CASE STUDY: A SEMIOTIC APPROACH TO META-PROGRAMMING IN OPERATIONAL RESEARCH

Martin Wheatman and Kecheng Liu
Informatics Research Centre, University of Reading, Whiteknights, Reading, RG6 6WB, U.K.

Keywords: Semiotics, Meta-Programming, Operational Research, Discrete Simulation.

Abstract: This paper is a case study of how semiotics is applied to the meta-programming of application provision in a Operational Research setting. As a text representing instructions, software source code is a syntactic layer of an organisational sign system. In that meta-programming generates source code, it is sign system which acts at a pragmatic level generating other sign systems, and has its own norm-base. The case study illustrates how a meta-programming norm-base can be directed towards the provision of specific application components, but that direction itself can be programmable.

1 INTRODUCTION

Meta-programming is a sign-system for creating other sign-systems. This case study highlights meta-programming as a translational process – the automatic production of software from a set of intentional parameters. It shows how simple meta-programming techniques can be relatively straightforward, but their norm-base is highly static and directed towards the production of application specific components. These techniques have also been adapted in this case study to generate other intentionally dependent components: composing in an inductive manner, along with other ancillary functionality, a norm-base directed towards a specific outcome.

This paper starts with a description of the context of this case study: why meta-programming is used an Operational Research (OR) environment (Kelton and Law, 2000). It then contains a brief introduction to the Semiotics and reasoning of C. S. Peirce, and its application to IS. It continues with the rational for the use of semiotic approach in OR in general and in particular its associated IS; a description of the web based solution; and, finally the shortcomings of such a solution, again from a semiotic perspective.

2 META-PROGRAMMING IN OPERATIONAL RESEARCH

This is a case study of a program created to support the discrete simulation modelling of the nuclear decommissioning processes involved at Sellafield Sites Ltd., though it could be applied to modelling any manufacturing process, or indeed any discrete state simulation. As a large, multi-billion pound organisation with a strong ethical policy and a potentially large environmental impact, decisions have to be supported by evidence. While OR cannot predict the future, it can allow analysts to illustrate particular scenarios.

The Sellafield OR Group (SORG) supports the decision-making process by modelling operational activity, using discrete modelling tools such as (Witness) and (FlexSim). This paper concentrates on the use of the latter, which is based on two forms of datafile: a model file and one or more model data files, or scenarios.

A model is constructed in FlexSim by an analyst as a .fsm file which provides a three-dimensional view of a factory. The analyst can add processes, modelled by conveyor-belted machines, and items being processed, modelled as either as one of many predefined 3D items or customised items which can be specified by the analyst. To this basic model, other predefined or custom processes can be added, such as cranes, conveyors, forklifts or storage

Liu K. and Wheatman M.
CASE STUDY: A SEMIOTIC APPROACH TO META-PROGRAMMING IN OPERATIONAL RESEARCH.
DOI: 10.5220/0003254400470051
In *Proceedings of the Twelfth International Conference on Informatics and Semiotics in Organisations (ICISO 2010)*, page
ISBN: 978-989-8425-26-3

systems, to effectively represent that being modelled.

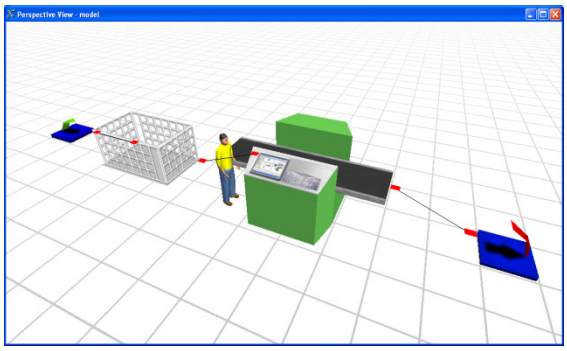

Figure 1: A FlexSim model showing (l to r) a source, a queue, an operator at a processor and a sink.

Models also have the ability to be programmed in a C++-like script language called FlexScript. This can, amongst other things, be used to load scenario data.

A scenario, along with its results once it has been run, is typically stored as a spreadsheet, which can be directly written by (or "exported to"), or read by (or "imported from"), FlexSim. Scenarios comprise all the parameters which are attributed to the items within the model, such as arrival times of item to be processed, time to process items, etc. Each scenario is run to ascertain the best model to be implemented, and hence affect decision making. The current use relationship between FlexSim .fsm files and .xls files is illustrated below.

