Can Behavioral Requirements be Executed? (And why would we want to do so?)

David Harel
The Weizmann Institute of Science

A novel approach to behavioral requirements for reactive systems is described, in which highly expressive scenario-based requirements are "played in" directly from the system's GUI, or some abstract version thereof [2], and behavior can then be "played out" freely, adhering to all the requirements [3]. The approach, which is joint with Rami Marelly, is supported and illustrated by a tool we have built – the *play-engine*.

As the requirements are played in, the play-engine automatically generates a formal version of them, in an extended version of the language of live sequence charts (LSCs) [1]. The extension includes symbolic instances [5] and time constraints [6]. As behavior is played out, the engine causes the application to react according to the universal ("must") parts of the specification; the existential ("may") parts can be monitored to check for successful completion. See Figure 1, which extends the series of figures appearing in [2], so that the current work is shown incorporated within the conventional framework for the development of reactive systems described in [2].

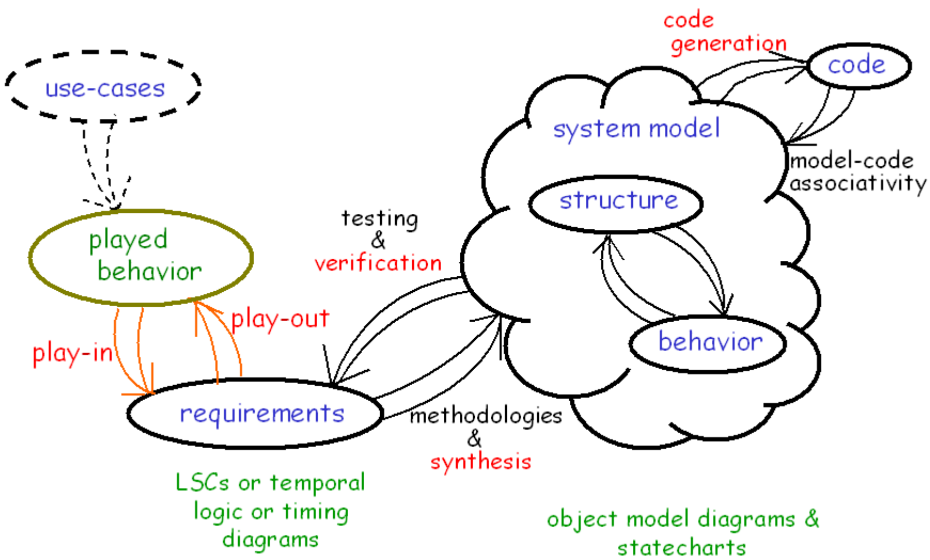

Fig. 1. Play-in/play-out within the development cycle; see Fig. 4 of [2]

We also describe our work on "smart" play-out, joint with Hillel Kugler and Amir Pnueli, whereby parts of the execution are driven by counterexamples produced using model-checking [4]. This makes it possible to avoid taking "bad" nondeterministic choices that may arise from the partial order within an LSC or the various interleavings of several LSCs. Thus, we employ formal verification techniques for driving the execution of the requirements, rather than for verifying the model's properties against those requirements later on. Smart play-out can also be used to find a way to satisfy an existential chart, i.e., to show how a scenario can be satisfied by the LSC specification, and is thus very useful for testing.

The entire play-in/out approach appears to be useful in many stages in the development of complex reactive systems, and could also pave the way to systems that are constructed directly from their requirements, without the need for intra-object or intra-component modeling or coding at all.

A particularly exciting application of the ideas is in the modeling of biological systems, where the information from experiments comes in the form of scenarios or scenario fragments. We are in the midst of a project involving modeling and analyzing parts of the development of the C. elegans worm using the play-engine, and will report on this effort separately.

References

1. W. Damm and D. Harel, "LSCs: Breathing Life into Message Sequence Charts", *Formal Methods in System Design* **19**:1 (2001).(Preliminary version in *Proc. 3rd IFIP Int. Conf. on Formal Methods for Open Object-Based Distributed Systems* (*FMOODS'99*), (P. Ciancarini, A. Fantechi and R. Gorrieri, eds.), Kluwer Academic Publishers, 1999, pp. 293–312.)
2. D. Harel, "From Play-In Scenarios To Code: An Achievable Dream", *IEEE Computer* **34**:1 (January 2001), 53–60. (Also, *Proc. Fundamental Approaches to Software Engineering* (*FASE*), Lecture Notes in Computer Science, Vol. 1783 (Tom Maibaum, ed.), Springer-Verlag, March 2000, pp. 22–34.)
3. D. Harel and R. Marelly, "Specifying and Executing Behavioral Requirements: The Play-In/Play-Out Approach", to appear.
4. D. Harel, H. Kugler, R. Marelly and A. Pnueli, "Smart Play-Out of Behavioral Requirements", *Proc. 4th Int. Conf. on Formal Methods in Computer-Aided Design* (FMCAD 2002), November 2002, to appear.
5. R. Marelly, D. Harel and H. Kugler, "Multiple Instances and Symbolic Variables in Executable Sequence Charts", *Proc. 17th Ann. AM Conf. on Object-Oriented Programming, Systems, Languages, and Applications* (OOPSLA '2002), November, 2002, to appear.
6. D. Harel and R. Marelly, "Time-Enriched LSCs: Specification and Execution", submitted, 2002.