**Department of Computer Science and Creative Technologies (UWE)**

**MSc Software Engineering**

**Research Proposal**

Proposed title: To improve the usability of the Bladed application through UI adaptations at runtime with machine learning

Supervisor: Dr. Jim Smith

# Background

I work as a Software Engineer at DNV GL (office in Temple Quay, Bristol). DNV GL is the world’s leading certification and advisory company in the renewables sector. My development team of 12 produces a family of products:

* Bladed – the industry standard wind turbine simulation package
* WaveDyn – the world’s first wave energy device simulation package
* Tidal Bladed – the world’s first tidal turbine simulation package

My daily work involves UI design and implementation, as well as interaction with the simulation code. We are currently in the middle of a re-implementation of an old UI, in parallel with adding new UI to support new capabilities. All new UI is developed using Microsoft .Net (WPF and C#). I have full access to the source code and the support of my manager to undertake the proposed project.

# Research Context

Bladed is the world leading wind turbine simulation package, used by hundreds of engineers worldwide. Users specify then run the simulation. There are many input parameters of diverse and sometimes complex data types.

Users have individual habits and preferences as to how they populate the data model. But there are common characteristics between users:

* Common to re-use previously inputted data, changing a subset.
* Common to start from scratch with an empty data model.
* Common to populate data over several user sessions.
* Very common to make revisions or corrections.

User interaction habits may change over time:

* Type of work a user does may change over time - different projects call for different interactions.
* User behaviour may change as the user develops his understanding of the data model and develops his process.

# Research Opportunity

Analysts often repeat the same interactions many times. Any reductions in task duration would benefit the user. Examples of possible ways to reduce duration of the data input task:

* Find the wanted widget more quickly
* Thereafter input the data to the widget more quickly
* Input multiple data in fewer interactions
* Predict the values the user will input and preclude the user needing to input anything at all

Improvements in task duration can be described as an increase in usability. Usability is defined as the ease of use of a human-made object.

# Literature Review

(Akiki, Bandara et al. 2014) provides a review of many different adaptive UI frameworks. The need for adaptive UI is explained:

*“The ‘one design fits all’ approach is unable to accommodate all the cases of variability in the context of use, in many cases leading to a diminished user experience. Building multiple UIs for the same functionality due to context variability is difficult since the scope of variability cannot be completely known at design time and there is a high cost incurred by manually developing multiple versions of the UI. Adaptive UIs have been promoted as a solution for context variability due to their ability to automatically adapt to the context of use at runtime.”*

(Norcio, Stanley 1989) suggests that the concept of an adaptive UI is straightforward: “*the interface should adapt to the user; rather than the user adapting to the system*”. They also note that despite this simple concept, adaptive UIs are anything but simple to implement.

(Akiki, Bandara et al. 2014) suggest that there are two stimuli that the UI could perform adaptations in response to:

1. Context – the UI would respond to changes in the environment of the user and device.
2. User interaction – the UI would respond to changes in user behaviour

The authors go on to suggest there are two ways in which a UI could adapt dynamically:

1. Self-configuration – features of the UI would be hidden in an attempt to present the minimal UI needed for the current task. Environmental constraints could be managed by “*graceful degradation*” of available features.
2. Self-optimization – component layout or presentation would be modified dynamically to optimize usability for a given sub-task.

(Paymans, Lindenberg et al. 2004) states “unpredictable autonomous interface adaptations can easily reduce a system’s usability”. This study tested whether this negative effect of adaptive behaviour could be reduced, by attempting to help users build adequate mental models of such systems. In this short, focussed paper a qualitative study was performed, concluding that an adaptive UI can provide an improved usability without necessarily providing improved learnability or an improved mental model.

(Blumendorf, Lehmann et al. 2010) describes the architecture for a model that will be able to adapt the UI to support the current task(s) that the user is performing. The author claims that using a model-based approach can reduce the complexity of system and user interface design to a machine-processable level. UI adaptation can be treated as an optimization problem. Build a task or process model and keep track of the interactions of the user on this level. Changes to the task model can alter the workflow of the application. Recent focus has been on utilising models at runtime. Possibility to adapt to multiple and changing contexts of use. Design decisions can become explicitly visible to the user, and design rationale is available to help learnability.

