Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 116 (2005) 99–111
www.elsevier.com/locate/entcs

Validation Coverage for a Component-based SDL model of a Railway Signaling System [1]

M. Banci[2], M. Becucci[2], A. Fantechi[2], E. Spinicci[2]

Dipartimento di Sistemi e Informatica
Università degli Studi di Firenze
Firenze, Italy

Abstract

In this paper we present an application of formal verification techniques to a component-based SDL model of a railway signalling system lent by General Electric Transportation Systems. A MSC-driven validation technique has been applied to verify the multiple-configuration features of the system. This work addresses the problem of validating a *component-oriented* designed SDL model, with a partial reuse of previously verified MSC scenarios if a new component is introducing or modified: some possible solutions based on the coverage metrics and information provided by the adopted tools are discussed.

Keywords: Component-based SDL model, railway signaling, validation, MSC scenario

1 Introduction

The availability of commercial tools for the support of formal specification and verification has boosted in the recent years the use of formal methods as a mean for preventing the introduction of software faults in safety critical systems, and in particular the use of those specific formalisms that the tools support. Two major examples in this direction are SDL [1], a standard developed within the telecommunication industry, and the Statecharts

[1] This work has been partially supported by the Italian Ministry of University and Research within the COFIN 2001 project "Quack: a platform for the quality of new generation integrated embedded systems"
[2] Email:[banci, becucci, fantechi, spinicci]@dsi.unifi.it

doi:10.1016/j.entcs.2004.02.083

[3] supported by the iLogix Statemate tool. We refer to [7] and [6] for two notable examples of application of such formalisms to railway signalling systems, that is the application field of interest of our study. Within a collaboration with General Electric Transportation Systems (GETS), we have studied first the introduction of SDL technology in the GETS development cycle for a multiple-configuration railway signalling system [4], assessing a suitable validation technique [5]; the next step has been the investigation in the direction of a component-based design of the previous SDL specification. Actually, a SDL model which should preserve multiple-configuration features needs to be described by a varying number of general components connected under a given topology, that is, by defining *configurable component types*: the number of the needed instances for each component type in a given configuration can be set by means of suitable configuration parameters. In this kind of model, inserting a new component or modifying an existing one may imply not only to change the interfaces between the existing components and the new one, but also to modify the topology of the model. This involves to execute again the verification activity on all the defined scenarios. In a *component-oriented* SDL model, basic component specifications must be developed and verified; then, one or more static replicas of each component, accordingly to the given configuration, are to be interconnected. A final verification of the obtained model will then be performed; Message Sequence Charts (MSC) [2] can be used to define simulation and test scenarios.
In this paper, we discuss the application of MSC driven validation to verify a simple configuration from the adopted case study, by pointing out the reuse of the test scenarios when a new component is introduced. The paper is structured as follows: in section 2 a description of the case study is presented, together with an overview of the component-based SDL model. Section 3 and its subsections introduce the adopted verification methodology, detailing the coverage measurement to evaluate the performed validation activity and finally the obtained results.

2 The case study: SCA system

The SCA system (Sistema Conta Assi, axle counter system), developed by General Electric Transportation Systems, G.E.T.S., is a device which counts the number of wheelsets belonging to trains which are travelling in a given railway section. This operation is made using suitable sensors, which are placed aside the track. The consecutive calculations of the number of wheelsets entering and leaving the railway section allows to establish if the section itself is free (if the number of leaving wheelsets is equal to the number of entering

wheelsets) or is still occupied by a train. This information is used to enable or to forbid the next train to enter the section, by means of a suitable (external) semaphore signalling. The SCA system is composed by several control and acquisition units, named UCA (Unità Conta Assi, axle counter unit), placed along the tracks, and by detection places located aside the track, named PRA (Punti di Rilevazione Assi, axle detection places). The UCA-PRA connection is shown in Figure 1.

