Using Games for Improved Diagnosis in Trustworthy Design of Autonomic Systems *

Tiziana Margaria, Christian Wagner, Marco Bakera
Chair Service and Software Engineering
University of Potsdam, Germany
margaria,wagner,bakera@cs.uni-potsdam.de

The GEAR game-based model checker has been used successfully to investigate properties of space systems that must function largely autonomously in inhospitable and far away environments, like the ESA ExoMars Rover. In this paper we summarize the adoption of game-based verification technologies for the long-running Voyager II space mission – long-running in this case means more than thirty years. To this aim, we are currently enabling GEAR's game-based verification techniques via systematic model extraction from a behavioral subset of a DSL for autonomous system specification.

1 Motivation

In the course of the SHADOWS project [SHA, SUM07] we developed a number of enabling techniques for functional self-healing. In particular, we introduced game based model checking of behavioral models as a deep diagnosis technique, that allows an early realignment between behavioral models and requirements expressed as temporal properties [BM]. We first applied it to the analysis of the recovery behavior of the ESA ExoMars Rover, using for this GEAR, our game based model checker for modal mu-calculus [BMRSntb, BMRSnta].

In the concrete mission example examined in SHADOWS, the ESA ExoMars Rover is sent on a surface mission on Mars where it has to accomplish several tasks, including the acquisition of subsurface soil samples using a drill. As customary, the mission is organized in a hierarchical three-tier control model which accounts for partial autonomy of the Rover. *Mission* plans are designed and enforced by the ground control center, while finer-grained operational decisions, at the *task* level, are completely autonomous: the Rover has its own planning capabilities, which allows it to transform a task assignment into a suitable executable sequence of *actions* in a context-dependent and error-aware way. In this case study, we took advantage of the interactive and exploratory benefits of game-based verification technologies. In the case of problems within highly reactive and concurrent systems – as in the context of autonomous aerospace missions – it is in fact hard to auto-

*This work has been partially supported by the European Union Specific Targeted Research Project *SHADOWS* (IST-2006-35157), exploring a *S*elf-*H*ealing *A*pproach to *D*esigning c*O*mplex soft*W*are *S*ystems.

matically find recovery mechanisms to overcome these problems. Even for human system developers it is non-trivial to completely understand the nature of a problem if mismatches between the behavioral specification and the system implementation occur. This is where games provide an added value over traditional model checking.

The weak point was however the lack of a link to an adequate, formal description of the Rover's behavior. We derived our models and properties from the literature (textual descriptions and previous studies) [BJK04, Kap05], while for a stringent demonstration of the techniques and for a validation of the underlying SHADOWS methodology it would have been advantageous to start from real models. This is now the case for the Voyager case study.

2 The NASA Voyager Mission

The NASA Voyager Mission started in 1977 and was designed for exploration of the outer planets of the Solar System. The twin spacecraft Voyager I and Voyager II are still now taking pictures of planets and their satellites in 800x800 pixel resolution, then radio-transmitting them to Earth. Voyager II has two on-board television cameras - one for wide-angle images and one for narrow-angle images - that record images in black and white. Each camera is equipped with a set of colour filters, which help images to be reconstructed as fully-colored ones. Voyager II uses radar-like microwave frequencies to send the stream of pixels towards the Earth. The signal suffers on this distance a 20 billion times attenuation [Bro89]. In [VH09], the mission is specified as an autonomic system composed of the Voyager II spacecraft, four antennas on Earth, and a Command Control Center, all specified as distinct autonomic elements.

We showed in [BWM$^+$] that we can link the behavioral modelling style of our techniques with ASSL [Vas08], a rich domain-specific language for the specification of autonomous systems, equipped with a formal semantics, and that we can easily and systematically translate (parts of) the specification of the Voyager's behavior into Service Logic Graphs (SLGs), thus enabling the application of self-healing technologies to the large class of autonomous systems describable in ASSL. The advantage of SLGs over other modelling styles is that they are closer to the field engineer's understanding, thus making advanced game-based diagnosis features accessible to non-experts in formal methods and models.

The translation of parts of an ASSL specification for autonomic systems into a behavioral model implies mapping the ASSL specific *self-management policy, action*, and *event* parts of the system's description to corresponding counterparts in a behavioral system model that is based on a Service Logic Graph. We applied this translation step to the Voyager II mission case study, opening up several options for verifying issues related to e.g. recovery issues. Having detected the absence of a recovery mechanism upon transmission error within the system specification, we can then leverage GEAR to fix this problem.