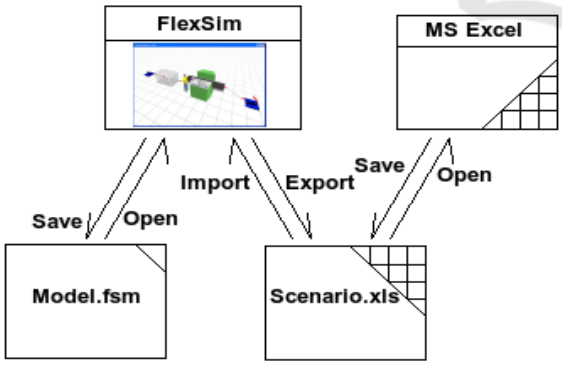

Figure 2: The existing relationship between FlexSim and Excel.

Practical problems have emerged with such use of spreadsheets: the file can easily be several hundred megabytes in size, which is neither easy to load nor easy to compare. Further, with the fact that analysts may have several models, even of each factory, and several hundred scenarios of each model; each of which might differ by only a few bytes. Moreover, with the need to archive data for several decades, it is not seen as being an efficient use of disk space. Plus there is the inability to efficiently compare results in separate files.

The solution has been to move to a database system which stores all scenarios as the differences from some baseline scenario. Working prototypes have being developed, using a WAMP Stack (Apache, CakePHP, MySQL). The solution is illustrated below: it is effectively the same information system but with a different implementation.

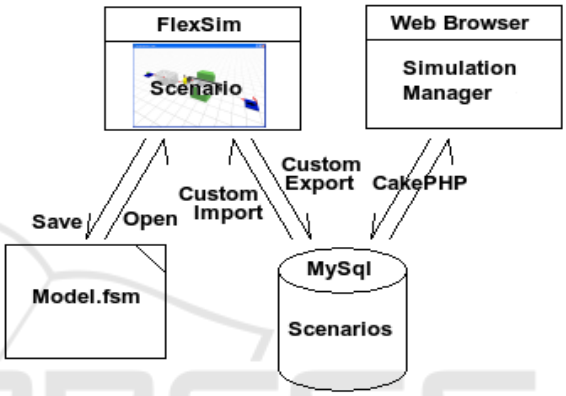

Figure 3: The proposed solution of using a database and web-browser architecture.

For this solution to be practical, though, existing scenario spreadsheets have to be converted into databases. This involves several processes which can be combined into one, largely dependent on the database schema and analyst intentions. To describe the semiotic perspective of this case study, some introduction to semiotics is needed.

3 SEMIOTICS

Charles Sanders Peirce developed the notion of a triadic sign model (Peirce, 1935-58), where a sign and the object to which it refers are connected only through an interpretant: a tangible reaction to the sign by an interpreter. It is therefore a model of subjective signs. He defined as "semiosis": the use and evolution of signs, how they lead to the creation of further signs, potentially endlessly. He also presented a triadic model of reasoning based on his sign model: hypothesis, guessing of the initial sign configuration given objects and interpretants or norms; deduction, determining the correct object

from signs; and, induction, the creation of norms from signs and objects.

Charles Morris organised the basic Peircean sign model into three layers of Syntax, Semantics and Pragmatics (Morris, 1938). His Pragmatic layer is concerned with the sign's life cycle: the creation, use, repeated use and destruction of signs, and their intentions. Stamper extended this model for IS, initially to a four layer model (Stamper, 1973) with and Empirical layer to cover the mechanisms involved in IS; then to a six layer model of signs (Stamper, 1991) with the addition of the Social and Physical layers.

Signs are the essential components of information systems (Stamper, 1985). Further, signs are modelled as being composite - macro-signs being composed of micro-signs (Gomez *et al*., 2002), and the resolution of their transformations into translations is the concept behind developing software (Wheatman, 2009). An overview of semiotics in information systems can be found in (Liu, 2000).

Semiotics, however, is not a method of producing software, but affords a framework for conceptualisation. If a formal description of functionality in software were required, techniques from UML toolbox could quite easily be employed to illustrate the design. This has an understanding of its own, through its own semiotic. So some rational for the use of the term "semiotic approach" in OR is required.

4 A SEMIOTIC APPROACH

The relationship between Semiotics and OR exists at several levels. It can be found in the comparison of modelling to semiotics in (Minsky, 1968): "To an observer B, an object A* is a model of an object A, to the extent that B can use A* to answer questions that interest him about A." Further, (Egesoy and Topaloğlu, 2009) notes that the terms object, model and observer are interchangeable with object, sign and interpretant, where interpretant/observer (what termed as a 'filter' (Minsky, 1968)). This approach is not pursued further in this paper.

Further, and from a wider perspective of an OR group within a larger department, the organisation of SORG falls into the organisational model, defined in (Stamper, 1991). This approach is also not pursued further in this paper.

