This is an *author-produced* version of an article appearing in the
IEE Colloquium on Real-Time Systems (Digest No. 1998/306),
York, UK, 21 April 1998, pp. 6/1-6/5.
The final publication is available via https://doi.org/10.1049/ic:19980527

'Lean' Formal Methods in the Development of Provably Correct Real-Time Systems*

Antonio Cau, Hussein Zedan and Ben Moszkowski
Software Technology Research Laboratory
De Montfort University, Leicester, UK.
E-mail: {cau, zedan, benm}@dmu.ac.uk

Alastair Ruddle
The Motor Industry Research Association (MIRA)
Nuneaton, UK.
E-mail: alastair.ruddle@mira.co.uk

1 Introduction

Designing software to control *real-time, reactive embedded* applications is non-trivial. And as the complexity of such systems increases, the present industrial practice for their development gives cause for concern, especially, if they are to be used in safety-critical applications. In order for the design of these systems to be optimised, it is necessary to take into account the interdependence of the hardware and software. Thus, the system needs to be assessed at all stages of the development life-cycle in order to minimise the potential for errors. This has resulted in the development of a wide range of techniques which aim to support the analysis and design of both systems and their associated software. These vary from those with sound mathematical basis (*formal methods*) to *structured methodologies*. The latter, while useful, do not provide a satisfactory and comprehensive solution. The former, on the other hand, are recognised as the most likely solution to the problem, but insufficient expertise and a lack of tool support have limited their practical deployment to highly specialised applications [2, 3]. In the automotive industry, however, the **MISRA** *"Development Guidelines for Vehicle Based Software"* recommend the use of formal methods in the specification and analysis of systems at safety integrity levels 3 (difficult to control) and level 4 (uncontrollable). At present this represents an extremely difficult and costly step which has not yet been widely investigated by system developers. These weaknesses in current analysis techniques therefore represent a significant obstacle for the deployment of advanced automotive electronics in the future.

The general objective should be therefore to bring real-time systems and software engineering activities to a similar level of maturity to that of traditional engineering disciplines. The key to this is predictability through the developoment of, what might be called, **'lean'** formal methods which will be strongly supported by a suite of powerful and tightly integrated, practicable and affordable software tools. By a **lean** formal approach, we take the view that such a method must be supported by automated tools that make the method more widely accessible to users. Such tools should provide compositional rapid prototyping, testing and verifiction. **Compositionality** is the mathematical translation of the well known

*Funded by EPSRC Research Grant GR/K25922: A Compositional Approach to the Specification of Systems using ITL and Tempura.

concept of *modularity* which has been advocated as the ultimate solution for the design of large scale systems.

2 Bomb Disposal Robot

The tele-operated robot [1] is a tracked device which was originally developed for military use. The carriage can easily traverse over rough terrain. The vehicle schematic is shown in Figure 1. The vehicle has on-board a manipulator arm that has three degrees of freedom controlled by hydraulic actuators. The electric drive motors, manipulator actuators and on-board lamps may be controlled manually by the operator via a control box that is linked to the vehicle by a RS422 link. The vehicle is controlled by an on-board distributed micro-controller system based on NEC 78330 devices, which have a 16-bit internal architecture and 8-bit external data bus. The controllers have no on-chip ROM but have 1K byte of RAM. A total of 9 8-bit I/O ports including digital, analogue, counter/timer and serial communication channels adequately meet the I/O requirements of the system. Currently one controller is dedicated to vehicle control, one to the infrared sensor interfacing and processing, and a third to the on-board camera.

The vehicle is driven by two motors, left and right, both of which can move forwards and in reverse. The vehicle is steered by moving one motor faster than the other. It is also possible to drive only one motor at a time, so that the robot will turn. The speed of the motors is directly proportional to the value written to them.

From a control point of view, commands are issued to the motors via a operator joystick which issues integers values in the range $0 \ldots 127$ for forward motion (127 max. speed) and $0 \ldots -128$ for reverse motion.

The robot is equiped with 8 infra red sensors. These return an integer value in the range $0 \ldots 255$ depending on whether an obstacle is present or not: 0 indicates no obstacle, 255 indicates obstacle very close by. We normally operate with a threshold of around 100, above which we take notice of the sensor readings, i.e., an obstacle is of interest. At this point reactive control takes over from the manual control by moving the vehicle away from the obstacle until the 100 threshold is not set. The sensor positions are as follows: N, NE, E, SE, S, SW, W and NW, covering the body of the robot and shown in Figure 1.

