

User Driven Modelling - Background Information

Explanation of the Problem to be Addressed

Research Aim

This research arises out of work to create systems to facilitate management of design and cost related knowledge within those organisations, with the aim of using this knowledge to reduce the costs of manufacturing products. This thesis identifies ways that problems arising from the model development process can be addressed by a new way of providing for the creation of software. With experience from projects, which have used a combination of proprietary software solutions and bespoke software, it is possible to identify the approach of User Driven Programming (UDP). This research unites approaches of Object Orientation, the Semantic Web, and Relational Databases and event driven programming. The approach encourages much greater user involvement in software development.

Software development is time consuming and error prone because of the need to learn computer languages. If people could instruct a computer without this requirement they could concentrate all their effort on the problem to be solved. This is termed User Driven Programming (UDP) within this thesis, and for the examples demonstrated the term User Driven modelling (UDM) is used to explain the application of user driven programming to model development. This research aims to create software that enables people to program using visual metaphors. Users enter information in a diagram, which for these examples is tree based. The program translates this human readable representation into computer languages.

Research Approach

This research demonstrates how a taxonomy can be used to automatically produce software. This technique is most suitable at present to modelling, visualisation, and searching for information. The thesis explains the technique of User Driven Model Development that could be part of a wider approach of User Driven Programming. This approach involves the creation of a visual environment for software development, where modelling programs can be created without the requirement of the model developer to learn programming languages. The theory behind this approach is explained and also the main practical work in creation of this system. The basis of this approach is modelling of the software to be produced in Ontology systems such as Jena and Protégé.

The research applies this User Driven technique to aerospace engineering but it should be applicable to any subject. The basis of the research is the need to provide better ways for people to specify what they require from computer software using techniques that they understand instead of needing to take the intermediate steps of either learning a computer language(s) or explaining their requirements to a software expert. These intermediate steps are expensive in terms of time, cost, and level of misunderstanding. If users can communicate intentions directly to the computer they can receive quick feedback and be able to adapt their techniques in a quick and agile way in response to this feedback.

An absolute requirement of this research is that no compromise is made in the openness of the source code of the application or of the information represented within the application. It is proposed that software and information represented by the software be separated but represented in the same open standard searchable way. This makes 'meta-programming' possible. Meta programming is writing of programs by other programs. The purpose of this is to provide a cascading series of layers that translate a relatively easy to use visual representation of a problem to be modelled into code that can be run by present day compilers and interpreters. This is to make it easier for computer literate non-programmers to specify instructions to a computer without learning and using computer languages. To achieve this any layer of software must be able to read the code or the information represented in any other. Code and information are only separated out as a matter of design choice to aid human comprehension, they should be represented by the same languages and in the same way. The methods used for this representation and translation will be explained in the rest of this document.

Why a different approach is needed

User involvement is important in the development of software but a domain expert does not necessarily possess expertise in software development, and a software developer cannot have expertise in every domain to which software might apply. So it is important to make it possible for software to be created using methods as close as possible to that which the domain expert normally uses. The proportion of domain experts in a particular domain (aerospace engineering) for example who can develop their own programs is fairly low, but the proportion that are computer literate in the every day use of computers is much higher. If this computer literacy is harnessed to allow the domain experts to develop and share models, the productivity for software development will be increased and the proportion of misunderstandings between domain experts and developers reduced. The domain experts can then explore a problem they are trying to solve and produce code to solve it. The role of the developer would then become more that of a mentor and enabler rather than someone who has to translate all the ideas of the expert into code themselves.

User Driven Model Development

The intention of the research into User Driven Modelling (UDM) and more widely User Driven Programming (UDP) is to enable non-programmers to create software from a user interface that allows them to model a particular problem or scenario. This involves a user entering information visually in the form of a tree diagram. The research involves developing ways of automatically translating this information into program code in a variety of computer languages. This is very important and useful for many employees that have insufficient time to learn programming languages. To achieve this visual editors are used to create and edit taxonomies to be translated into code. To make this possible it is also important to examine visualisation, and visualisation techniques to create a human computer interface that allows non experts to create software.

The research mainly concentrates on using the above technique for modelling, searching and sorting. The technique should be usable for other types of program development. Research relevant to User Driven Programming in general is covered as this could be applied to the problem in future.

This research unites approaches of object orientation, the semantic web, relational databases, and event driven programming. Tim Berners-Lee defined the semantic web as 'a web of data that can be processed directly or indirectly by machines' <http://www.w3.org/People/Berners-Lee/Weaving/Overview.html>. The research examines ways of structuring information, and enabling processing and searching of the information to provide a modelling capability.

UDM could also help increase user involvement in software, by providing templates to enable non-programmers to develop modelling software for the purposes that interest them. If more users of software are involved in creation of software and the source of the code is open this allows for the creation of development communities that can share ideas and code and learn from each other. These communities could include both software experts, and domain experts who are much more able to attain the expertise to develop their own models than they are using current software languages.

Criteria necessary for User Driven Model Development

This chapter explains the factors necessary to make the User Driven Model Development approach later outlined possible.

Firstly it is necessary to find a way for people with little programming expertise to use an alternative form of software creation that can later be translated into program code. The main approach taken was the use of visual metaphors to enable this creation process, although others may investigate a natural language approach. A translation method can then be provided that converts this representation into program code in a number of languages or into a Meta-language that can then be further translated. In order to achieve this it is necessary for the translator to understand and interpret equations that relate objects in the visual definition and obtain the results. In order for the user to understand the translation that has been performed it is then important to visualise the translated code and this must be accessible

to others who use the translated implementation. Web pages are a useful mechanism for this as they are widely accessible.