ABSTRACT

Rule testing in transport scheduling is a complex and potentially costly business problem. This paper proposes an automated method for the rule-based testing of business rules using the eXtensible Markup Language for rule representation and transportation. A compiled approach to rule execution is also proposed for performance-critical scheduling systems.

Keywords: Rule-Testing, Scheduling, Temporal Business Rules, JIT Compilation, XML and .NET Framework.

1. INTRODUCTION

This paper reports the outcome of research undertaken to develop a system to test driver schedules against the driving rules in force in the UK and the EU (DoT 1998). The intention of the project was to reduce commercial scheduling costs by minimizing human intervention in the scheduling process.

One of the greatest challenges faced was interpreting the legal driving rules and translating them into the chosen programming language. This part of the project took considerable effort to complete the programming, testing and debugging processes.

A potential problem then arises if the Department of Transport or the European Union alter or change the driving rules. Considerable software development is likely to be required to support the new rule set.

1.1. Domain

Previous work within this area would seem to be spread across a number of sub-domains.

Transport Scheduling: (Weerdt 1999) reports a review of mathematical approaches to resolving the transport scheduling problem. Various algorithms and approaches are considered for moving from Point A to Point B in the most optimal way. This was useful to the research reported in this paper in the sense that it helped to understand the background of transport scheduling research but it doesn’t help to resolve the issues of rule testing as part of the problem.

(Wren, Fores et al. 2002) summarises the process of scheduling drivers to bus and train jobs by means of shift patterns. Whilst the rule testing element of scheduling is mentioned as important, there isn’t any discussion on how this is approached with the focus more on the scheduling aspect of the problem.

(Freling, Huisman et al. 2003) discuss the combined transport scheduling of drivers and vehicles as opposed to the independent scheduling of one followed by the other. Rule testing in scheduling, again, isn’t discussed.

In an earlier paper, (Freling, Huisman et al. 2000) consider the rule testing part of transport scheduling as an integral part of the process. It is suggested that it is heavily interconnected and that if the rules were to change in any substantial way, the process would no longer be valid and redevelopment would be necessary.

(Fischetti, Lodi et al. 2001) is an additional paper that investigates transport scheduling, however, as with most of the other papers in the area, it would benefit from a module that could be integrated into the system in order to carry out the rule testing process.

Rule Based Systems: The rule testing problem has been extensively researched in the past and the concept of the rule testing engine has moved from the research domain into the commercial domain and can be found integrated in a variety of applications.

This paper does not aim to develop a new rule testing engine as such, but to provide a means of working with rules of a temporal nature that could be used with scheduling systems. (Chen, Wetherall et al. 2005) looked at the same problem as this research but, in order to improve computation performance of the rule testing, the paper proposes the distribution of the rule testing process within a networked environment. This research uses an existing rule testing engine, (Dietrich 2003), to carry out the rule testing, however, this rule testing engine does not support the testing of temporal rules.

(Hayes-Roth 1985) provides a good introduction to rule based systems although it does tend to look at the If-Then approach to testing rules which differs from the types of rules outlined in (DoT 1998).

(Mak and Blanning 2003) provides a solid starting point for the domain expert to interpret the rules to be tested by a system into a simple set of business rules and would be a useful part of the bigger rule testing process.

The RuleML group has authored a variety of important papers, (Lee and Sohn 2003), (Wagner 2002) and (Boley, Tabet et al.
2001), which help to define an approach that can be used to represent rules in a way that can easily be transported across a network based upon the eXtensible Markup Language (XML) (Bray, Paoli et al. 2000).

Various papers in Genetic Algorithms have been investigated to see if there are ways, other that in rule-testing, to improve the computational performance of the scheduling system. The most recent genetic algorithm research tends to focus on algorithm distribution and parallelism, such as (Tongchim and Chongstitvatana 2002) and (Alba and Troya 2002). Further investigation into the current system, however, suggests that the majority of the computational performance is taken up in the rule-testing stages so further research into genetic algorithms was not necessary.

1.2. Contributions

The authors consider that this paper makes a contribution in three areas:
1) It proposes a scheme which can be used to improve the integration between rule-based testing and scheduling.
2) A rule definition language is proposed that is based on the concept of RuleML, however, it has been designed around the need for describing rules which relate to temporal problem domains such as scheduling.
3) Unlike other rule based systems, as the rule testing process is an integral part of the scheduling problem which can be very computationally expensive, this paper presents an approach to compiling the rules to optimise performance in continuous testing scenarios such as scheduling.