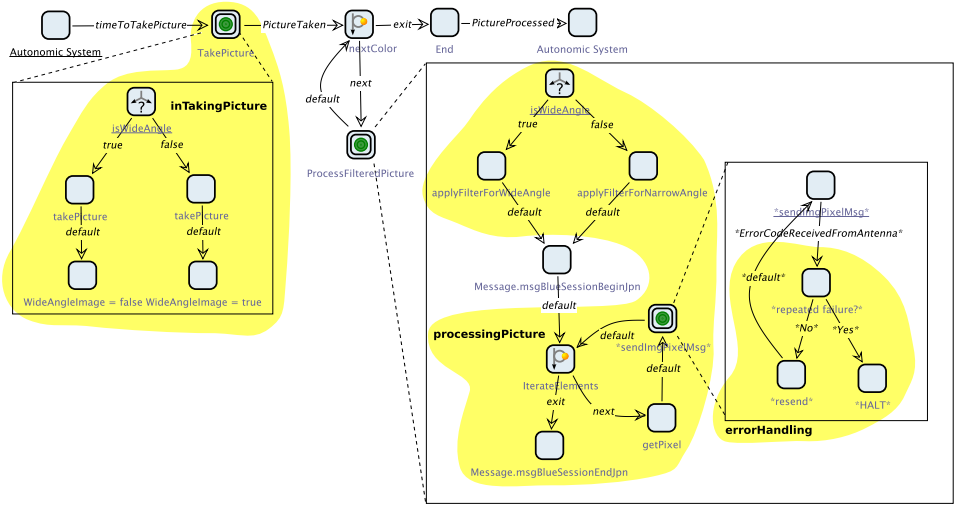

Figure 1: Behavioral model of the picture transmission process. Bottom right: a new error handling recovery mechanism.

3 Verifying the Voyager's Behavioral Model

Figure 1 contains the behavioral model of the Voyager II spacecraft. Note that the error handling graph at the right was not part of the original ASSL specification.

A simple verification issue that immediately emerges is whether the system includes an error-handling process whenever picture pixels are transmitted. This can be easily expressed in CTL [EJS93] as

$$\mathsf{AG}(\mathsf{inProcessingPicturePixels} \Rightarrow \mathsf{EF}(\mathsf{errorHandling}))$$

This formula can be interpreted as follows:

> Wherever the system evolves to (the AG-part), whenever picture pixels are about to be processed (the atomic proposition inProcessingPicturePixels) it follows that the system has an option to evolve into an error-handling process (the EF(errorHandling)-part).

Since the original model of Figure1 does not support any kind of self-healing capabilities, this property does not hold.

Therefore, in a first attempt to reconcile model and property, we added an error-handling routine directly in the model. We slightly changed the design manually, by refining the sendImgPixelMsg action, originally atomic, to an entire routine. Now, if problems during the transmission process occur, the system tries to resend those picture pixels that were not transmitted correctly. If the problem still exists afterwards, the system is halted and needs manual interaction from ground control.

4 Enabling Model based Self-healing

Within SHADOWS, we adopt a model-based approach, where models of desired software behavior direct the self-healing process. A game-based approach can do much more than just allowing the identification of the missing recovery mechanism in the original specification. Enabling this investigation for self-healing and self-healing enactment is our aim. A domain-dependent guidance also enables to pinpoint that part of the model which is best-suited for integration of recovery mechanisms. In particular we exploit the interactive character of game-based model checking to show how to discover an error, then localize, diagnose, and correct it. Design-time healing technologies that naturally emerge when dealing with self-adaptive systems, as in the context of the SHADOWS project, demand for a deeper insight of design-time faults to effectively identify and overcome them.

In general, model checking is used to decide whether an abstraction of a reactive system, modeled, e.g., using a transition system or a Kripke structure, satisfies a requirement, specified, e.g., using temporal logics. In the case of failure, typically error paths are provided as diagnostic information. This is unfortunately not possible as soon as branching-time properties are considered, as their violation cannot be explained in terms of paths. Rather, the diagnostic information has to be generalized to winning strategies of parity games. Parity games are played by two players, both having complete information. Go or Chess are examples of such games. They can be used for game-based model checking as introduced in [LS], which is available for the full modal μ-calculus [Koz82] and thus also applicable to, e.g., CTL and CTL*., which are expressible in μ-calculus. In a parity game, the game graph derived from a model has game-graph nodes partitioned in two sets, one per player. Whenever the game reaches a game-graph node, the player who "owns" that game-graph node has to move to another game-graph node, otherwise he loses the game.