The author suggests that the UI system “*can either push information into the model or pull information out of the model. … UI adaptation can be expressed as evolution between the current state of the model and a new state, through a safe migration path*”

The paper goes on to describe the architecture for a model that will be able to adapt the UI to support the current task(s) that the user is performing. Examples are given of shopping list items automatically transferring from desktop PC screen to a mobile device screen when the mobile device comes into proximity of the desktop PC. Another example is given where UI elements increase in size as the user moves away from the display. This is perhaps slightly different to the specific aim I had in mind of adapting a very narrow set of UI elements based on previous data that had been gathered about the user’s interaction to that UI. This paper is about adapting the UI in the way that we would normally hard code. I am looking at a way of optimising the layout within some fairly narrow constraints of a fairly local part of the UI.

In (Nazemi, Stab et al. 2010) the authors describe an increment to an existing technique for determining what task the user is performing based on analysis of user interactions with the application. In particular, the user should not have to actively input information about what they are doing to the application. The authors identify a possible source for capturing user information in an implicit way as “*the sequence of interaction events that occurs as natural consequence of the usage of the user interface*”, which can be captured easily. Thereafter, automatic analytic methods are required to extract user information from the captured data. 4 existing algorithms are listed for machine learning of user intent:

1. Bayesian Networks
2. Probabilistic Relational Models
3. Markov Models
4. LEV- and KO-Algorithm

Google Now is an example of a successful modern adaptive application. It adapts to the following aspects of context:

* Daily routine
* Location
* Travelling patterns
* Search history
* Personal information held in the user’s Google Account

User context data is sent back to Google for processing. The user model is all calculated on the server side, and data sent back to the device for UI adaptation.

# Research Questions

The primary question of this research is: **Can the usability of the Bladed application be improved through UI adaptations at runtime with machine learning?** The hypothesis is yes, based on the literature review. The questions that follow on are related to: *how can this be tested*?

The following general questions will need to be answered to some extent in order to test the hypothesis:

1. How can the usability of an application be measured?
2. How can conclusions be drawn from user interaction data?
3. How can UI elements be adapted to improve usability?
4. How will users respond to UI adaptation?

The following questions are specific to the Bladed application, but will yield conclusions applicable to other applications with similar user interaction paradigm:

1. How can user interaction data be gathered in the Bladed application?
2. What UI adaptations are possible in the Bladed application?
3. What kind of UI adaptations are the most useful for the Bladed application?
4. What kind of user interaction data is most useful for the Bladed application, given conclusions from 6 and 7?

Finally

1. Given the best type of user interaction data is gathered and the best type of UI adaptation is made, is the usability of the Bladed application improved?

# Why is this research needed?

Bladed is a complex piece of software, among countless others, and gets used heavily by many different users. Any improvement in usability will increase the value of the product. There is only so much that can be done with static UI adaptation, based on user interaction behaviour as perceived by the development team to be representative of real users. The development team rarely gets feedback from real users in a competitive industrial environment, who are cagey about their behaviour, concerned about protecting their intellectual property, and very busy. Realistically, the only way we could gather significant amounts of data would be through instrumentation of the user interface.

Automatic UI reconfiguration based on user interactions is not a common feature in applications. I estimate that the typical Bladed user would initially be sceptical about the potential efficacy of such a system. Users will probable feel uneasy with changes to a UI they are familiar with, unless there is a clear benefit. Proper research could give confidence in this. In order to even consider usability improvement through UI adaptation, the ability to gather user interaction data and analyse it must be demonstrated.

Much academic research has been performed on the generalised algorithms that may be used to enable machine learning and potentially UI adaptation. Arguably the machine learning algorithms are of most interest to academic research, but my knowledge of the domain and the amount of time available for this research project would not be sufficient for such a topic. Instead I propose to take an existing machine learning algorithm and attempt to integrate it with the prototype instrumentation system, and eventually with the UI adaptation system.

Due to the unique characteristics of the Bladed application, any kind of UI adaptation system would have to be tailor made. This is no different from almost any other application. The subtleties of user interaction vary so much between applications. I hope to be able to draw some conclusions from the experience of attempting to apply some state of the art technology to a real industrial application context.

# Proposed Solution Architecture

There are three parts to the architecture of an adaptive UI.

