided
- The tool is not integrated into the overall content of the website/system
- The tool is not linked to any of the business processes that it is supposed to be triggering (e.g. investment products, borrowing products, payment systems etc.) and is therefore not part of a complete customer-centred solution
- Overall design is not appealing and just like with a the previously discussed example (snapshot **2.5**) – no adequate background is provided to increase focus on the tool

Last but not least, *Legibility* of content and functional legibility need to be integrated with one another. Overall, the concept of ‘’integration’’ is very important to UCD. Users are not to be provided with ‘’fragments’’ of a system but with a complete solution that is tailored to their needs. Having discrepancies between content and functionality offerings is very contrast to what adoption of the UCD approach is trying to achieve in the first place.

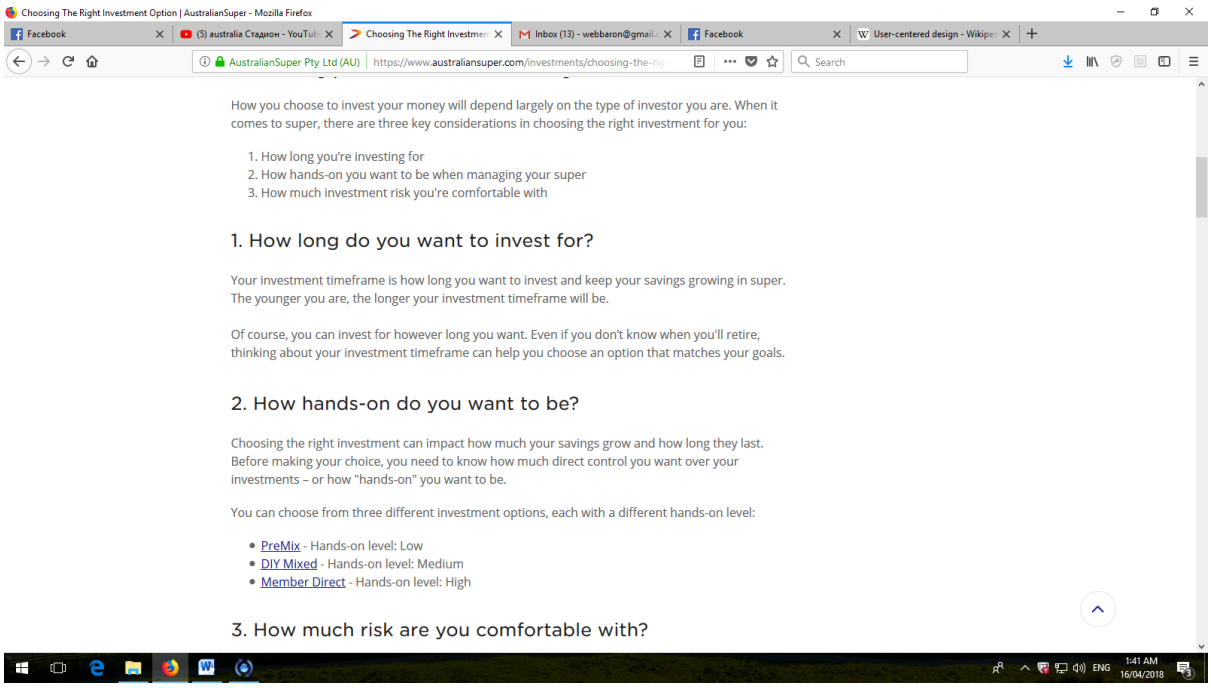
One reason for such discrepancies being not so uncommon is lack of coordination between the content management team (content writers) and the technology management team (app writers, system developers etc.). So the role of UCD is to bring all contributors to

the development process together to ensure that their work is truly collaborative. Likewise, when assessing user experience (front-end users in particular), the focus should be not on asking the target users to rank informational content or functionality alone but to obtain a consolidated perspective. *Not a single dimension of UCD can ever be achieved at the expense of ignoring other dimensions!*

2.5 Tailoring Language Usage to the Target Audience

Language is an essential instrument of the UCD. While importance of using correct words and expressions to reach the target audience is rather obvious, we need to understand what kind of words and expressions to use and how to link different content elements together. Also, even technical functions (e.g. applying for a loan or seeking a quote from a superfund) can only be completed through the use of language-powered commands. Effective usage of the *Language* is linked closely to the rhetorical situation our UCD projects need to service as ‘’power of the language’’ varies depending on the context.

There are some general principles of language usage in UCD. Short sentences, explanations and commands are easier to grasp. The language should be simple and clear. If not for a specific reason, jargon or technical terms should be avoided. Also, simple sentence structure (e.g. avoiding usage of complicated sentences or command names) helps in fragmenting the key points.



2.7 AustralianSuper Superannuation Page Snapshot, made on April 14th, 2018.

Source: https://www.australiansuper.com/investments/choosing-the-right-option

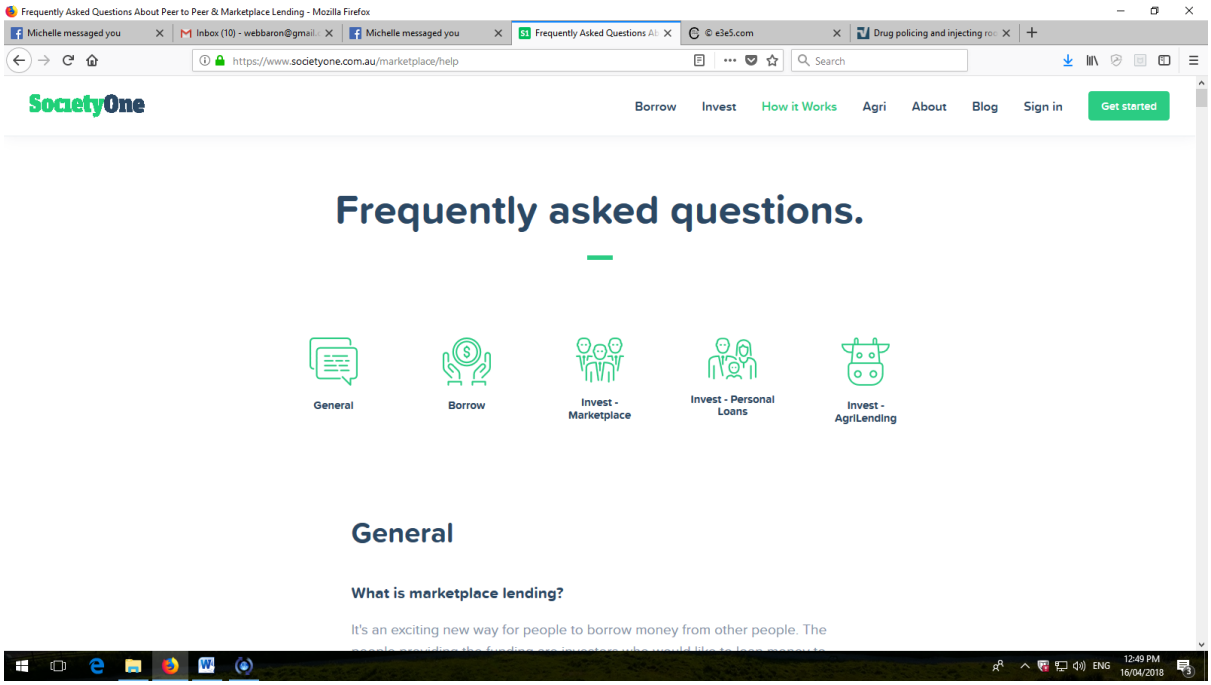
The snapshot **2.7** is for the AustralianSuper Superannuation page. The *Language* used throughout the page is in line with the principles outlined above. As explaining in detail how superannuation works along with pros and cons of all of the super options is not a realistic task to accomplish, UCD needs to focus on providing minimal yet critical information about the services provided and introduce core concepts for each of the options. As evident from the snapshot, the requirement has been met. The front-end users (AustralianSuper customers and potential customers) are given plain language guidance throughout the entire process.

