Formal Virtual Modelling and Data Verification for Supervision Systems

Conference Paper · June 2015

DOI: 10.1007/978-3-319-19249-9_41

1 author:

Thierry Lecomte
ClearSy System Engineering
54 PUBLICATIONS **418** CITATIONS

SEE PROFILE

Formal Virtual Modelling and Data Verification for Supervision Systems

Thierry Lecomte[1]

[1] ClearSy, 320 avenue Archimède,
13857 Aix en Provence, France
thierry.lecomte@clearsy.com

Abstract. This paper reports on the use of formal techniques to ensure as far as possible a safe decommissioning of a plant several decades after it was designed and built. Combination of supervised learning, formal modelling, model animation and model checking enabled the recovery of an almost lost specification and the design of a virtual supervision system that could be checked against recorded plant data.

Keywords: B Method, safety critical system, Event-B, model animation, model checking

1 Introduction

Industry plants that were built in the 60s or 70s (nuclear plants for example) usually have their design documentation far from current standards, some could be handwritten or referring to punch-card-based programs. In any case, initial designers cannot be asked as they are now retired or dead. As such, any modification to existing installation (decommissioning) is a real engineering challenge and requires cautious investigation and experiments before the deployment on the real plant, especially when it fulfills a safety critical mission.
Recovering the original specification of the plant, as a virtual model that could be checked against recorded data during the lifetime of the plant, would constitute a good starting point prior to any functional modification of such a plant.
Formal methods [1] enable to obtain precise specification document, as all ambiguities are removed due to the semantics of the formal language used for the modelling [4]. Such a model is then likely to be animated [2] (i.e. virtually executed) through a number of scenarios elaborated from real life. The construction of scenarios is tough work as the recorded data are interleaved and not tagged: some pattern recognition is required to sort out the data.
The objectives are to recover the system specification, to design a supervision system that can bootstrap the existing one and to ensure forthcoming functional evolution.

2 The Supervision System

The supervision system (Fig 1) described in this article does not control directly the plant. Instead it is in charge of surveying actions issued by other systems, displaying information (messages) when an action has been completed successfully or not. Its mission is to assist human operators to make sure that the different actions carried out complete successfully or to provide a negative feedback. For example, when a pump is switched on or when a valve is open, it may result in the increase of a level in a tank, measured by a sensor after some time. When the threshold level is reached, the system may display a successful message. In case the level is not reached after a certain period, an error message may be displayed. This survey is performed through a number of programs, each program being in charge of one particular sequence of actions.

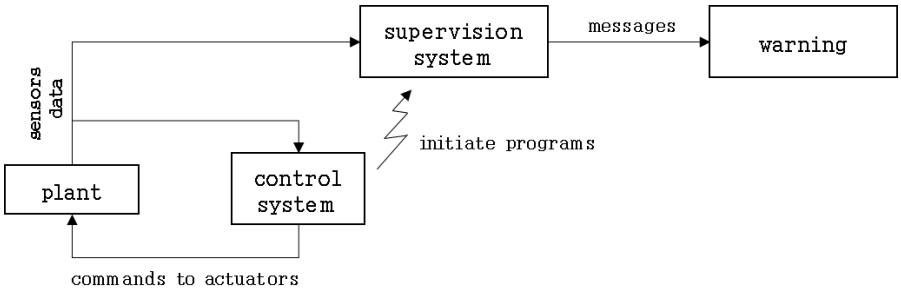

Fig. 1. Architecture of the industrial plant: the supervision system provides feedback for the human operators.

These programs (Fig 2) are directly triggered by control & command systems that require their execution to be followed-up. Basically these programs set up predicate listeners and watchdogs, start other programs and then go to sleep. Their execution is resumed when, following the update of a variable, a predicate state changes to TRUE, or when a timer elapses. Execution, sleep and resume actions are managed through an execution queue: the program on top of the queue is the only program executed, the others are waiting to be executed or are sleeping. A program entering a sleep period is moved at the end of the queue. A program resuming its execution is put on top of the execution queue. Executed program is also able to generate messages (on various displays, on printers, etc.).

In practice, this supervision system is made up of several execution queues: one with few programs able to be executed and hence able to take into account events (inputs) more quickly, others with potentially more programs in the queue (then with slower response time). There are some constraints like the impossibility for a program to be executed at the same time on different execution queues. Such system has to deal with a lot of asynchronous events ranging from milliseconds to half-an-hour periods, with an avalanche effect leading to more than 10,000 events to process, when the plant meets a functional transition that affects most sensors.

3 Formal Techniques in Action

Several formal techniques are used in combination in order to address all engineering problems that has appeared during the ongoing project, namely: recovering the specification of the supervision system and checking it against real data to ensure a high level of confidence.