1. Gather user interaction data
2. Analyse the data
3. Modify the UI

Part 2 is probably the most domain-invariant part of the architecture. It has been the subject of research papers and has evolved slowly over the last decades. On the other hand, parts 1 and 3 are highly domain- and even application- specific.

Gather user interaction data

The Bladed application is being transitioned from an old UI technology to a new Microsoft .Net based architecture. Windows Presentation Foundation is the name of the framework used for generating the user interface. This framework is event-driven, and designed to respond to all kinds of user interaction event, so it will be straightforward to intercept and log all user interaction event. Once the target part of the UI has been identified for instrumentation, I will need to modify that part of the application to log interaction events to a service. I will then write a module that can be loaded into the application, and will provide the event-logging service to the UI module. The event-logging module will store data in a database of some kind. In the end product, this data store would reside locally on the PC of the user running the UI. However in the development stage the data may be transmitted back to a central location over the network so that I can analyse it.

Analyse the data

The machine learning algorithm will need to be executed at intervals, or in response to user interaction events. In this way, over time it will be able to build an evolving model of the user. Since my proposal is to at first use a generic algorithm, I will search for an appropriate existent library. In a preliminary search I have come across Infer.NET from Microsoft, WEKA from the University of Waikato and many others. I will need to be able to use whichever library I go with in the Microsoft .NET framework and the C# language. This algorithm will need to access the database used by the event-logging service. It will need to create and update a model of the user. The architecture of this model will depend on the types of UI adaptation identified as being most suitable for the investigation.

Modify the UI

Parts 1 and 2 of the architecture are layers that get added on top of the existing application structure. This part will require modification of the way the application works. As mentioned in the literature review, the UI can pull information from the model, or the model can push information to the UI. In the former case, the UI will need to be modified to know something about the model. In the latter case, the UI will need to offer an interface that the model will know about. Since my project is limited by time, I will be looking for the solution that requires the least modification to the existing application code. This would probably be the latter case – introduce an interface on the UI that the model can manipulate based on conclusions drawn about the user requirements.

# Proposed Research Method

The main research question I hope to answer is: Can the usability of the application be improved with UI adaptation. This could be seen taken as a qualitative research question. However I would like to make it partly quantitative by taking the meaning of ‘usability’ as the speed with which a user can perform a data-entry task. This allows me to quantify usability and compare before and after results. If I do observe an increase in the efficiency of data entry, I will need to assess whether it is down to the UI adaptation or other factors. I will need to somehow control for factors such as:

* increased user familiarity with the test UI in the ‘after’ case
* user motivation to apply themselves to the task properly

Within the scope of a UI window whose sole purpose is data entry, arguably the most important metrics of usability are resulting speed of data entry and quality of data entry. If the user thinks they have finished the task more quickly but have made more errors in the dataset, then this could be seen as a negative outcome.

After attempting to draw quantitative conclusions, I will attempt to assess the wider sense of the word ‘usability’ as it means the whole package, not just speed of data entry. I will conduct a questionnaire-type study about the user’s perceptions of the implemented system, what they do and do not like, what they perceive as potentially useful aspects that would warrant further investment.

In stead of using parts of the real Bladed UI for this quantitative study, I could use a mock UI which would allow me to control more of the variables. This is feasible given that way in which I will be implementing the systems will be easily extended to new UI screens.

# Ethical

As I will be collecting data about user behaviour I will need to bear in mind that the data could potentially reveal personal details about the user that should be private to the user. I will need to take due care that I do not distribute any data that can be personally identifiable. I will need to take care that personally identifiable data is secure against accidental divulgence. I will need to ensure that once these safeguards are in place, I ask permission of the user to gather data, ensure they are aware that data is being collected and ensure they are aware of the type of information that is being collected.

# Planning and Practical Considerations

I am a part-timer so optimistically project completion will be in Nov 2015, with the fallback date of April 2016. The following schedule is based on the optimistic hand in date.