2. REQUIREMENTS

On the initiation of the research reported in this paper, the following requirements were identified:

- The solution must be able to represent rules in an abstract way, so that they can be transported between the rule provider and the rule consumer and so that they work amongst systems of different architectures. In the transport scheduling scenario, the rule provider may be an entity such as the UK Department of Transport, and the consumer is likely to be the application using the rules to test for infringements.
- The solution must be able to take the rules provided in the rule definition language and compile them into a form of executable code. This is in order that the rules will execute as fast as possible as per the requirements of a scheduling system.
- Additional supporting functionality or libraries should be able to provide host applications of the rules with the ability to integrate easily with the solution whilst relieving the need to implement functionality that would typically be common among all systems requiring the testing of temporally related rules.

3. RULE DEFINITION

Other Areas: Other areas of research which have also been investigated include more generic scheduling systems such as the work carried out by (Lu, Stankovic et al. 2001), (Beck and Fox 1998) and (Barták 2003).

There are a number of constraints placed on describing a rule which must be temporarily aware.

Take, for example, the rules provided by the UK Department for Transport (DoT 1998) of daily, weekly and fortnightly limits. Simplistically, the rules are that a driver can drive up to 9 hours a day, 56 hours a week and 90 hours in a fortnight.

Each of the three lengths of time mention above can be classed as a ‘Period’. Each Period has a set of rules that must not be broken, but some of the Periods are dependent on the information stored inside others in order to test the rules.

Figure 3-1

The Daily Work Period (DWP) is made up of a number of appointments or jobs with each job having a certain amount of driving time and a certain amount of rest time; the work length
of a DWP is the sum of the work length within each of its contained appointments.

The Weekly Work Period (WWP) is made up of a number of DWPs with its work length being the sum of the work length of the contained DWPs. Similarly, the Fortnightly Work Period (FWP) contains a number of WWP with its work length being the sum of the work lengths of the WRPs contained within.

It is clear to see that each Period needs a way of describing the other Periods that may be contained within it. It is also possible to realise that a Period needs to be able to describe which other Periods are able to come before or after it. For example, a DWP must follow or precede either a Daily Rest Period (DRP) or Weekly Rest Period (WRP). Similarly, a DRP must follow or precede a DWP.

This complex organisation of Periods is one of the main elements that separates the traditional rule-based system and specifically RuleML (Lee and Sohn 2003) from the research reported in this paper. Figure 3-1 illustrates an example of the DWP representation.

The DWP is a type of ‘Work Period’ and, as such, shares common attributes with other work periods. These are defined as part of the ‘Rule Support’ described in Section 5. As part of the description of the Period, the XML PrePosts element contains a list of the other Periods that may come before or after the described Period. The ‘Contains’ element describes the Periods that may be contained within the described Period.

Clearly, the ‘Description’ associated with the Period is just metadata for use at a later point to help understand the organisation of the Periods. A rule testing engine or rule compiler would be able to take the metadata and understand how the temporal rules work together.

The ‘Rules’ element contains a description of the rules in human-readable form that are associated with the Period. The rules are designed to be used with scheduling systems to determine if the rule is broken or not and, as such, only returns a Boolean value. In this example, there is only one rule which is a comparison to determine whether the Work Period described has a WorkLength <= 8 hours. As this Period is a type of Work Period, the WorkLength attribute relates to the sum of the WorkLengths of the contained elements, the ‘Appointments’.

4. RULE COMPILATION

It is widely accepted that the computational performance of software which has been compiled is greater than an interpreted equivalent. One of the important requirements of the research reported in this paper is to look at approaches that will help to improve computational performance as the rule testing process has to work as part of an already computation-intensive activity, the scheduling process.

Various approaches exist to provide the means to turn human-readable text into machine code. Traditionally, for example, the C language would use a compiler and linker to carry out this process. Modern programming languages and infrastructures exist which provide additional levels of abstraction from the traditional low-level compiler and linker approach. The .NET Framework provides a solid foundation of base classes, a powerful runtime and a Just-In-Time (JIT) approach to code compilation from an Intermediate Language (IL) to machine executable code. This JIT approach provides the facility to produce machine code that takes advantage of advanced processor specific instructions that more general executable programs don’t often cater for.

Part of the .NET Framework foundation classes is the ‘Reflection’ namespace which provides the ability to construct elements of executable code by emitting specific IL instructions. The executable can then be invoked as part of the running process causing the JIT compiler to compile the IL into machine code and run at the fastest possible speed on the given architecture.

A rule compiler has been developed which takes the XML described rules and produces executable code. The process of compiling the rules requires very little execution time and only needs to be carried out once per rule description. Once compiled, the resulting dynamic link library (DLL) is cached and the rules are only recompiled if they change. This can also be the case for the JIT compilation process of the IL which can store the machine code version of the DLL in the Global Assembly Cache (GAC) maintained by the .NET Framework.

