

Design Rationale Management in Concurrent Engineering

Dennis Bahler (drb@adm.csc.ncsu.edu) and James Bowen
Dept. of Computer Science, Box 8206, North Carolina State University
Raleigh, NC 27695-8206, (919) 515-3369, (Fax: 515-7896)

Abstract

The ability to capture and use design rationales, part of what we call Design Rationale Management (DRM), is an important facet of Concurrent Engineering. Galileo2 is a language for constructing interactive design advice systems for Concurrent Engineering, based on a generalization of constraint processing. We briefly describe the features of Galileo2 which make it especially suitable for constructing Concurrent Engineering applications and supporting DRM. We also sketch an example interaction with a typical application program written in the language; this example scenario shows how an advisor written in Galileo2 uses DRM among other methods to support consultation, coordination, and negotiation among multiple engineers who are expert in different aspects of the product life cycle.

Galileo2 is a generic language for constructing constraint-based design advice systems for Concurrent Engineering (CE). Design advisors have been written in Galileo2 for a wide variety of application areas[1, 3, 6]. These advisors essentially “look over the designer’s shoulder,” monitoring user decisions for compliance with CE requirements, inferring the consequences of these decisions whenever possible, and offering suggestions about what to do in the event the user violates a constraint. Galileo2 is especially attuned to supporting coordination between multiple users from different CE perspectives[2, 5]. A paper describing a real-world Design For Testability (DFT) application which uses this technology[8] recently won a national paper competition at AutoTestCon ’91.

To support Concurrent Engineering including DRM, the Galileo2 system provides: frame-like structures with full property inheritance; atomic, compound, and quantified constraints over structured and unstructured objects; conditional existence of frames and scalar objects[4]; heterogeneous attribute value types; the ability to run autonomously on small problems and interactively on larger problems; mixed-initiative interaction; design history recording with recapitulation for purposes of design audit, redesign, reuse, and maintenance; multiple perspectives on the same design task/artifact; user-directed disabling of constraints, provided reasons for these decisions are recorded for later use in design audits and/or negotiation with representatives of other life-cycle perspectives; machine-generated suggestions for design patches; automatic rationale for machine-generated design decisions; explanation of machine-generated requests for user information; support for arbitrary design methodologies and styles; database interfaces together with query capability; and feature-based CAD capture of functional design decisions with machine-generated drawing modifications to be accepted at user option. Newer work has added provision for imprecise/uncertain values and relations and varying degrees of constraint satisfaction[7].

To illustrate how Galileo2 supports DRM, we will consider a design advisor written in Galileo2 to assist in the concurrent engineering of printed wiring boards (PWBs). This program, called KLAUS, supports interaction with several members of the design team, including designers, manufacturing engineers, and test engineers. We will briefly see how the differing perspectives of each team member are supported, and how KLAUS provides the capability for each representative of the CE team to record decisions and the rationale behind them so as to be recapturable by other team members.

Several points deserve emphasis about this scenario. First, this scenario presents only one of very many possible orders of interaction with KLAUS. Second, we show only a few steps in this interaction. Finally, although the figures in this paper are not actual screen dumps, KLAUS and Galileo2 are fully implemented, and these figures faithfully represent the various interfaces presented to the manufacturing, testing, and design users.

In the following scenario, we assume that the project leader has set up a database entry for a new project. We take up the story when the test engineer, who in this case happens to be the first team member to make

some decisions about this new project, starts to interact with KLAUS about the project. Figure 1 shows the interface presented by KLAUS to the test engineer after he has selected the test equipment to be used for the project. The largest window in this screen is a single-column spreadsheet, or “scrollsheet,” in which each cell occupies one or more lines.

[]Help	[]File	[]New	[]Utilities	[]Search
[]Up	[]Down	[]Focus	[]Toggle	
[]the name of the equipment at the facility where the board will be tested	erdsys			
[]the maximum clock frequency testable at the facility where the board will be tested	9.8			
[]the maximum number of test points testable at the facility where the board will be tested	200			
>>>				
KLAUS - a PWB Design Advisor (Testability)				

Figure 1

Suppose the manufacturing engineer also interacts with KLAUS before circuit design begins. Figure 2 shows the results of his decisions about the drilling and assembly equipment to be used. Notice that the parameters representing the number of chosen hole sizes and the list of chosen hole sizes are not yet bound because their values depend on choices not yet made by the board designer.

