Safety-Critical Systems 4:
Engineering of Embedded Software Systems

AG Betriebssysteme, FB 3
Jan Bredereke

WiSe 2002/03

issued: Jan. 15, 2003
due: Jan. 21/23, 2003

Assignment 9

Parsing Dates – Abstract Interface Design

This assignment could be about an abstract interface to a hardware device in a safety-critical application, such as avionics or railway signalling. Since we have no real experts for such hardware at hand, and since these devices tend to be complex, we choose another example about which we know well and which is sufficiently simple.

There is no standard format for representing dates. Among the many ways of representing dates are:

February 10, 1941
10 February 1941
10 February 41
10.2.1941
2/10/1941
41.2.10
41 February 10
41,41

Design the abstract interface for a module that converts a string with a date into an internal representation used by the rest of the software.

1. Make a list of assumptions that is true for all possible actual interface modules.
2. Specify access programs that exactly embody these assumptions.
3. Send these two descriptions for review to Jan Bredereke and iterate.

In addition to the descriptions, document how and where you looked to find out about date formats and about date uses.

Your first draft will not be marked. Instead, your final version and in particular your success in finding errors in earlier versions will be marked. You must perform at least one iteration, and you can do more. You will receive reviews within at most two work days. The first draft is due on Tuesday January 21. If necessary, you might negotiate the due date for the final version.

When you send a version for review, prepare an active design review. Write questions to the reviewer about your design. The process of the reviewer answering these questions shall check whether the following design properties are given (they are desirable for *any* interface module):

- Well structured. The design should be consistent with chosen principles, such as the principle of information hiding.

- Simple. The design should be as simple as possible, but no simpler.
- Efficient. The programs provided by the design should be computable with the available computing resources and the design should not interfere with meeting space and time requirements.
- Adequate. The design should satisfy present requirements.
- Flexible. The design should make it as easy as possible to respond to requirements changes.
- Practical. Module interfaces should be sufficient for the job, neither providing unneeded capability nor requiring the user to perform functions that properly belong to the module being used.
- Implementable. The functions offered by the design should be theoretically computable with the information available.
- Standardized. The design should be represented using a standard organization for design documentation.

Don't write questions that can be answered with yes/no. You will never know whether the reviewer really checked, or whether he just chose the easy way. The questions should "test" your design documentation in a similar sense as software is tested by executing test cases. (Of course, the amount of execution effort must be limited. You cannot request to implement several versions of the module. Complete "test" coverage is not possible anyway.)

Document why your active review questions cover all the design properties reasonably well.