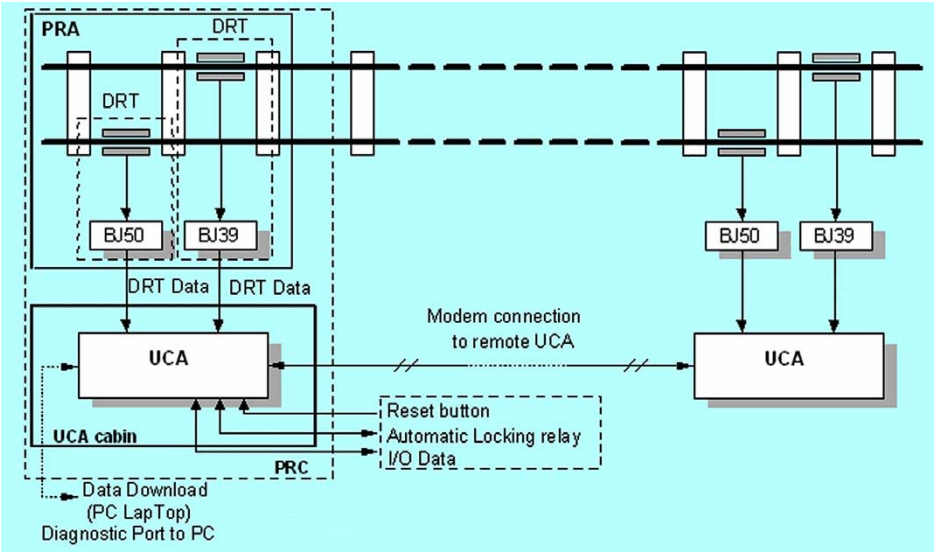

Fig. 1. SCA system in single-section configuration, highlighting UCA-PRA interconnection.

A PRA is formed by two magnetic sensors named DRTs (Detettore Ruota Treno, train wheel detector), located each at one of the rails, for redundancy. The two magnetic detectors must give the same signal consecutively. If only one of them sends the signal of a passing wheel, the UCA goes in a "fail-safe" state, in which a suitable automatic locking relay, connected to the external signalling system, is enabled. A typical effect is to set all signals to red. UCA units are located at the beginning of a section; they receive and process the signals modulated by the DRTs, in order to determine the number of axles running on the track; these data are sent to remote UCAs via modem. Using remote data, the UCA can determine whether the section is still occupied or not, and in the first case it stops a following train by enabling the automatic locking relay. Since a track can be generally run in two directions, UCAs are able to check both the trains entering the railway section and the leaving ones. The connection between remote UCAs, depicted in Figure 1, shows the basic configuration of the system: the single-section SCA can check only one

railway section. It is formed by two UCAs, each of them placed at one end of the section; they can receive signals coming from up to 3 PRAs. Moreover, the SCA system can present several configurations depending on the number of composing UCAs and on a set of parameters that are located in an external NOVRAM and that determine the functionalities of the given configuration. We refer to [4], [5] for a detailed description of the other configurations and their functionalities.

In this paper, we have investigated a particular case of the single-section configuration, in which the primary track includes a point from which a secondary track departs. In this kind of SCA system, two PRAs are linked to the entry UCA, in such a way as to make a so called *flank-protection* (see Figure 2). Due to the presence of an additional PRA, the communication between PRA and the UCA may result more noisy than in the single-section configuration, so an additional filtering of the signals coming from the DRTs may be required.

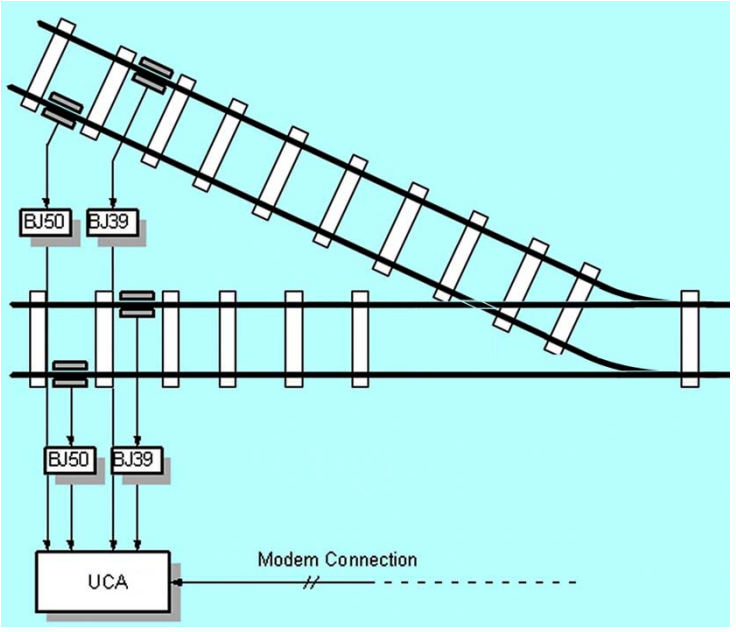

Fig. 2. Single-section SCA with track point.