The use of models rather than code is already a significant step towards the understandability of the actual behavior's description to non programmers, like the engineers, in charge of designing a space module. This enables e.g. early discovery of misbehaviors, hazards, and ambiguities via design-time analysis. We strive to improve the diagnostic features making them as detailed as necessary yet as intuitive as possible.

For the Voyager mission, behavioural properties can be used to check for complete picture transmission to the four antennas in case of transmission interrupts. The verification process is able to assure at design time the application of all four color filters before picture transmission. It is also essential for the picture transmission to send closing notification signals of transmission endings to the antennas. These as well as other, more technical business rules can be endured on the design by the mentioned model checking techniques.

If problems occur in the verification task, an immediate result of the game-based algorithm of the model checker is an interactive counter-example. This counter-example both pinpoints the problem of the property mismatch and provides a strategy encoded into the counter-example to adapt and self-heal the system. In its display and interaction capabilities, GEAR [BMRSntb] is in fact tailored for use by engineers. Its rich user interface that allows engineers to interactively explore the problem space in a game-based way, and this way discover and pinpoint the problems in system design.

Acknowledgement

We thank Falk Howar, Alexander Wickert, Bernhard Steffen, Mike Hinchey, and Emil Vassev for the ongoing cooperation, that includes also model learning and other formal-methods based analysis techniques.

References

[BJK04] G. Bormann, L. Joudrier, and K. Kapellos. FORMID: A formal specification and verification Environment for DREAMS. In *Proc. 8th ESA ASTRA Workshop*, 2004.

[BM] M. Bakera and T. Margaria. The SHADOWS Story on Implementation, Verification and Property-Guided Autonomy for Self-Healing Systems. *ERCIM News N.75, Special theme: Safety-Critical Software, October 2008, pp. 38-39.*

[BMRSnta] M. Bakera, T. Margaria, C. Renner, and B. Steffen. Game-Based Model Checking for Reliable Autonomy in Space. *Journal of the American Institute of Aeronautics and Astronautics (AIAA)*, in print.

[BMRSntb] M. Bakera, T. Margaria, C. Renner, and B. Steffen. Tool-supported enhancement of diagnosis in model-driven verification. *ISSE, Innovations in Systems and Software Engineering - a NASA journal, Springer Verlag*, in print.

[Bro89] M. W. Browne. Technical Magic Converts A Puny Signal Into Pictures. *NY Times*, 1989.

[BWM$^+$] M. Bakera, C. Wagner, T. Margaria, E. Vassev, M. Hinchey, and B. Steffen. Component-Oriented Behavior Extraction for Autonomic System Design. *NASA Formal Methods Workshop, April 2009, NASA AMES, Mountain View, CA*.

[EJS93] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On Model-Checking for Fragments of μ-Calculus. 1993.

[Kap05] K. Kapellos. MUROCO-II: FOrmal Robotic Mission Inspection and Debugging. Technical report, European Space Agency, 2005.

[Koz82] D. Kozen. Results on the Propositional μ-Calculus. In *ICALP*, volume 140 of *LNCS*, pages 348–359, Aarhus, Denmark, 12–16 July 1982. Springer-Verlag.

[LS] M. Lange and C. Stirling. Model Checking Games for CTL. In *Proc. of the Intern. Conference on Temporal Logic, ICTL 2000, Leipzig, Germany, October 2000*.

[SHA] SHADOWS. A Self-healing Approach to Designing Complex Software Systems. `https://sysrun.haifa.ibm.com/shadows/`.

[SUM07] O. Shehory, S. Ur, and T. Margaria. Self-Healing Technologies in SHADOWS: Targeting Performance, Concurrency and Functional Aspects. In *10th (CONQUEST)*, 2007.

[Vas08] E. Vassev. *Towards a Framework for Specification and Code Generation of Autonomic Systems*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2008.

[VH09] E. Vassev and M. Hinchey. ASSL Specification Model for the Image-processing Behavior in the NASA Voyager Mission. Technical report, Lero - The Irish Software Engineering Research Center, 2009.