Another aspect of efficient use of *Language* of UCD is creation of User Manuals for both back-end and front-end users. Given ever-increasing use of technology and diversity of the technical developments for service delivery (growing number of apps and tools to incorporate), supporting documentation is essential for assisting users to surf through the systems smoothly.

In industries such as Finance, it is very rarely that the same User Manuals can be shared by the back-end and the front-end users. The front-end users require a plain language guide through the main system functions, capabilities and tasks that these functions and capabilities are set to accomplish. The back-end users may need a far more detailed User Manual that includes expanded overview of the system and its functions to optimize their opportunities to use the features provided.

For the front-end users, a common way of providing user assistance for deployment of online systems is to do so via a dedicated section of the website. It is usually offered via a

‘’Help’’ or ‘’FAQ’’ page. Sometimes (when there is a lot of supporting information to be shared), it could be done in the form of a downloadable User Manual document.

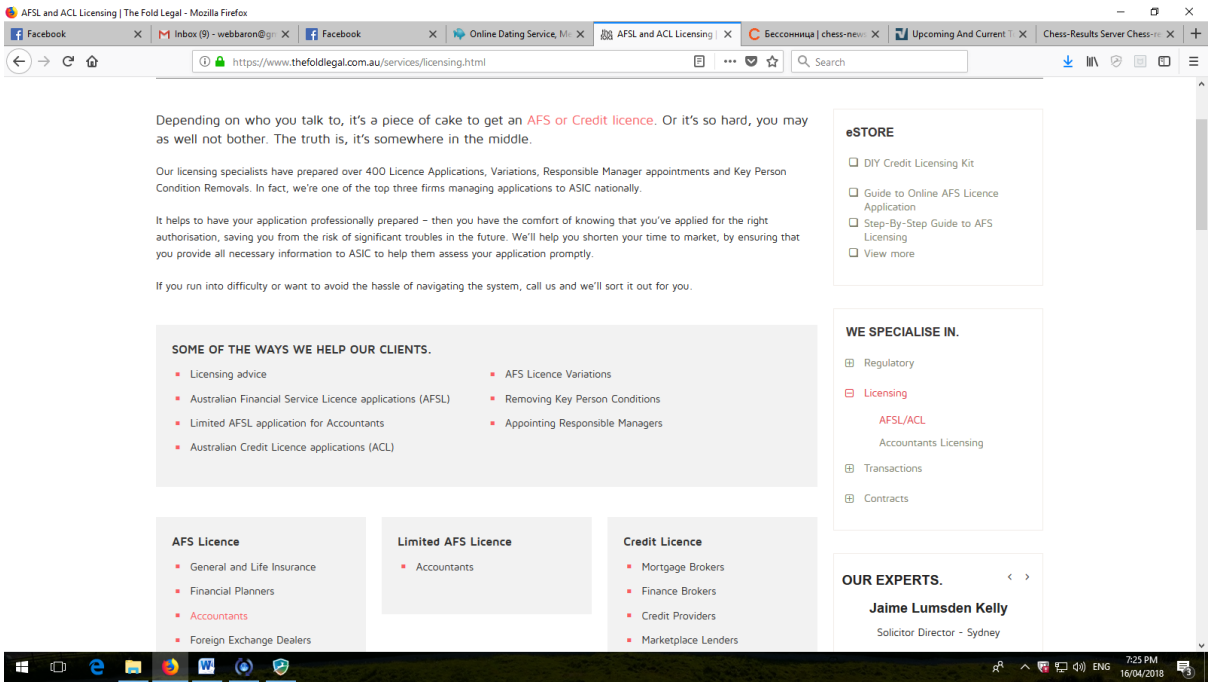


2.8 Society FAQ Page Screenshot, made on April 14th, 2018.

Source: https://www.societyone.com.au/marketplace/help

The Snapshot **2.8** above is an example of a well-organized Help section for the front-end users of SocietyOne – financial services provider. As the organization provides a wide range of services and its service proposition is rather unorthodox, the Help section needs to include not only responses most commonly faced problems but also general explanation of the company’s offerings and comparative descriptions of these offerings. On the other hand, would the company offer standard financial services rather than unconventional ones, service descriptions would not be included into the Help section (it would still be provided on other pages of the website though). This is an example of how the UCD approach is instrumental in making the development (Help section of the website) tailored to the users’ needs!

Language management for servicing back-end users involves not only preparation of longer/more comprehensive documents but may also require reviewing some of the ‘’traditional’’ approaches to the UCD. We should note that ever rule/principle has exceptions and *Language* selection for back-end users is often home to such exceptions. For example, as we already discussed above, it is generally good to use simple language and to exclude jargon. However, back-end user groups may include segments that are using jargon on regular basis and find the jargon words most appropriate to describe certain system elements. But how are we going to know this? Well, as we discussed in **Module 1**, UCD involves incorporating user requirements and feedback into the development process!



2.9 Fold Legal Financial Advisors' User Manual Snapshot, made on April 15th, 2018.

Source: https://www.thefoldlegal.com.au/services/licensing.html

The Snapshot **2.9** is for an online User Manual provided for Financial Advisors/Planners (back-end users). Do note that accessing complete back-end information for unregistered users (such as us since we are not members of the Fold Legal team) is not possible so we

cannot get to see the ‘’complete picture’’. However, the documents and tools that are available for ‘’open viewing’’ are sufficient to show scope of the user manual and reveal the amount of work put into development of the user portal.

Just like with the other aspects of UCD – the key is to be able to balance between all of the user groups so all of their requirements are catered for!

