Assuring the Safety of User-Configured Software

Notes from Just-in-Time session held at the 26th International System Safety Conference, Vancouver, on 27 August 2008.

The JIT topic was proposed by James Inge to investigate views on how the safety of a system could be assured, if the user had the ability to alter its configuration. This reflects a concern that the more ability a user has to configure a system and define its behaviour, the less control the designer has, and the less assurance is available as to whether the deployed system will function safely. The session participants were a mixture of Canadian, US and UK delegates. Unfortunately a list of participants is not available.

User-configured Software

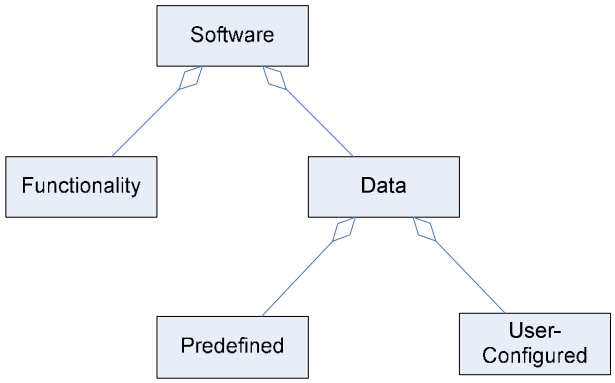


Figure 1. Breakdown of a railway automation application

Data was generally seen by participants as a part of software, as shown in Figure 1. It could either be predefined, or user-configured. Predefined data would typically be produced by the system developer, either within the software development lifecycle, or as part of the design and commissioning effort of adapting generic software to a specific installation. In either case, generation of the data would be handled within the developers' configuration control and design review processes. The system would then be tested as a whole, with the predefined data included.

Some participants used checklists to aid the customisation process, to guide them to complete all the necessary changes and checks required to specialise a generic system for a specific application.

Participants expressed a preference for configuration data to be predefined, or factory-settable, rather than user-configurable, to simplify the testing and proving of a system. Ideally, the system would be demonstrated to be safe with all possible combinations of user-configured parameters and environmental conditions, but this was recognised not to be tractable in many real-world systems.

Where the system was to include user-configured data, the key strategy cited to address assurance was to limit the number of degrees of freedom given to the user, to reduce the testing burden. This could be done by both restricting the number of user-variable parameters and restricting the possible user inputs, e.g. by allowing the user to select only "known good" values, rather than allowing continuous variation.

The idea of constraints could be extended by runtime checking, to give assurance that user-configured parameters were correct after they had been entered. This could range from

checking simple value-based constraints (such as ranges), to more complicated consistency checks involving multiple parameters or historical data. How much configuration checking is viable will depend on how often the configuration data is expected to change and the system's requirements for real-time operation.

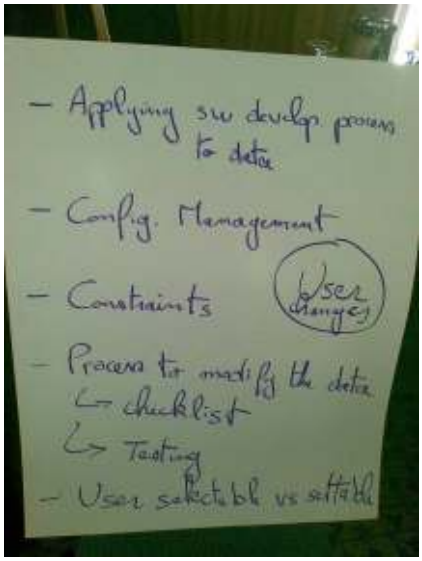
User-modified Hardware

A potential hazard was identified where a preconfigured system was changed after development, without the configuration data being amended accordingly. Although the data would not have changed, it could nonetheless have become incorrect.

User-generated Input Data

The discussion broadened from systems with modifiable settings to systems that use more general user-provided data to define their behaviour. In this case, the data being used by the system is expected to change regularly. It is also likely to be produced outside the control of the developer, and without the benefits of design reviews or configuration control.

Data in such a system might be produced by third party software (including Commercial Off-The-Shelf (COTS) software), or might be supplied as COTS data in its own right. As well as issues of provenance and correctness, this raises questions about the compatibility a data formats produced by different products. No particular best practices were identified for these types of system. However, some participants believed that Open Systems, with publicly available interface standards, could help enable safe use of data from diverse sources.



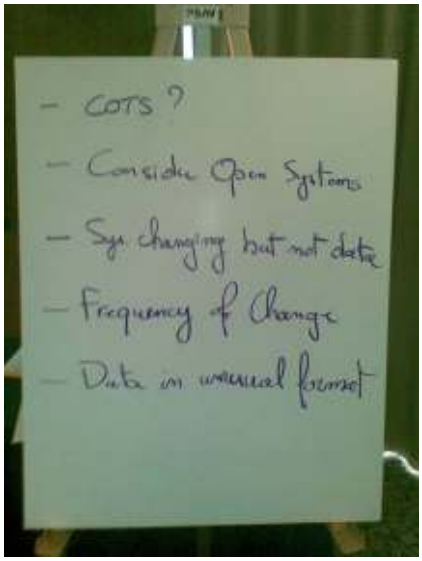


Figure 2. Just-in-Time Session flip chart notes

James Inge
Bristol, UK
6 Oct 2008