2.1 Modelling SCA

We now give a brief overview of the SDL specification for this configuration: the model mainly reproduces the entry section UCA subsystem related to the acquisition and processing of the signals coming from the two connected PRAs. Actually, in a component-oriented SDL design, we need to combine

all the replicas of the reading, filtering and axle sequence checking processes which are necessary to handle the data coming from the two PRAs. Figure 3a shows this elaboration chain up to the AnalyzeTableSlot process, which determines if the section is free or occupied by a train, accordingly to the number of axles counted by the PRAs. In the case of a noisy channel between the PRA and the connected UCA, the chosen configuration can be adapted introducing an auxiliary filtering process: in fact, in this SDL model it is possible to insert a new filtering component using the same interfaces between the existing ones; this reflects the real case of adding a series software filter rather than modifying and making more sophisticated, at a higher cost, the existing one. In Figure 3b the model modified with the auxiliary filtering component is depicted. In the following section, we will show a scenario-based validation technique starting from a set of MSCs used to verify the model of Figure 3a. We will also point out how the component-oriented design allows a partial reuse of these scenarios to validate the modified model.

3 Verification of the component-oriented design

A first approach to verify a SDL specification is by *simulation*. In the experience reported in [4], the simulation capabilities of the Cinderella SDL tool were exploited to perform an initial verification of the SCA system of its conformance to the system requirements. During the simulation activity, the system is handled as a black-box, which, following suitable input stimuli, must send the expected output signals to the system environment. Simulation is essentially a testing activity performed at the specification level; in practice, simulation verifies the correct execution of a working scenario, that is a path in the system state space. Instead, a *validation* consists in exploring the state space (often represented by a *behaviour tree*) generated by the processes of the system by executing a suitable algorithm: either exhaustive search until reaching a specified depth, or partial search following various different criteria in the selection of the next state to be explored; validating a system is often conceived as providing the evidence that all the possible behaviours of the system have been verified, in reference to the requirement specifications, and that the system presents no undesired behaviour. Unfortunately, though being more powerful, a validation algorithm often cannot explore all the possible states of a system, since the *state space explosion* phenomenon arises, particularly for complex systems, composed by concurrent state machines as the SCA. Actually, this fact prevents from scheduling all the possible signals from the environment to stimulate the SDL system (this situation is described, in System Theory terms, as a *persistent exciting signal*), particularly when