2.6 Module 2 Summary

In this Module, we have used the Case of Finance Industry to learn:

- How to Identify User-Centred Design Elements
- Optimizing Visibility of the User-Centred Design
- Addressing User-Centred Design Accessibility Issues
- Managing User-Centred Legibility and Content

- Using User-Centred Design to Tailor Language Usage to the Target Audience

2.7 Module2 Review Questions

1. What are the 4 dimensions of User-Centred Design?
2. What are the Key Elements of Visibility Optimization
3. Consider the Screenshot **2.3**. Apply User-Centred Design Principles to improve Visibility.
4. What are some of the strategies and patterns for improving Design Accessibility? Discuss with financial industry-related examples.
5. What are the strategies for involving system users into improving Legibility of content? Discuss with examples.
6. What are the differences (if any) in effective Language usage when addressing needs of back-end and front-end users in the finance industry? Discuss with examples.

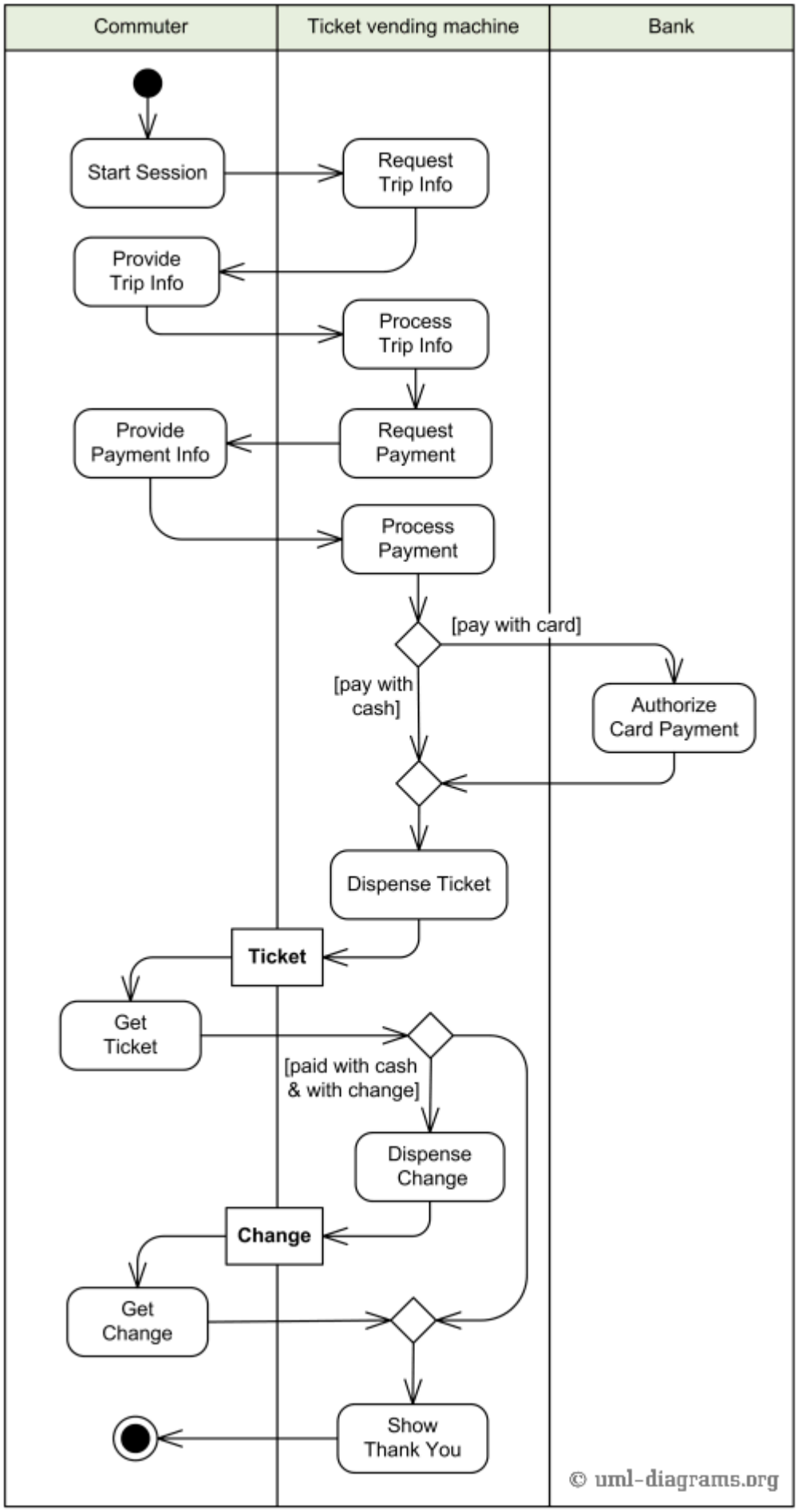
Module 3

Documenting and Presenting User-Centred Designs: Unified Modelling Language

3.1 Unified Modelling Language (UML) Explained

UML (Unified Modeling Language) is a graphical language that can be used for specifying, constructing and documenting system/project requirements in a concise manner. Consequently, UML can be used effectively for the UCD projects. From the UCD perspective, the main "attraction" of using UML is its ability to visualize and document even the most complex systems graphically. Working with a set of UML diagrams is much easier and faster than reading through lengthy reports. Furthermore, the diagrams present data more accurately and are less open to misinterpretation as opposed to non-graphical ''descriptive'' documents.

Many of the UCD projects require use of UML to one extent or the other. It is true, that UML-powered diagrams are often incapable to provide developers with an "ultimate" solution that illustrates each and every facet of the project in detail. A diagram (even if it is accurate and detailed) can provide an outline of a system but without sufficient annotations and/or step-by-step implementation plan – the developers will not be able to get started with the actual construction of the system.


Commuter
Ticket vending machine
Bank
Start Session
Request Trip Info
Provide Trip Info
Process Trip Info
Provide Payment Info
Request Payment
Process Payment
[pay with card]
[pay with cash]
Authorize Card Payment
Dispense Ticket
Ticket
Get Ticket
[paid with cash & with change]
Dispense Change
Change
Get Change
Show Thank You
© uml-diagrams.org

3.1 Sample UML Diagram, Ticket Vending Machine, accessed on April 17th, 2018

Source: https://www.uml-diagrams.org/ticket-vending-machine-activity-diagram-example.html

The diagram **3.1** above is a very basic example of how UML can be used to illustrate key components of the system as well as interrelationship between these components. This particular diagram is an *Activity Diagram* – one of many types of diagrams that can be produced using UML.

Another useful feature that can be of benefit to UCD developers is global acceptance of the UML standards. UML has already been in use for many years and is recognized as a standard for many development projects (including the UCD ones) so all members of the development teams will find it easy to understand the diagrams.