The process of compilation is recursive in the sense that the top level rule is told to compile, emitting its IL into the DLL, this in turn calls any rule elements stored within it to emit their IL into the DLL and so on. This continues until all elements have emitted their IL and the DLL is produced.

5. RULE SUPPORT

In order to effectively compile and execute rules, especially those of a temporal nature, a certain amount of additional support must be present. This rule support must provide the ability for system integration of the solution into existing host applications, scheduling or otherwise, that wish to test rules of a temporal nature.

An appointment in a traditional worker system, such as a driver scheduling system, has a set of common attributes that would be consistent with other systems such as a school timetabling system. To provide a tighter means of integration an ‘Interface’ has been designed to allow a common means of describing the raw data that a system will provide to the compiled rules for testing.

Figure 5-1 shows the defined interface of an IPeriodElement, an element or period of time represented by a start and finish, also taking into account the need to describe a WorkLength and RestLength related to that period of time. The Appointment class which is shown to implement the interface describes a period of time that a driver may be carrying out an appointment. In this example, the WorkLength may be the length of time between the Start and Finish, or, in more advanced systems, the WorkLength may be calculated based upon the length of time a driver is driving between locations, providing the flexibility for an Appointment to also describe a RestLength. This is based upon the consuming application’s implementation of the IPeriodElement interface.
In addition to the integration functionality built into the support elements of the solution, there is also the need to provide a method of organising the data provided by the consuming system, by means of IPeriodElement implemented objects, into a work plan. This needs to be based on the description of the rules provided within the rule definition language.

If, for example, a rule states that a driver can drive for 7 hours but then must have a 12 hour rest before being able to drive again, the rule testing process would need to be able to organise the timed elements in such a way that they can be tested against that rule.

For the benefit of testing the solution, a simple approach was developed to build up a work plan based upon the metadata provided in the rule definition and the data itself, provided in the form of IPeriodElement objects.

Figure 5-2 is a flowchart illustration of the process used to build up a work plan. The figure is a simplified version of the complex process and, as such, omits some detail.

6. VALIDATION AND VERIFICATION

The value of the research reported here can be measured in two ways, firstly, in a traditional research method of computational performance comparisons of the previous solutions against the proposed solution or, in some ways more important to this research, using Measurable Organisational Value (MOV).

As highlighted in section 1.2, the authors consider that there are three elements of traditional research based contributions made by the work reported here. The first involved looking at approaches to rule testing that can support a wide range of scheduling problems. This has been achieved in a number of ways, most noticeably in the ability to integrate the prototype rule testing system into a variety of scheduling applications. As part of this research the rule testing engine was integration with a transport scheduling system and school timetabling system.

The second contribution was in the specification of a rule definition language, similar to that of RuleML, but that has the additional ability to describe temporal rules. This has been achieved with an example of the working solution outlined in Figure 3-1.

Finally, the third contribution is considered to be in the approach of compiling the rules into executable code in order to speed up the computational performance of the testing process, tailored toward scheduling algorithms which are themselves often computationally expensive. This has been achieved and is outlined in Section 3.

In terms of the MOV for the reported solution, the cost of employing a member of staff to undertake the transport scheduling process within a commercial organization is high. The staff member must have a strong understanding of the legal driving rules as well as a strong understanding of scheduling. The solution reported in this paper transfers the burden of scheduling from a staff centered process to a staff supporting process with input required only when an optimal solution cannot be obtained automatically. The reduction in computation cost gained from the compiled approach to the processing of the rules speeds up the scheduling process, providing more time for the domain expert to tweak the automated schedule, as necessary.

7. CONCLUSION
The authors consider that the findings of this research can help to improve productivity across the transport scheduling sector, where scheduling is a key activity.

Branching further out into other scheduling application areas, the rule testing solution proposed can help to undertake a wide range of other scheduling processes more efficiently.

As laws and regulations change, the working practices of companies have to be able to adapt with these changes. Having the ability to update rules, compile them, then carry on running with a new business process, automatically, with little or no user involvement, improves the productivity in these areas and helps organisations get the most from their IT systems.

8. FURTHER RESEARCH

There are a number of areas where the authors have identified that further improvements might be made. The most prominent to the given solution is in the organisation of the data provided by the host application into an effective work plan. Currently the technique used is overly simplistic, helping to prove the concepts of the other areas but undoubtedly having a negative impact on the full potential of the solution.

In addition, for the system to become easier to use for the rule generator, whether this is the organisation using the system or the organisation defining the rules, a Graphical User Interface (GUI) for rule generation would be necessary otherwise the user would need to understand the XML syntax which maybe inappropriate in many situations.

9. REFERENCES