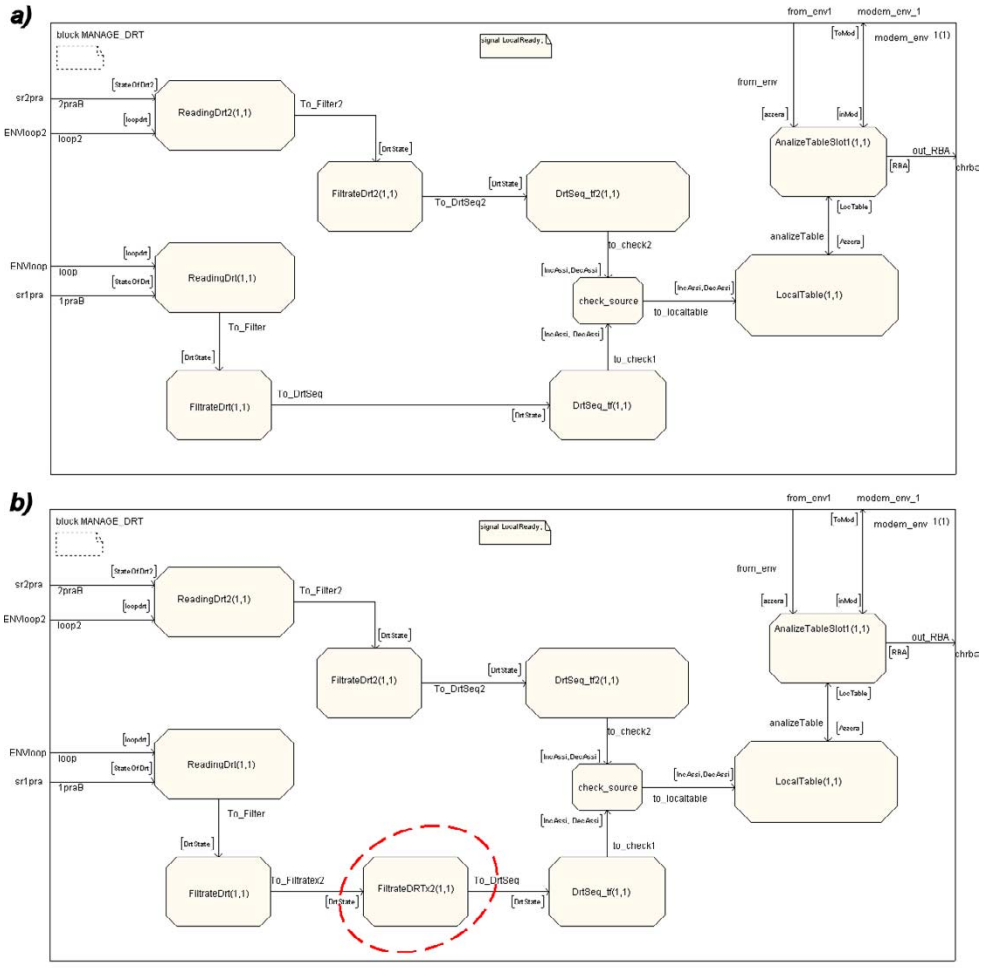

Fig. 3. SCA model without and with the auxiliary filtering component (dashed encircle).

adopting an exhaustive exploration algorithm. Therefore, the validation of a large system like the SCA needs to measure the actual amount of verification activity performed. We can transfer to the validation of an SDL model the concept of *coverage*, which is widely used in the area of software testing to measure the percentage of the system which has been exercised by the performed tests: particular structures of a program are chosen as the unit of coverage, so that the notion of *statement coverage, branch coverage, condition coverage,* etc. . . are commonly used.

In the case of a SDL specification, commercial tools such as *Telelogic TAU* refer to the concept of coverage applying it to the symbols composing the SDL model: a 100% *symbol coverage* is achieved by a validation effort if all the

graphic elements appearing in the SDL descriptions of the system state machines (that is, in the SDL processes) have been exercised. This assures that all the states defined in the SDL processes have been properly validated, but it may be that not all the possible values of the process variables have been considered: in fact, if we consider the system as composed by extended-finite state machines (EFSMs), two system states are different not only because of different states of a process, but also because of different values assigned to the process variables. Symbol coverage hence abstracts from values of variables. Since the whole state space cannot be explored, the validation of the system has been limited to the exploration of the system behaviour close to the working scenarios; therefore, we have used a MSC-driven validation to verify scenarios of interest described using the Message Sequence Charts formalism; the SDL Validator provided in the Telelogic TAU SDL and TTCN suite [8] allows to perform this partial verification technique, by adopting Holzmann's Bit-State algorithm [9], already experienced with SDL specifications [10].

3.1 Incremental and MSC-driven Validation

Due to the component-oriented design of the case study SDL model, the most natural verification technique is to perform an *Incremental Validation* of the system, which consists in validating first each component, then the verified components together, connected in a progressive way, up to reach the final configuration of the system. In order to use the validation for the SCA system in the considered configuration and components, we need to use suitable MSC scenarios to reduce the state space explosion, which is generated by the progressive growing up of constraining interactions between the connected components. We have first considered and validated the configuration of SCA without the auxiliary filtering component (Figure 3a); a set of 53 MSCs has been successfully verified obtaining a symbol coverage of 91,75%. These MSCs can be classified in two groups:

- 10 MSCs representing Black-Box interactions (Input-Output signals of the SDL system with its environment);
- 43 MSCs designed to exercise internal states of the SDL processes.

Validating MSCs of the former group means verifying at the system level the behaviour of the model interacting with the environment, leaving to the validator tool the task of scheduling the transitions of the internal processes; applying MSCs of the latter group is necessary to investigate the internal dynamics and particular non-visible system states from the environment, for example of fail-safe type. Figure 4 shows a MSC of the first type, related to the counting of an axle entering the section; this MSC aims at verifying the

response of the system at a correct input signal sequence from the DRTs, to perform the section occupation (each signal *StateOfDrt* is sampled three times accordingly to the filtering rules of the inputs from the DRTs); the target of the MSC in Figure 5 is instead to verify the system entering the internal fail-safe state *Errors* if an incorrect axle sequence is coming from DRTs.