To sum up, key benefits of using UML as a method of documenting UCD project data are clarity and simplicity that the data can be presented with. Use of the UML diagrams can ensure that:

- The Data is easy to read and will be interpreted in a consistent manner
- There are going to be no misunderstandings between members of the UCD development teams due to a lack of common interpretation of the diagrams
- The UCD project data is going to be easier to manage and document

No data documentation system is spotless, so challenges and risks related to use of UML in UCD projects also need to be considered. Given graphical nature of the UML, even one relatively "insignificant" documentation error can place the entire project into a danger zone. The developers will be ''clear'' what they need to get done …while diverging in the wrong direction! On a positive note, if the UML diagrams are supported by detailed annotations/commentary, an experienced UCD developer should be able to spot the

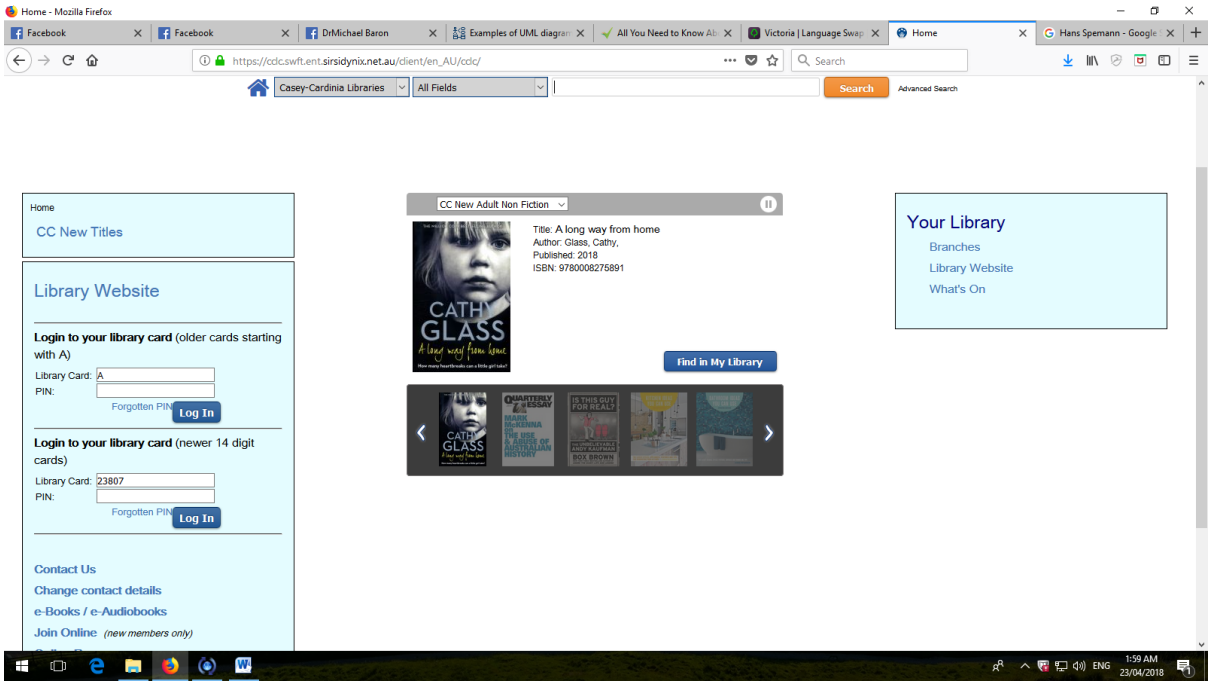
discrepancy between the diagram and the annotations. Consequently he will be able to alarm the development team to ensure that should there be a scope for an error, all the inaccuracies are going to get rectified before any irreversible mistakes are made.

So now that the key concepts behind usage of UML for the UCD developments have been discussed, it is time for our next assignment…as we got a new client waiting! ☺

Our client is Public Library in Regional Victoria. The library has been running for many years and is very popular with the locals as it is the only library available in the area. The library has a limited stock of books, newspapers and journals that are made available to all of the interested users.

Our client has just received a government grant to upgrade its service delivery by the means of effective implementation of the emerging technologies. The client wants you to identify the best strategy for the technology implementation as well as specific services that are to be delivered. Furthermore, we need to ensure that the service delivery is going to be optimal for all of the users groups. Some of the ideas that are currently under consideration (and more original ideas/suggestions are always welcome) are: digitalization of the items that are particular popular with the readership, updates to the online reservation system to personalize the e-booking process, creation of online reading clubs and study groups and development of a customer support portal.

Your job is to examine the current facilities and systems available and to re-engineer these facilities through the use of UCD. UML diagrams will be used to prepare and document your proposals as well as to guide you throughout the upgrade process.



3.2 Casey Libraries Item Search Page Snapshot, made on April 17th, 2018.

Source: https://cclc.swft.ent.sirsidynix.net.au/client/en_AU/cclc/

Majority of the UML Diagrams can be classified under 1 of the 3 major categories:

- *Behavioral Diagrams*
- *Structural Diagrams*
- *Interaction Diagrams*

Behavioral Diagrams illustrate what activities and actions should be incorporated into the system developed. This category of the diagrams is particularly convenient to use for UCD projects as the diagrams are very straightforward and can be easily understood by all of the stakeholders. Examples of how the *Behavioural Diagrams* can be used for upgrading Library

Services can include both simple (e.g. activities involved in checking number of books issued to a patron or activities involved in seeing if a particular item is available for borrowing) and fairly complex developments such as the entire process of handling the digital archive.

While *Behavioural Diagrams* illustrate activities and actions, *Structural Diagrams* illustrate components and processes that must be included into the systems developed. For example, a *Structural Diagram* can demonstrate how the library membership application process is split up into a range of components as well as relationships and dependences between these components. From the UCD perspective, this can be a starting point for identifying ways of simplifying the system usage path for both front and back-end users by the means of decreasing the number of dependences, removing components that are not critical for the system's performance from the business process and most importantly – getting a clear picture about the requirements.

Interaction Diagrams is the most complex of the 3 categories. *The Interaction Diagrams* illustrate flow of control, data and communications between the system components. For example, they can be used to demonstrate how the components communicate with one another as well as the sequence of messages. The reason they may get complex is variety of the system responses/tasks that need to be considered. For example, an *Interaction Diagram* for a library patron accessing the only database to search for the books/journals – the sequence of messages and structure of the process may depend upon whether he is searching for digital resources or hard copies.

3.2 Behavioral Diagrams

Behavioral Diagrams are the most commonly used diagrams in the UCD projects. The 6most common types of the *Behavioral Diagrams* are:

- Activity Diagrams
- Use Case Diagrams
- Timing Diagrams
- State Machine Diagrams
- Communication Diagrams
- Sequence Diagrams

While it is well beyond the scope of this book to discuss all of them in detail, let's consider how we could use different types of *Behavioral Diagrams* in our UCD Library System Project:

Activity Diagrams are most common to use as they describe the flow of activities and actions. The diagrams can be either sequential or parallel, with the sequential diagrams illustrating the order in which the activities take place and parallel including activities where there is not interdependency between the tasks and/or functions. For example, when we develop a process for online reservations of books, we can choose whether we want the process to be a step-by-step one (and this is probably best option of all, given the need to keep things simple at the front-end of the UCD systems) or whether we want parallel tasks to be included as well (to accommodate some more sophisticated search and booking functions for the advanced users).