[]Help	[]File	[]New	[]Utilities	[]Search
[]Up	[]Down	[]Focus	[]Toggle	
[]the name of the drilling machine	bransch2			
[]the magazine size of the drilling machine	2			
[]the list of standard drill sizes for the drilling machine	{0.04,0.045,0.05,0.055,0.06,0.065}			
[]the list of chosen hole sizes				
[]the number of chosen hole sizes				
[]the name of the assembly robot	yotsui19			
[]the insertion oversize needed by the assembly robot	0.016			
>>>				
KLAUS - a PWB Design Advisor (Manufacturing)				

Figure 2

Figure 3 shows the perspective of the circuit designer. Notice that this perspective provides a second type of interface, feature-based CAD, rather than the scrollsheets provided to the test and manufacturing engineers. The drawing in this screen represents the circuit after (i) the designer has introduced a CPU, (ii) KLAUS has used its constraint information to induce automatically the need for an associated pullup resistor and oscillating crystal, and (iii) the designer has decided to accept these suggested components as part of his design.

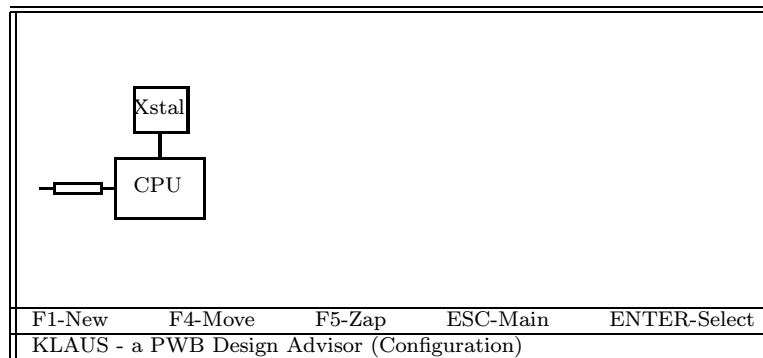


Figure 3

Suppose that the designer, having accepted the crystal, specifies that it should oscillate at 25 Mhz. Now, however, the following constraint in the KLAUS program comes into play:

```
all X : osc_crystal(X) and test_facility.maxfreq < X.freq
implies
exists(X.'an ancillary divider for'(ad) : dvdr).
```

and, because 25 Mhz exceeds the maximum testable frequency of 9.8 Mhz specified earlier by the test engineer (Figure 1), the system introduces an ancillary divider circuit for this oscillator. The result can be seen in Figure 4, where KLAUS is suggesting that the new component on the screen should be added to the circuit.

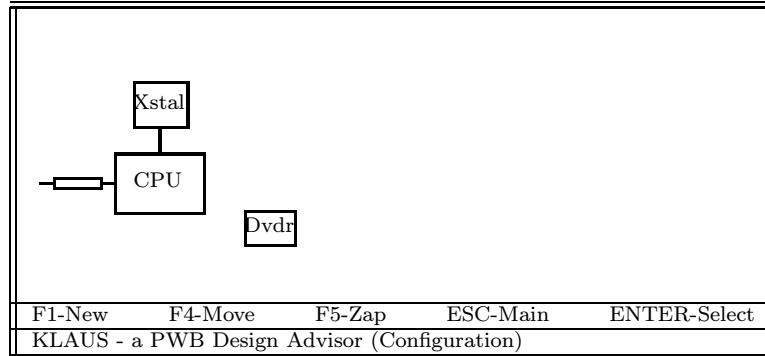


Figure 4

The designer is surprised by the introduction of this component, so he asks KLAUS to explain its rationale. In KLAUS's rationale (Figure 5), the constraint given above, which introduced the component, is paraphrased in natural language; note the use in this paraphrase of the natural language synonym "an ancillary divider for" defined in the constraint for the field "ad."

Rationale
<p>an ancillary divider for the oscillator for the cpu exists because of the following constraint</p> <p>(38) every crystal oscillator must satisfy the following: if the maximum clock frequency testable at the facility where the board will be tested < the oscillation frequency of the crystal oscillator then an ancillary divider for the crystal oscillator must exist and must be a frequency divider;</p> <p>and because of the following parameter value(s): the maximum clock frequency testable at the facility where the board will be tested = 9.8; the oscillation frequency of the oscillator for the cpu = 25.</p> <p>the maximum clock frequency testable at the facility where the board will be tested was established according to the perspective taken by test engineers.</p> <p>the oscillation frequency of the oscillator for the cpu = 25 because you said so.</p>

Figure 5

Despite this rationale, the designer decides to reject this ancillary component. Now, however, the constraint which wanted to introduce the ancillary divider is violated, leading to the message shown in Figure 6.