The preliminary plan of action runs in 2 streams. Each stream can progress independently of the other. Stream 1 will consist of instrumentation of the application in order to gather data on user interactions. Stream 2 will consist of creation of an algorithm to process gathered data, and draw some conclusions about how the UI could be optimised to improve usability. At the end of these streams some modifications could be made to the application and an attempt made to quantify any improvement in usability.

|  |  |  |
| --- | --- | --- |
| Date | Stream 1 | Stream 2 |
| April – May | Instrumentation of application, generate mock data | Identify suitable UI adaptations, implement demonstrator app. |
| June – July | Gather real user interaction data | Get machine learning algorithm working with mock data |
| Aug - Sept | Integrate instrumentation, machine learning and UI adaptation into application |  |
| Oct - Nov | Gather real user interaction data on adapted UI | Write up |

This project has quite a few stages so I will need to make progress on the implementation of systems early. If gathering user interaction data proves problematic, I will consider creating mock data sets for testing the machine learning algorithm.

Bladed is a large application with many different data entry screens. In order to limit the scope I will choose a very modest and tightly targeted part of the application to study. I will consider creating a mock user interface if a suitable area cannot be found (although this would make it harder to gather “real” user interaction data).

Quantifying usability is going to be difficult, given the wide range of user interaction modes and the relatively small number of users

Intellectual Property will be a consideration. I will need to agree with my employer what data can be published. I will need to ensure that no intellectual property of the users is compromised in the data that is collected. For example if commonly used values are stored, they should be stored locally and not transmitted back to the development team.

# Bibliography

AKIKI, P.A., BANDARA, A.K. and YU, Y., 2014. Adaptive Model-Driven User Interface Development Systems. *ACM Comput.Surv.,* **47**(1), pp. 9:1-9:33.

BLUMENDORF, M., LEHMANN, G. and ALBAYRAK, S., 2010. Bridging Models and Systems at Runtime to Build Adaptive User Interfaces, *Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* 2010, ACM, pp. 9-18.

NAZEMI, K., STAB, C. and FELLNER, D.W., 2010. Interaction Analysis for Adaptive User Interfaces, *Proceedings of the 6th International Conference on Advanced Intelligent Computing Theories and Applications: Intelligent Computing* 2010, Springer-Verlag, pp. 362-371.

NORCIO, A.F. and STANLEY, J., 1989. Adaptive human-computer interfaces: a literature survey and perspective. *IEEE transactions on systems, man, and cybernetics,* **19**(2), pp. 399 <last\_page> 408.

PAYMANS, T.F., LINDENBERG, J. and NEERINCX, M., 2004. Usability Trade-offs for Adaptive User Interfaces: Ease of Use and Learnability, *Proceedings of the 9th International Conference on Intelligent User Interfaces* 2004, ACM, pp. 301-303.

# Appendix - Brainstorm of possibilities avenues of investigation

|  |  |
| --- | --- |
| **Measuring usability of the application** | Usability metrics:   * User perception of “ease of use” (qualitative) * Measure of time to input a given data set (quantitative) * Measure cognitive load (high load could lead to more errors) * Measure number of errors   Baseline:   * Could be to input a typical dataset using the existing UI * Could be to input a random dataset using a randomly-configured UI to control for pre-existing familiarity   Measuring usability:   * Could use the actual application to get real-world data. * To limit the scope and make the study more focussed, could create a test application with which the subjects would not be intimately familiar. |
| **Improving usability of the application** | Potential UI enhancements:   * Initial expand/collapse state * Re-ordering the fields for input * Automatically moving focus to the next field * Optimising fields for the user to type or paste in data * Optimising numerical fields to present user with most useful type of control:   + Text box   + Combo box   + Slider   + Check boxes   + Others…? * Present user with suggested values based on their historical interactions. * Arrangement of fields and labels on the screen   Alternative schemes / processes for UI enhancement:   * Application is deployed with instrumentation. Telemetry is sent back to the development team. They compile data over a period of time and then run through the algorithm. Conclusions drawn lead to UI enhancement for the next release of the application. * Application is deployed with instrumentation and algorithms. Telemetry is processed locally and UI enhancements made in-situ. |
| **Creating an AI algorithm** | Algorithms can be tested with mock data sets and conclusions drawn. Rough-and-ready data sets can be generated early in the project using a quickly-put-together application. I have contacts within the internal user community who may be willing to take part in experiments. Efficacy of algorithms may be difficult to judge until real and extensive data sets are gathered. Some evaluation of different algorithms may be possible simply using deduction or very fundamental synthetic data sets. |