The main application of semiotics is applied to the production of software source code. The concept of a model being composed of a model file and many scenarios composed of many tables is a Pattern which it is assumed can be applied to many OR applications. A Design Pattern is a representation of signs (Noble and Biddle, 2002), which implies that this should have its own life cycle – creation, (repeated) use, and destruction of (models and) scenarios.

However, the difficulty in producing the solution in this case study has been the lack of account taken of the pragmatics of meta-programming – that the meta-programming software chosen in this project is not flexible enough to support all required code generation. The contention here is that the configuration of a meta-programming solution amounts to an interpretant.

5 NORM-BASE INDUCTION

Prior to the work in this case study, creating scenario databases and all ancillary coding had been performed manually in a systems administrator role, as a transformation of the spreadsheet. Without deep analysis to the level of the computability of this process, it is possible to envisage automation of individual components. This automation is a semiotic transformation of a manual translation into an automated translation, parametrised by a few intentional details dependent on schema.

There is insufficient room in this paper to provide a complete breakdown of the design options open to scenario conversion. In practice, the working prototypes show that the chosen solution is valid – based on outcomes, rather than process. Expressing this as a formal sign system, in source code, provides an organisational norm-base which can be appended.

The fundamental process of transferring scenario data from spreadsheet to database, by reading named tables using an ODBC link and using the MySQL ODBC driver to write to database, is fairly straightforward and had already formed part of the manual process. There are many off-the-shelf components which will do such conversions; however, there is some extra information stored per row concerned with the versioning of the scenario. This is intentional, and details need to be supplied by the operator converting the scenario. To this norm-base, other processes can be added.

Further, the translation of a scenario also includes: creating the FlexScript to Import and Export to the database; driving CakePHP to generate the database application to support the model data in the database. Further processes may be included in

future. Scenario conversion means much more than data conversion, but its meaning in terms of action a singular meaning into which many ancillary processes may be subsumed.

The resultant tool, SM Import, sits behind a web front-end, and is illustrated below.

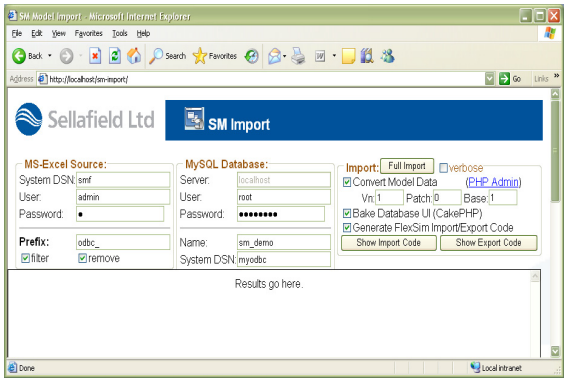

Figure 4: The SM-Import tool interface.

6 META-PROGRAMMING OF A DATABASE USER INTERFACE

CakePHP was selected early on for the generation of the database front-end, and has been used in creating all prototypes since, as a Model View Controller pattern. Its Bake process has been used manually to create interfaces for all tables in the database. This is a script based process which elicits intentions, e.g. database parameters, from the user from the command line. However, it generates code for an existing database – this is not necessarily the case for the SM-Import tool: the open source nature of CakePHP allowed it to be developed further in-house for the SM-Import process.

In creating the SM-Import tool, it was decided that the CakePHP code could be integrated into the SM-Import web application. Because the coding of the Bake class is relatively straightforward, this task mainly involved removing the prompting/script based methods from the class, and keeping the code generation methods. These methods can use the schema which is read during the conversion of the named tables to database data. The code generation methods mostly contain code which appends code to a string which is eventually written to a file, such as:

```
$add_file.="<div class=\"submit\">\n";
$add_file.="<?php echo \$html->".
    "submit( 'Add' );?>\n";
$add_file .= "</div>\n";
```

Therefore, they are directed towards generating code for specific tasks. This code is easy to modify, especially in that the resultant database does not need to record relationships between tables been generated.

6.1 Meta-programming of a FlexScript Database Interface

The other main function of SM-Import is the generation of the Custom Import and Export FlexScript code, which is similarly dependent on the schema; however, this has had to be written from scratch. This again was a relatively simple process of writing specific code portions for each table found in the spreadsheet, in statements such as:

```
fwrite($fh, "//clean up\n" );
fwrite($fh, "dbclose();\n" );
fwrite($fh, "pt(\"End Import\");
fwrite($fh, "pr();\n" );
```

From a syntactic point, the two code examples above are dissimilar, but their intention – the creation of source code – is the same technique. The reason for this syntactic dissimilarity is historic: the Import/Export code was written before it was decided to include the CakePHP code portions within SM Import.