The mounted manipulator arm on the robot may only be activated once the mobile base has stopped moving. When the base is moving the manipulator is in its down position. There are three hydraulic drives, using on/off control valves. That is, if they are switched on then they drive that axis forward or backward until the end limits are reached. There are no sensor readings for the manipulator.

3 Integrated Lean Approach

The approach which has been developed provides a reliable link between a logic-based formalism, namely, Interval Temporal Logic (**ITL**), Tempura (an executable subset of ITL) and a real-time refinement calculus, known as **TAM**. The resulting ITL-workbench comprises various tools which support the process depicted in Figure 2. This process may be summarised as follows:

1. The system specification is expressed as an ITL formula.

2. The ITL formula is then refined into TAM code if possible, otherwise the specification is reworked until this can take place.

[1] The UK Defense Research Agency kindly provided the bomb disposal robot.

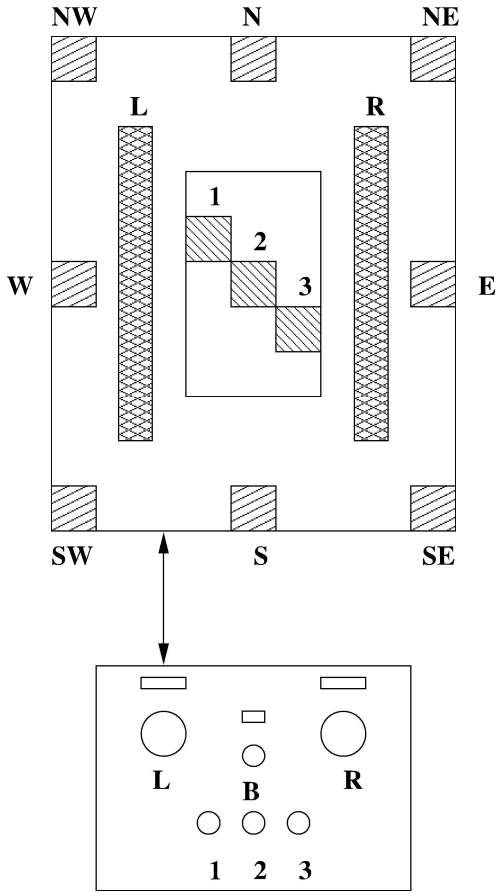

Figure 1: The robot control system

3. The TAM code is simulated using Tempura. If this is successful, the TAM code may be further refined otherwise the specification is reworked to provide a more accurate representation of the original requirements.

4. The TAM code is translated into a real programming language such as C or ADA.

We should note here that the TAM code represents the last stage in our formal development method. The TAM code could either be executed directly, or be translated into an industrially accepted target programming language such as C or Ada. Care must be taken, however, to ensure that the target language and TAM are semantically equivalent.

We make the following observations about the proposed process.

1. Once the formal specification phase has been completed, various properties may be proved about the specification itself. This can provide assurance that the final specification meets the informal requirements. This is acheived within our ITL-workbench using PVS [1, 4]

2. The ITL specification can be directly executed using Tempura. This gives further confidence in the validity of the specification at an early stage of the development. In addition, various timing analyses may be performed.

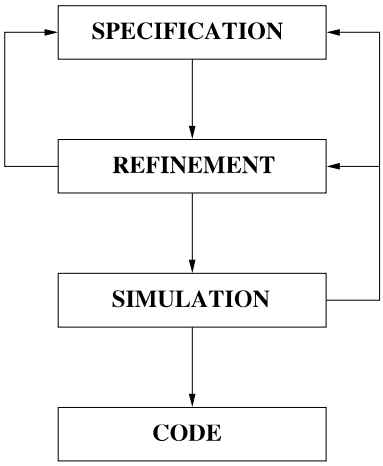

Figure 2: The design strategy

3. At each refinement step, the resulting (sub)system may be simulated. This gives some useful guidance in the selection of the subsequent refinement rules.

4. The refinement calculus used in this technique (TAM) is based on a **scheduling-oriented** model which was introduced in [5]. In this model, scheduling issues, such as, scheduling analysis and the construction and correctness of schedulars, are handled. It also uses the **delayed specification** technique, which we believe makes formalizing many types of temporal requirements much easier and less error prone.