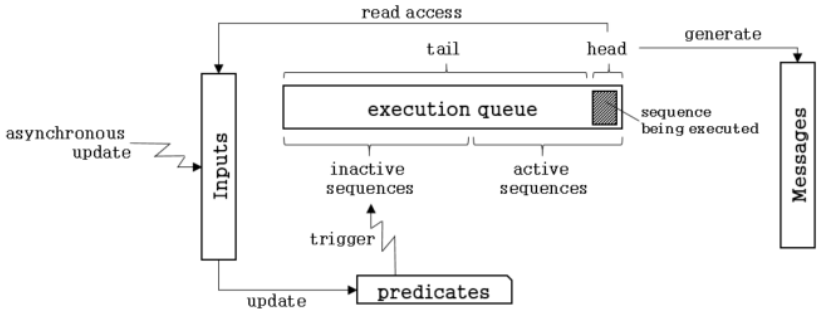

Fig. 2. Architecture of the interpreter in charge of executing sequences of instructions (i.e. programs). The real interpreter is made of several execution queues with different priorities. A sequence can only be executed in a single execution queue.

3.1 Recovering the System Level Specification

The specification was entirely modelled with Event-B and animated with the ProB model-checker. The plan was first to mimic the behavior briefly described in various technical documents available: events were added first, then some scenarios were designed by-hand and replayed. Preconditions for events were completed to ensure that only regular scenarios to occur. Invariants (other than typing) were then added to add some guarantees to the model. Finally the model was proved against these invariant properties. This modelling was iterated several times, by including larger parts of the specification. If the core specification was quite easy to complete, the introduction of exploitation modes was trickier; for example the fact that the supervision system cannot be shut down at any time. So the upload of new programs has to be done on-the-fly and some new properties emerged (never upload a new version of a program that is being executed) and the model had to be slightly refactored to integrate these new features.

3.2 Checking against Decades of Recorded Data

The formal model obtained previously has been checked against our understanding of the natural language specification documents. It constitutes a good starting point but we had no idea how close our model was from the real system. The documents may not correspond to the system being executed - remember that documents are hand-written and some of these might have not been updated. Hopefully decades of recorded data are available even if they are stored on very old medium. Recorded data include timed inputs (sensors), commands (actuators) and messages identifiers emitted by the supervision system. The objective of this phase was to extract real scenarios in order to check if they can be replayed with our formal model of the supervision system. The major issue was to identify programs being executed, as a huge number of programs (up to 10,000) can be executed "simultaneously" in a short period of time. Allocation of data to scenarios was completed with simulated-annealing techniques: several weighting methods were used to distinguish interleaved data and 100% was managed for the latest recorded years. Unfortunately we did not managed to apply this technique to the full set of recorded data as it appeared that programs were modified several times. As only the latest version of these is currently available, the simulated-annealing process cannot be applied if the patterns to search for are not completely known.

4 Experience gained and Conclusion

We have been able to reverse-engineer the specification of the supervision system with a help of formal modelling and animation in Event-B, by using Atelier B and ProB tools, which enabled us to make a number of "wise guesses" and to slightly improve the correctness of the model. This specification has then been checked with success against some scenarios extracted from real recorded data. Indeed the behavior exhibited by our virtual animation seems to comply with the existing supervision plant. The next step would be to connect the software model to a virtual model of the

plant, and to obtain a co-simulation that would allow to check timing behavior especially when a large number of events has to be taken into account. Forthcoming investigations will be linked to the H2020 Into-CPS project. Another issue is the coupling of the interpreter with its virtual model to check at runtime discrepancies between the real plant and its model. Formal data validation [3][5] will also be used for checking parameters and programs correctness.

This way of reverse-engineering old systems seems promising, as formal techniques and tools are now mature enough to be applied to real industry-strength systems.

References

1. Abrial, J.R.: The B-Book: Assigning Programs to Meanings (2005), Cambridge University Press
2. Hallerstede, S., Leuschel, M., Plagge, D.: Validation of Formal Models by Refinement Animation, Science of Computer Programming, 03/2013
3. Lecomte, T., Burdy, L., Leuschel, M.: Formally Checking Large Data Sets in the Railways, Proceedings of DS-Event-B 2012: Workshop on the experience of and advances in developing dependable systems in Event-B, in conjunction with ICFEM 2012 – Kyoto, Japan, November 13, 2012
4. Lecomte, T.: Safe and Reliable Metro Platform Screen Doors Control/Command Systems, 15th International Symposium on Formal Methods, Turku, Finland, May 26-30, 2008
5. Leuschel, M., Falampin, J., Fritz, F., Plagge, D.: Automated Property Verification for Large Scale B Models with ProB, Formal Aspects of Computing, 11/2011 (683-709)