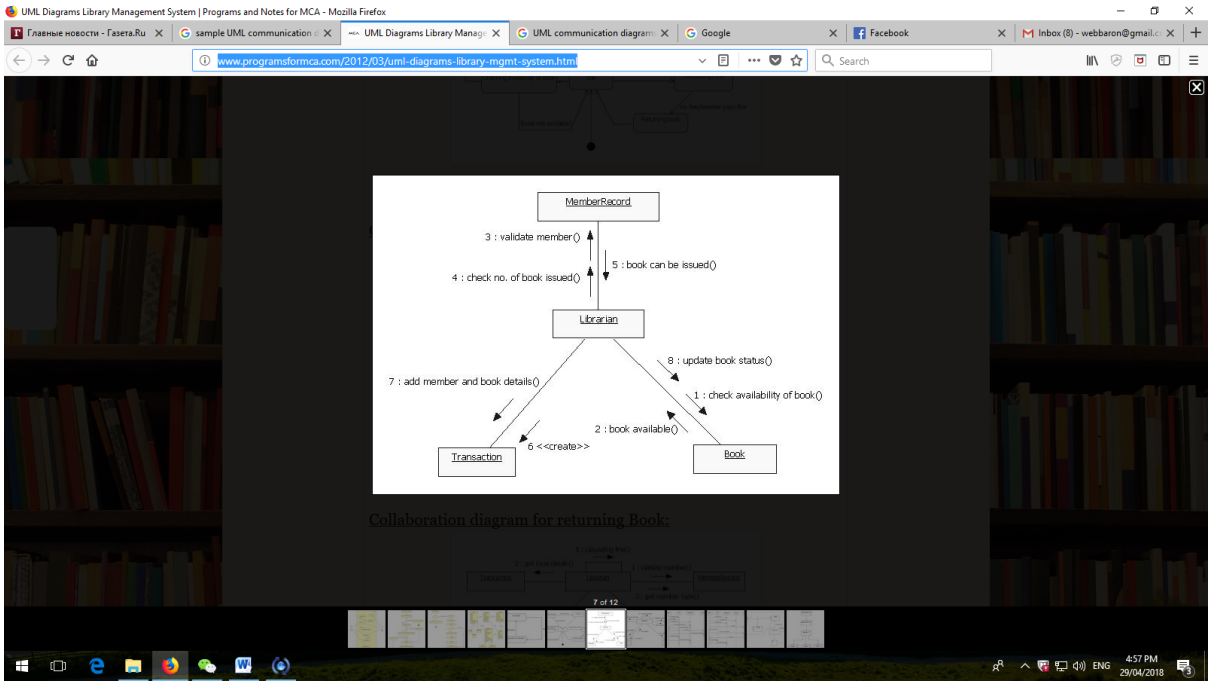
Use Case Diagrams are usually less detailed when compared with the *Activity Diagrams* as they are used to document main requirements/components of a system. With library systems, the requirements included may focus on: transaction processing, feedback turnaround times, recording unreturned items, having library collections updated etc.

Timing Diagrams illustrate how system performance changes over a period of time. In a library environment, some of the systems and processes in use are legacy ones. Over period of time, performance of such systems is likely to become less effective and the *Timing Diagrams* can show these changes. From the UCD perspective, performance declines over the time are indicators that that the systems are no longer having the user satisfaction focus.

State Machine Diagrams illustrate different states of a system as these states are changing. One example that is commonly used to explain how the diagrams work is that of a game of chess: playing chess involves White and Black players to alternate moves. After every single move, position on the board is changing and so can assessment of the position. The position may be winning for White but after a poor move, it may immediately become winning for Black.

With the library systems, *State Machine Diagrams* can be used for a number of UCD processes. One of the processes where it could be useful is *Customization*. State Machine Diagrams will assist to determine how different customization activities impact performance of the system.

Communication Diagrams focus on relationships between different system elements (not to be confused with the *Sequence Diagrams* as there are indeed some common characteristics between the 2 categories) *Communication Diagrams* link these system elements through the messages exchanged.



3.3 Communication Diagram for Issuing a Book Screenshot, made on April 27th, 2018

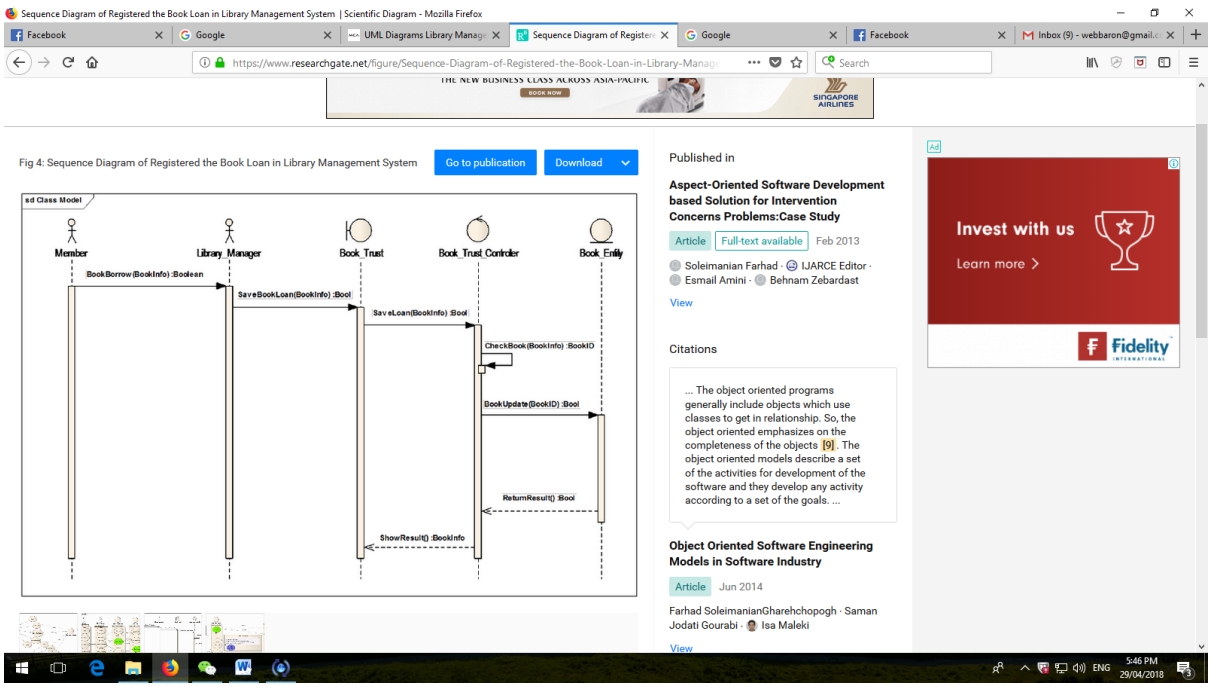
Source: http://www.programsformca.com/2012/03/uml-diagrams-library-mgmt-system.html

The diagram **3.3** above is an example of a very basic *Communication Diagram* for a Library System. Note that if you are to do further study/research on *Communication Diagrams* you may discover that sometimes they are being referred to as *Collaboration Diagrams* in most cases – it is just a different name for the same category of the UML Diagrams.