5. The output from Tempura provides a useful source of design documentation.

Figure 3 below shows the Tempura screen output of a partial simulation run. This figure shows that in state 3, for example, both the infra-red sensor (*ia*) and the operator (*op-active*) are enabled, whilst the operator commands (*l-op-cmd* and *r-op-cmd*) are taken as the operator overrides the commands of the infra-red control.

In the interest of brevity, we have not included either the ITL specification of the case study or the derivation of the C code using our refinement calculus. The full paper could be obtained from the authors or may be viewed on our web page: **http://www.cms.dmu.ac.uk/STRL/**. From our web page you may also obtain a full description, together with the asscociated published papers, of ITL and the TAM calculus.

4 Conclusion

The practical deployment of formal methods in real industrial problems will not be possible without the support of a tightly integrated suite of powerful software tools which are accessible to engineers. This work has demonstrated the successful integration of several tools supporting the formal analysis of real-time systems. These include PVS for theorem proving, Tempura for executing the ITL specification, and TAM for the scheduling analysis. Nonetheless, further work will be required in order to attain our goal of a truly 'lean' formal method. This includes the development of graphical model building tools to insulate the user from the underlying ITL representation, and refinement tools to support the automatic generation of code in the selected target language.

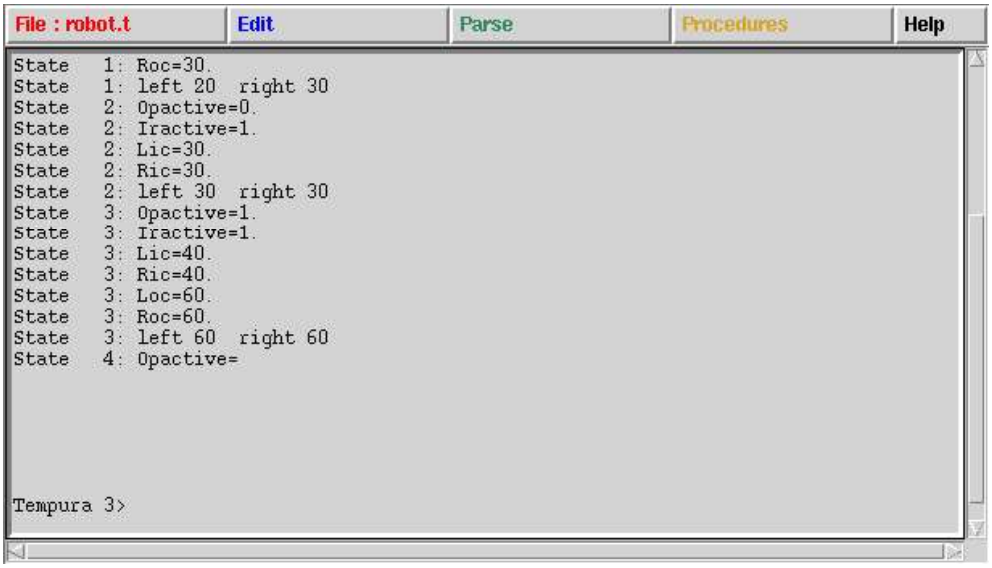

Figure 3: Screen dump of the simulator

References

[1] A. Cau and B. Moszkowski. Using PVS for Interval Temporal Logic Proofs. Part 1 The syntactic and semantic encoding. Technical report, SERCentre Technical Monograph 14, De Montfort University, 1996.

[2] MoD. *The development of safety-critical software for airborn systems*, Interim Defence Standard 00-3/1 (1987).

[3] MoD. *The procurement of safety-critical software in defence equipment*, Part 1, *Requirements*, Part 2, *Guidance*, Interim Defence Standard 00-55, Issue 1 (1991).

[4] A. Cau and H. Zedan. Refining Interval Temporal Logic Specifications. In M. Bertran and T. Rus (eds.), *Transformation-Based Reactive Systems Development*, *LNCS* 1231, 79–94. AMAST, Springer-Verlag, 1997.

[5] G. Lowe and H. Zedan. *Refinement of complex systems: a case study*. *The Computer Journal*, 38, No. 10, 1995.