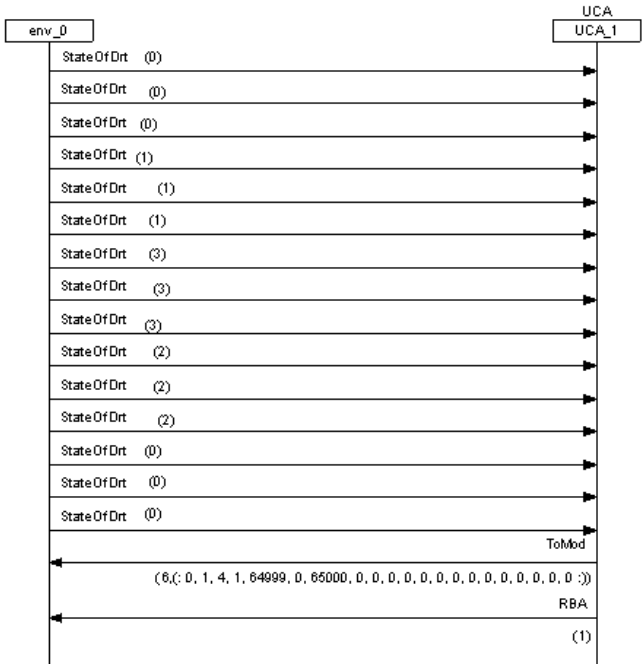

Fig. 4. MSC scenario referred to external I/O behaviour.

3.2 Validation as regression testing after component modification

Adding or modifying a component inside the SDL model requires the new verification of the scenarios described accordingly to the system requirements. We can adopt the MSC verification technique as a suitable regression testing activity on the modified model. Two results are expected after a component insertion/modification:

- the symbol coverage on the SDL model will be reduced;
- some scenarios will not be verified.

These consequences are due both to the insertion or change of the SDL symbols in the model, and to new functionalities introduced in the system.

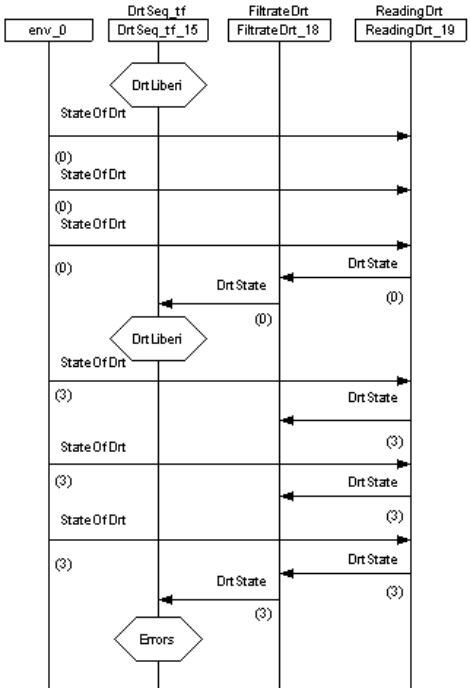

Fig. 5. MSC scenario specifying the internal dynamics related to an error handling.

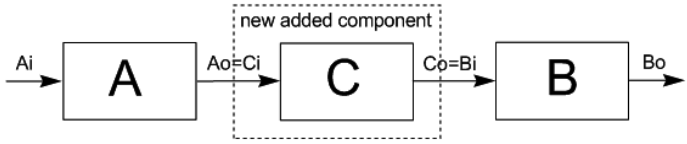

Fig. 6. Interfacing a new component with the other parts of the model.

Moreover, inserting a module in a component-oriented design is based on the rule, sketched in Figure 6, that the new component should conform to the interfaces with the other parts of the model which it will link to, using the same signals communication syntax. In the adopted case study, the insertion of the auxiliary filtering component has produced a decrease of the symbol coverage, which has been resulted equal to 70,76% after the regressive validation on the set of 53 MSCs, which verified the original model. Of the 53 MSCs, 27 have not been verified; in validation terms, it is said that they have been *violated*. Of the 27 violated MSCs, 7 belong to the first group, describing interactions of the system to and from the environment, while the other 20 specify internal behaviours.