In our UCD Library project, we will need to use the *Communication Diagrams* to show linkage between different facets of the systems we are going to design. One of the key aspects of the re-engineering that we are undertaking is to provide customers with more

options throughout their ‘’library journey’’ on every step of the way. This can only be done if relationship between these options is transparent.

The *Sequence Diagrams* are likely to offer same/similar type of information as the *Communication Diagrams*. However, there is a critical difference between these 2 types of Diagrams: Sequence Diagrams emphasize *time and order* of the events.



3.4 Sequence Diagram for Library Management System Screenshot, made on April 27th, 2018.

Source: https://www.researchgate.net/figure/Sequence-Diagram-of-Registered-the-Book-Loan-in-Library-Management-System_fig3_262640433

The screenshot, *3.4* shows a very basic *Sequence Diagram* for borrowing a book.

3.3 Structure Diagrams

The types of Structure Diagrams that we need to consider are: *Class Diagrams, Object Diagrams, Component Diagrams, Composite Structure Diagrams, Package Diagrams, and Deployment Diagrams.* Below are brief definitions for each of the types with a Library-system related example:

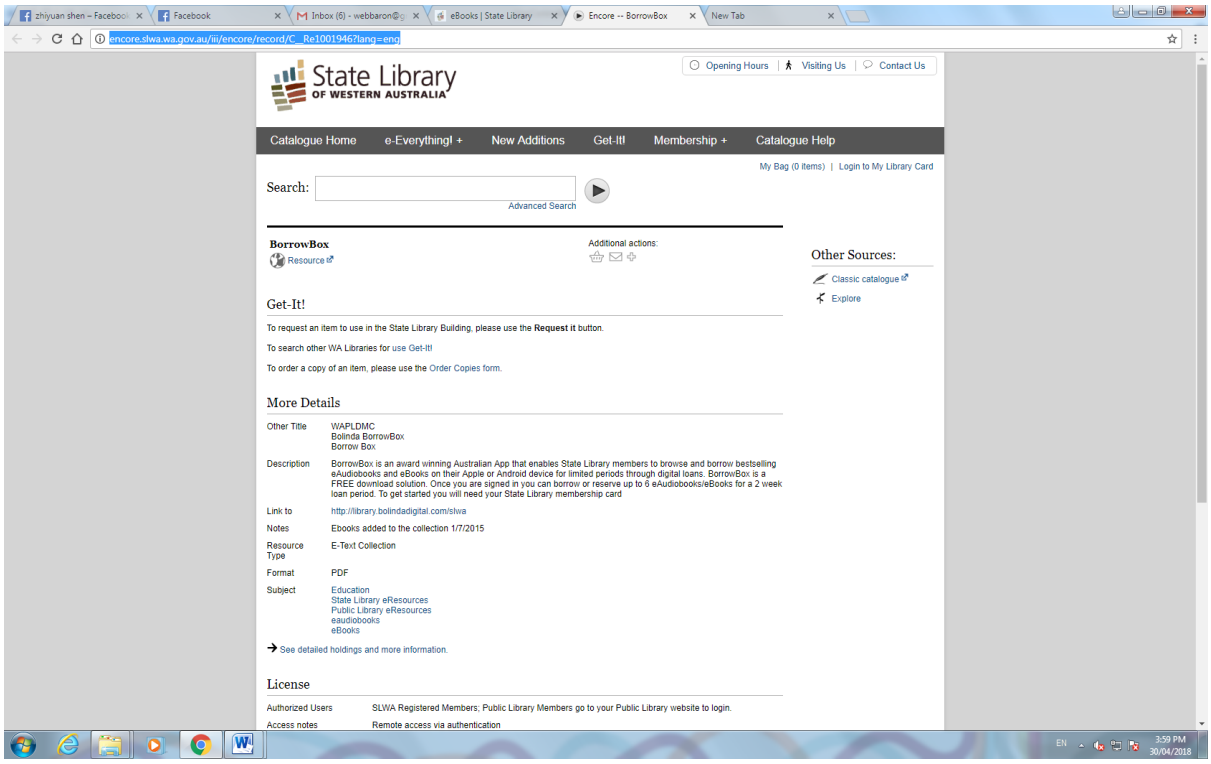
Class Diagrams – are static structure diagrams that demonstrate different classes of a system. Each of the classes is identified via provision of 3 compartments: The top compartment includes the name of the class (e.g. library customer ID search). The middle compartment includes key attributes of the class (e.g. what the library customer ID search does) and the bottom compartment includes functions that the class can fulfil. A diagram for a Customer ID search would not be complete with any of these 3 compartments missing!

Object Diagrams- are graphic representations of instances, including objects and data values. When compared to the *Class Diagrams* - they incorporate more detailed specifications and are therefore more concrete. From our perspective, we may consider using the *Object Diagrams* for the task of testing of our UCD systems as many of the system functionality requirements (e.g. ability to perform Library Customer ID search) need extensive testing to avoid errors and discrepancies. Furthermore, the users' needs could be best addressed by providing a range of potential options and testing them all out to compare.

Component Diagrams – show different components of a system as well as relationships (how they are connected to one another) between those components. That makes the *Components Diagrams* particularly effective when illustrating technical aspects of the system as there is often a range of technologies used to connect different components with one another.

There are 2 main reasons why we can consider using *Components Diagrams* in our UCD-powered Library System development project. Firstly, we are going to deal with a number of technologies and need to make sure that the technologies used are well-integrated. This cannot be done without identifying ''what should be connected to what''. Secondly, we are

not developing a library system from scratch but re-engineering an existing system. This system already has a technical infrastructure that we need to consider and use as a base for further developments as component-based approach to the UCD should be carried out with understanding that current technologies and tools could be reused (e.g. to cut down cost of the development) and/or that these components could be replaced by some other "equivalent" technologies or tools, if needed.



3.5 WA State Library BorrowBox App Screenshot, made on April 28th, 2018.

Source: http://encore.slwa.wa.gov.au/iii/encore/record/C__Re1001946?lang=eng

The screenshot **3.5** is for BorrowBox - an award winning Australian App that enables WA State Library members to browse and borrow eAudiobooks and eBooks via Apple or Android devices for set time periods through digital loans. BorrowBox is free to download and once the library patrons are signed in, they can borrow or reserve up to 6 eAudiobooks or eBooks for a up to 2 week loan period. To get started with the app, the patrons require WA

State Library membership card and a password. Initially, their surname will be used as a ‘’default password.