Note that the code in the string constants, in both examples, needs special characters to be escaped by a '\' character. This is interference between the generating and generated syntaxes. Further, the escaping of the escape character can be alleviated by either the nesting of single and double quotes in languages such as PHP, or by the use of a meta-syntax, such as is used in HTML.

7 CONCLUSIONS

OR and Semiotics are both concerned with a defined representation of things and particularly with the action resulting from these representations. The repeated use of such concepts defines patterns. Meta-programming is a key technique to producing patterned solutions such as in this case study. The open availability of the CakePHP source code is vital to the implementation of an automated solution, as this allows its norm-base to be enhanced as necessary. The shortcoming is that CakePHP is geared towards producing web front ends: the norm-base is stored as hard-coded strings. Other code translational parts, while employing similar techniques, are less well supported.

This paper has outlined the author's work in designing and implementing an OR based web application, as a semiotic transformation of a manual sign translation process into an automated one. This transformation is constrained by the organisational norms determining SM-Import as a sign system in itself. One advantage of such automation is enabling the dissemination organisational norms, both throughout the organisation and to third party suppliers. Such descriptions necessitate at least the implicit use of the notions of semiosis: signs and their creation. Indeed, meta-programming is implicitly a sign translation process instigated at a pragmatic level.

A generic source code generator should therefore be useful in such applications. A Model Driven Architecture (MDA) approach could have been a starting point; however, since the prototypes already work this approach may require remodelling to fit it to an MDA framework. The solution as described has taken a lightweight approach as it requires a very straightforward code generation. A templated generator, such as the Java Emitting Template (Steinberg *et al*., 2008), might prove fruitful – as the template can be created from the prototype; however, this appears to output Java which outputs the template, rather than it outputting the raw text.

ACKNOWLEDGEMENTS

The authors would like to thank Sellafield Sites Ltd, and the staff of the Sellafield OR Group for the opportunities expressed in this paper. Particular thanks go to Simon Hughes and Daniel Braund, and Janet Davison for supplying the example model.

REFERENCES

Apache, see http://www.apache.org/

CakePHP, see http://cakephp.org/

Egesoy, A., Topaloğlu, Y. (2009) A Bottom-up Model of Computational Semiotics, *Information Systems in the Changing Era: Theory and Practice*, Proceedings of the 11th International Conference on Informatics and Semiotics in Organisations, 11-12 April, 2009, pp26-32.

Flexsim, see http://www.flexsim.com/

Gomez, A., Gudwin, R., Queiroz, J., Towards Meaning Processes in Computers from Peircean Semiotics, SEED Journal (Semiotics, Evolution, Energy and Development) 2002, pp69-79

Kelton, W. D., Law, A.M. (2000) *Simulation Modelling and Analysis 3rd Ed.*, McGraw-Hill

Liu, K., *Semiotics in Information Systems*, Cambridge University Press, 2000

Minsky, M.L., *Matter, Mind and Models: Semantic Information Processing*, ed. Marvin Minsky, MIT Press, 1968

Morris, C. W., *Foundations of the Theory of Signs*, Encyclopaedia of Unified Science, 1(**2**), University of Chicago, Chicago, 1938

MySQL, see http://www.mysql.com/

Noble, J., Biddle, R., Patterns as Signs, *ECOOP Proceedings*, 2002

Peirce, C.S., *Collected Papers of Charles Sanders Peirce*, Eds., Hartshorne, C. and Weiss, P., 1935-58

Stamper, R. K., *Information in Business and Administrative Systems*, BT Batsford, London, 1973

Stamper, R. K., Towards a Theory of Information: Information: Mystical Fluid or a Subject for Scientific Discourse?, *The Computer Journal,* British Computer Society, 28(**3**), pp195-199, 1985

Stamper, R.K., The Semiotic Framework for Information Systems Research, *Information Systems Research: Contemporary Approaches and Emergent Traditions*, Eds: Nissen, H.E., Klein, H.K., and Hirschheim, R., 1991, pp515-527

Steinberg, D., Budinsky, F., Paternostro, M., Merks, E. EMF: *Eclipse Modeling Framework - 2nd Ed.*, Addison-Wesley, 2008.

Wheatman, M.J., Semiotic Translation and Transformation in Source Code Development, *Information Systems in the Changing Era: Theory and Practice*, Proceedings of the 11th International Conference on Informatics and Semiotics in Organisations, 11-12 April, 2009, pp 56-62

Witness, see http://www.lanner.com/