We can detect in which parts of the modified model the execution of the violated MSCs has broken off using the *Coverage Viewer* and the *SDL Trace* utilities, offered by the SDL Validator, which can support the re-engineering of the MSCs to obtain the desired symbol coverage.

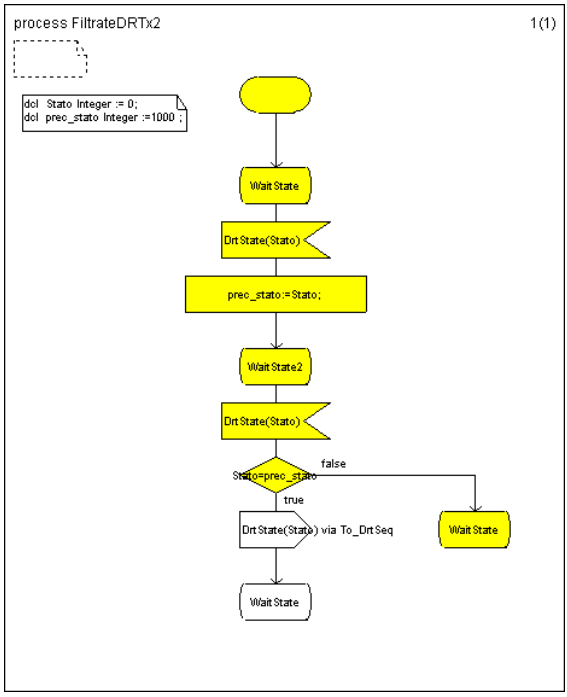

Fig. 7. Partial coverage of the auxiliary filtering component after MSC violation.

Figure 7 shows the partial coverage, evidenced in gray, of a violated MSC applied to the new filtering component; this is due to the fact that the communication semantics of the original scenario, described by the MSC, does not agree with the new component one. Actually, introducing a series component implies the change of the synchronization patterns between the other processes interacting with it.

After having detected which parts of the system result uncovered, using the tools previous mentioned, the following criteria may help in reusing the corresponding no more verified MSCs. In order to define a MSC that can be satisfied by the system in which the new series component has been introduced, we should concentrate on the inputs needed by this component to produce the outputs needed by the next components, as specified by the satisfied MSC regarding the original system: therefore, we need to mark the processes situated in the elaboration chain after the inserted component; then,

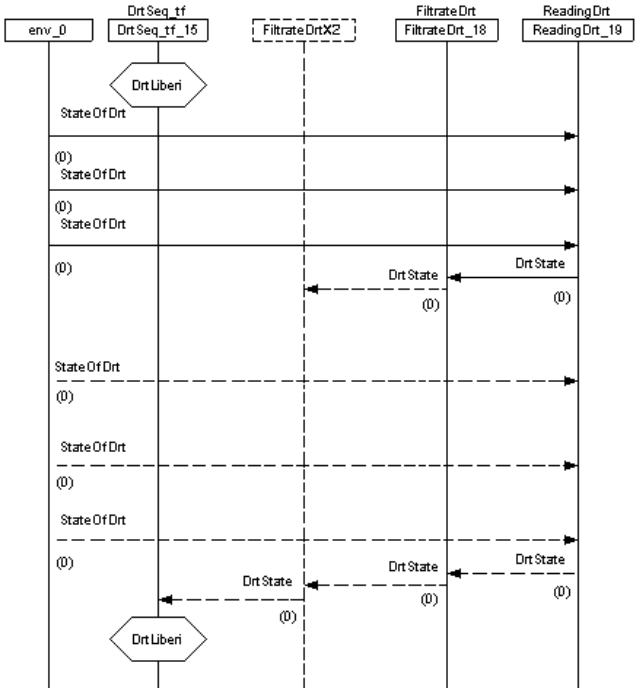

Fig. 8. Reuse of violated MSC: the added component (dashed drawing) should be stimulated in a suitable way to obtain the same behaviour afterwards in the processing chain.