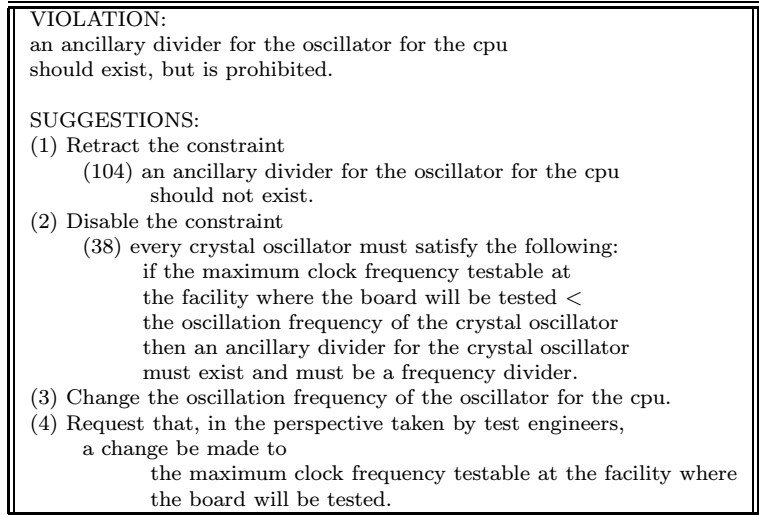


Figure 6

Choosing among the suggestions offered in this message, the designer decides to disable constraint (38). However, this constraint also refers to a parameter in the test perspective, so the decision to disable the constraint must be accepted by the test engineer. Whenever a user disables a constraint other than one he previously asserted himself, he is required to enter a free-text rationale for his action, which is saved for possible use in a design review. These explanations are also used as the basis for system-mediated interaction between users of different perspectives when they make conflicting decisions. This interaction may take place via interactive workstation-to-workstation communication, email, or face-to-face.

When the test engineer next logs into KLAUS, he is told that a constraint of interest to him has been disabled by another user. Checking on this, he calls up the design state from the database. The system tells him which constraint was disabled and produces the free-text rationale given by the circuit designer.

The test engineer decides that he is unwilling to allow this constraint to be disabled because of the difficulty in testing that would result. However, to compromise, he changes the test equipment to one which is able to handle a frequency of 25 MHz. After making this change, the test engineer reactivates the disabled constraint (38). Because of the higher frequency testable by the test equipment, the re-enabled constraint does not attempt to re-introduce an ancillary divider circuit, so no constraint violation occurs. The test engineer saves the new design state, together with its rationale, and starts to work on another project.

When the circuit designer next logs in, he is told that the test engineer has re-enabled the constraint but, since the unwanted divider circuit has not reappeared, the designer is content.

Suppose, however, that no such easy compromise was possible. For example, the designer might have selected an oscillation frequency that exceeded even the upper limit of the fastest available tester. In this case, the test engineer and designer will successively disable and re-enable their shared constraint (38), offering free-text rationales to each other until one or the other gives way or appeals to the project leader. In this case, the test engineer could give way by deciding to build a special divider test fixture; the circuit designer could give way by using a lower oscillation frequency. In either event, these decisions and the underlying rationale would be communicated to the rest of the team in a similar manner.

We recognize that this simple iterative mediation constitutes only the beginning of a facility for design rationale capture and use. Indeed, more comprehensive support for rationale management constitutes a major task in our ongoing research.

References

- [1] Bahler, D. and Bowen, J., "Intelligent Software Design Using Generalized Constraint Processing," *Automating Software Design: Interactive Design*, Workshop Notes, Anaheim, CA, 1991.
- [2] Bowen, J. and Bahler, D., "Negotiation in Constraint-Based Advisors for Concurrent Engineering," *Proc. 2nd International Conf. on Artificial Intelligence in Design*, Pittsburgh, June 1992.

- [3] Bowen, J., and Bahler, D., "Frames, Quantification, Perspectives, and Negotiation in Constraint Networks for Life-Cycle Engineering," *International Journal for Artificial Intelligence in Engineering*, 1992 (to appear).
- [4] Bowen, J., and Bahler, D., "Conditional Existence of Variables in Generalized Constraint Networks," *Proc. 9th Natl. Conf. on Artificial Intelligence (AAAI-91)*, Anaheim, 1991.
- [5] Bowen, J., and Bahler, D., "Supporting Cooperation Between Multiple Perspectives in a Constraint-Based Approach to Concurrent Engineering," *Journal of Design and Manufacturing 1*, 1991, 89-105.
- [6] Bowen, J., Bahler, D., and Dholakia, A., "An AI Constraint Network-Based Approach to Bed-of-Nails DFT for Digital Circuit Design," *Computers in Electrical Engineering*, 1992 (to appear).
- [7] Bowen, J., Lai, R., and Bahler, D., "Lexical Imprecision and Fuzzy Constraint Networks," *Proc. 10th Natl. Conf. on Artificial Intelligence (AAAI-92)*, San Jose, July, 1992.
- [8] Dholakia, A., "RICK: A DFT Advisor for Digital Circuit Design," *AutoTestCon '91*, 1991. **Winner, National Student Paper Competition.**