UCD-powered re-engineering/development for tools such as BorrowBox can utilize *Component Diagrams* to ensure that this fairly complex app is both working flawlessly and is user-friendly!

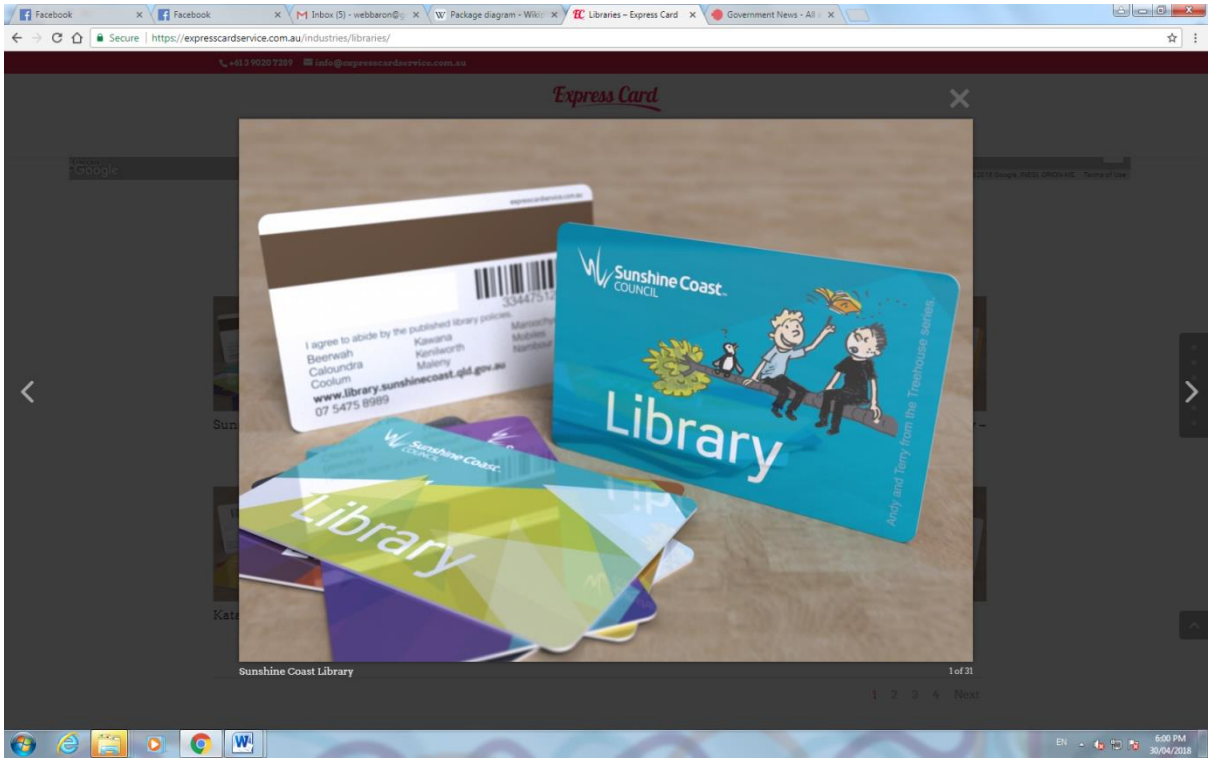
Composite Structure Diagrams- are static diagrams that are used to illustrate not only the general infrastructure of a system, but also respective parts and elements of these components. It enables developers to understand performance requirements of a system in greater detail. A single *Composite Structure Diagram* could be made in an ‘’all-inclusive’’ format and incorporate not only relationships between the system components but also the so-called ‘’external relationships’’ – to illustrate how the system is connected to other systems.

With the Library Systems, the UCD approach involves creating a greater range of options for queries and relationships for the Search Databases. We could use the *Composite Structure Diagrams* to document the interactions/combinations of components required to make these queries run in an optimal way.

Package Diagrams- is arguably one of the most complex categories of the UML diagrams. The *Package Diagrams* are used to illustrate dependencies between different elements of the system. Given the growing number of dependencies in a contemporary business environment (including external dependencies e.g. Extranet-related), it is essential to identify all of the dependencies within a system accurately so failure/underperformance of a particular component does not result in consequent failure of the dependant components.

One technology initiative where *Package Diagrams* could be used effectively for the UCD system development is Shared Library Smartcard technology (as shown on the

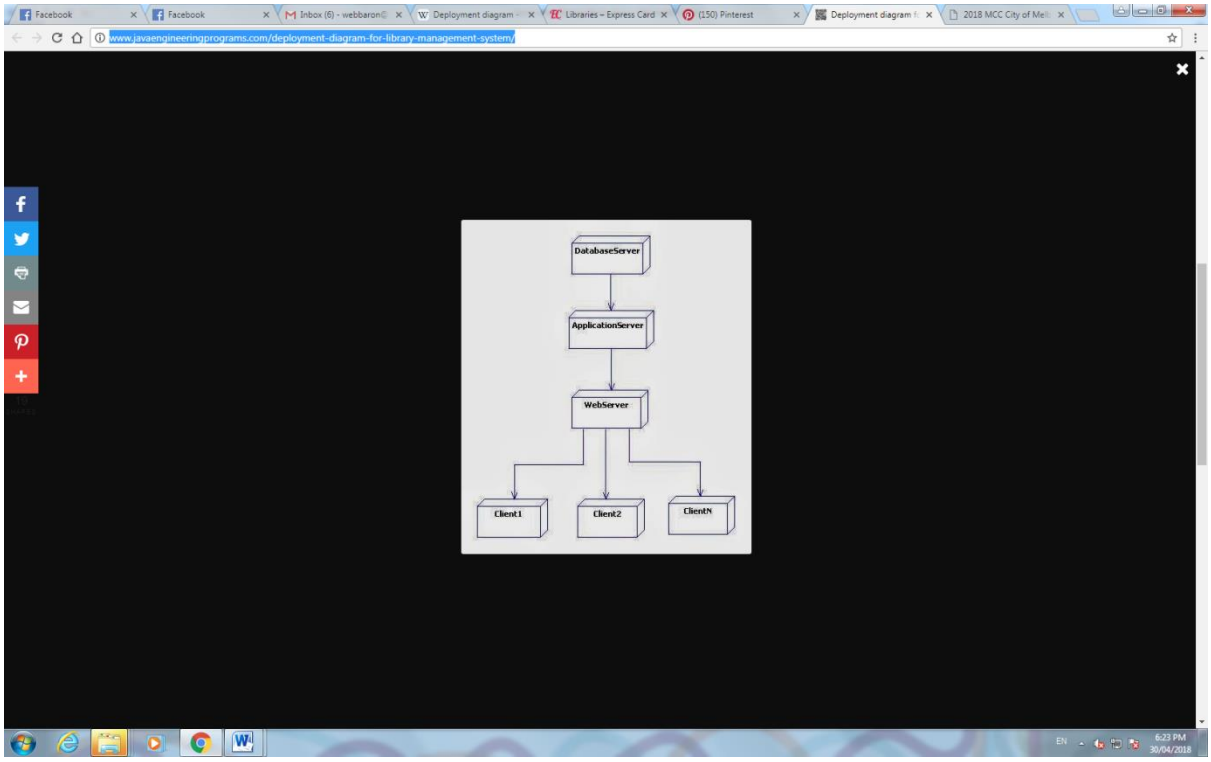
screenshot **3.6**) that enables library staff and patrons to tap into multiple facilities with a single device and it could be a very efficient solution both in terms for convenience and cost savings.