to obtain the same output signals from this processes, that is the same external behaviour, we need to analyze backwards in the processing chain and detect, starting from the processes just before the new component up to the environment, which are the stimuli sequence that lead the new component to produce the signals expected by the marked processes, in order to give the desired output. This is exemplified in Figure 8, which shows the redefinition of the first part of the scenario presented in Figure 5, with the process *Drt-Seq_tf* reaching the internal state *DrtLiberi*: the communication semantics for the original scenario (that is, which signals are required as input to let a process produce a particular output) establishes that the transition in the state *DrtLiberi* is possible if the signal *DrtState* is sent by the process *FiltrateDrt*. In the modified MSC, the presence of the new filtering component *Filtrate-DrtX2* between the the two previous processes, implies that, to produce in output the same *DrtState* signal previously sent by *FiltrateDrt* process and so allowing the transit of *DrtSeq_tf* in the state *DrtLiberi*, two signals *DrtState* must be received by *FiltrateDrtX2*, accordingly to the semantics of the new component. This drives us to duplicate the whole sequence of signals from the

environment (signals *StateOfDrt*) up to the *FiltrateDrt* process. The signals which need to be added in the new scenario are drawn as dashed lines. These guidelines can apply provided we know the communication semantics both of the new component, and of the existing ones: this is possible if we know the internal structure of the component, or if we have the MSCs describing all its possible behaviours; the latter is applicable only for small components. Moreover, we can note that the closer the added component is to the starting point of the processing chain, the lesser the coverage will result in the regression validation, since the MSC will be violated very soon; however, in this case the reuse of the MSCs will result less expensive, since fewer changes will be applied from the new component backwards to the environment.

4 Conclusions

We have discussed an experience in the validation of a system specified in SDL by means of commercial support tools, namely the Telelogic TAU SDL Validator and Simulator. The experience has concentrated on the regression validation of variations of an already validated model, showing how the offered tools can support the evaluation of the coverage of the regression validation performed. In particular, we have addressed the case of a new software component added in a model, and we have studied how the violated scenarios described by Message Sequence Charts can be used to define new simulation scenarios aimed at the full coverage of the new model.
More experimentation is needed to confirm the generality of the approach; in particular, different topologies of the connected components will be addressed in the future work.

As final considerations, we want to underline the benefits added to the automatic code generation by a component-based SDL modelling, since such a design results more accurate than a multiple-configuration one in representing the system which will be downloaded into the target device. Work is going on dealing with the comparison using performance and complexity estimators based on static analysis metrics between the C code generated by the Cmicro code generator of Telelogic TAU and the handwritten version developed in GETS.

Moreover, we can note how the availability of validation scenarios specified by MSCs allows the test-case generation to be performed automatically using suitable tools, such Telelogic TTCN.

Acknowledgements

We want to acknowledge General Electric Transportation Systems for kindly lending the case study.

References

[1] ITU-T, *Recommendation Z.100 - Specification and Description Language (SDL).* Geneva, 1992

[2] ITU-T, *Recommendation Z.120 - Message Sequence Charts (MSC).* Geneva, 1999.

[3] D. Harel, *Statecharts: A visual formalism for complex systems.* Science of Computer Programming, Vol. 8, pages 231-274, 1987.

[4] S. Bacherini, S. Bianchi, L. Capecchi, C. Becheri, A. Felleca, A. Fantechi, E. Spinicci, *Modelling a railway signalling system using SDL.* Proceedings of FORMS 2003 International Symposium on Formal Methods for Railway Operation and Control Systems, Budapest, May 2003.

[5] A. Fantechi, E. Spinicci, *Modelling and Validating a Multiple-configuration railway signalling system using SDL.* Proceedings of TACoS 2003 International Workshop on Test and Analysis of Component Based Systems, Warsaw, April 2003.

[6] J. Bohn, W. Damm, H. Wittke, J. Klose, A. Moik, *Modeling and Validating Train System Applications Using Statemate and Live Sequence Charts.* Integrated Design and Process Technology, IDPT-2002, United States of America, June 2002.

[7] A. Cimatti, P. Pieraccini, R. Sebastiani, P. Traverso, A. Villafiorita, *Formal specification and validation of a vital protocol.* Proceedings of FM'99, Toulouse, France, September 1999. LNCS 1709, Springer-Verlag.

[8] *Telelogic TAU 4.4 User's manual*, Telelogic AB, www.telelogic.com.

[9] G. J. Holzmann *Design and Validation of Computer Protocols.* Prentice-Hall, 1990.

[10] G. J. Holzmann *Validating SDL Specification: an Experiment.* Proceedings of the 9th International Workshop on Protocol Specification, Testing and Verification, Twente University, The Netherlands, June 1989.