3.6. Express Card Service for Libraries Screenshot, made on April 29th, 2018.

Source: https://expresscardservice.com.au/industries/libraries/

Deployment Diagrams – can be used to illustrate physical deployment of a system.



3.7 Deployment Diagram for a Library Network Snapshot, made on April 29th, 2018.

Source: http://www.javaengineeringprograms.com/deployment-diagram-for-library-management-system/

The **Snapshot 3.7** illustrates a very basic (obviously our UCD project will need to use a way more complex one) Library Network and how different components of the network are deployed within. Our project is likely to require *Deployment Diagrams* of significantly greater complexity. For instance, the network type selected is likely to be an Extranet.

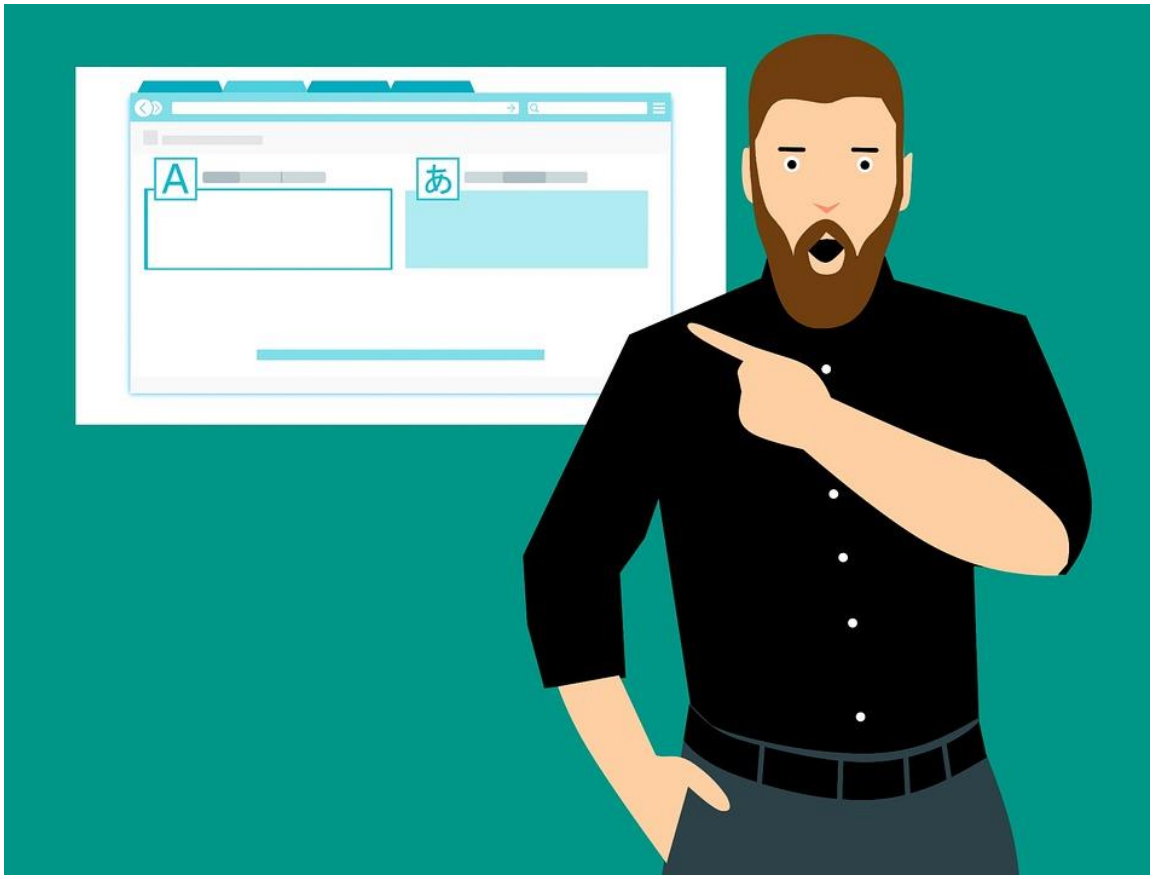
3.4 Interaction Diagrams

Interaction Diagrams is a relatively new diagram category (subset of the *Behavior Diagrams* that has only recently been ‘’elevated’’ to a stand-alone category due to the growth in both scope and significance.

Interaction Diagrams (as evident from the name) visualize sequence of activities and communications between different system components. They do have some similarities with *Activity Diagrams* as both categories look into sequence of the activities and *Communication Diagrams* as they also look into communications between the components. However, it is ability to combine both that adds to complexity (and importance!) of the *Interaction Diagrams* and makes them so valuable in the UCD Projects. So in a nutshell, an *Interaction Diagram* can be the ‘’icing’’ on the UML cake.

In the introductory section of the Module 3, an interaction example of a library patron accessing the database to search for the books/journals – with the sequence of messages and structure of the process depending upon whether he is searching for digital resources or hard copies of the publications has been given. If we consider the challenge of developing the search process from the UCD perspective, there are several factors to be considered and the *Interaction Diagrams* may be able to address at least some of these factors better than other diagram categories. The User Focus can be optimized by considering issues such as: a) variety of ‘’walk-through’’ scenarios for the patrons to undertake b) fulfilment issues from the provider (library staff) end to ensure that the orders recorded can be complied with c) providing greater number of booking options and simplifying these options to make them more user friendly d) identify items that have the greatest demand for, so they could be digitalized first etc.

3.5 Module 3 Summary



In this Module, we have used the Case of Library Systems to learn:

- UML Principles
- 3 Categories of the UML Diagrams
- Practical Applications of UML in UCD

3.6 Module 3 Review Questions

1. What is UML and why it can be useful for UCD?
2. What are the 3 Categories of the UML?
3. Research the Applications and Tools Available for Creation of the UML Diagrams. In Your View, Which Applications and Tools are of Greatest Benefit to the UCD Developers? Which Applications and Tools Are User-Friendly From the New Users' Perspective?

HAVE A HAPPY CAREER AHEAD!


GROW
LEARN
EXPLORE