Interlocking Design Automation

The Process
Introduction

Imagine an infrastructure manager in need of a new rail control system; maybe a new line is to be built, extended or re-signaled to increase capacity or to solve issues with aging equipment requiring more and more maintenance. This is a complex endeavor, with many subsystems that needs to be integrated and adapted to meet the specific needs of the particular railway or metro.

One key component of the overall system is the software controlling and monitoring the rail traffic using physical signals, switches, automated train control and protection, to ensure safe operation. This vital software - the brain of the rail control system - is often surprisingly difficult to develop and achieve safety approval for, resulting in delays and complications in the commissioning phase of the project.

There are a number of root causes for this situation:

- The requirements are vague, often already in the tender specifications from the infrastructure managers.
- Design, test and verification methods are manual, relying heavily on the skills of individual senior engineers, rather than on a well defined process supported by software automation tools.
- Safety verification is not integrated with the development process, increasing the risk for rejections of deliveries.

We will show that these problems can be avoided as long as proper care is taken from the very start of the project. We will present a process that gives significant benefits for both the supplier and the infrastructure manager.
The Process

In any complex software development project, the key to success is to not only know what you want the system to do, but also to be able to specify this in a clear and precise way so that this is known to all parties involved. Normally the input for creating this specification comes from the end customer, in this case the infrastructure manager, as a set of requirements that are refined together with the system supplier.

A common cause of project delays, divergence in functionality and exceeded budgets is that proper care has not been taken in defining the end user requirements. In the end, when the system is delivered, it is revealed that even though it satisfies its specification, it does not meet the expectations of the customer. To help avoiding this situation, two simple considerations can be made:

1. Do not underestimate the effort needed to define the specification
2. Ensure that the right set of tools for evaluation and verifying the requirements are available already at the start of the project.

The process that we will present here is centered around the specifications and software tools for developing these specifications. But it does not end there. With properly defined specifications, using this process, it is also possible to automate the majority of the development including implementation, testing and safety approval. This will not only guarantee that the delivered system meets the specification and the customer expectations, it will also significantly reduce the time and effort needed to develop and maintain the system, resulting in a drastically reduced life cycle cost. How you work in this process is not fundamentally different to how most infrastructure managers and signal suppliers are working today, but it brings a new level of focus to the requirements and the tools that support you along the way so that you do not deviate from your intended process.
This Interlocking Design Automation process is built around three main concepts:

- Formalized specifications, defined in the PISPEC language.
- An application engineering tool suite, Prover iLock, supporting the development of PISPEC specifications, and automating the coding, testing and verification of the application software.
- Automated sign-off verification, generating the input for the safety case of each developed application, using the Prover Certifier tool.

The same process can be used for the development of software for many different signaling systems, but for the case of simplicity we will restrict ourselves to interlocking systems.

Prover's implementation of this process is called Prover Trident.
Developing Requirement Specifications

We have argued that the key to a successful project that will produce on-time, high-quality deliveries, lies with the specifications, and it is now time to look into the details on how you can go about developing your specifications to ensure success.

First of all we need to bring some structure to the specification work, and we present such a structure that has been proven successful in real life projects, and also explain why this is the case.

We recommend splitting the requirement specification into the following parts:

- **Object Model, OM**, describing the types of objects like tracks, points and signals, that are to appear in the interlocking in terms of inputs, outputs, states and relations to other objects, ensuring that design, safety and test specifications all refer to the same things.

- **Generic Design Specification, GDS**, describing the design principles, or the functionality, that the different object types are to provide.

- **Generic Test Specification, GTS**, describing the expected behavior of the specific application software in terms of test cases.

- **Generic Safety Specification, GSS**, describing the safety requirements that the specific application software is to satisfy.

- **Generic Application Configuration, GAC**, describing how a Specific Application is to be configured.

Having a specification with this structure means that it is easy to have several teams working in parallel, to reduce time to market, but also to have a clear separation between the team responsible for the safety and the team responsible for the design. It may seem obvious that it is a good idea to separate the safety requirements used for the safety case from the design requirements that tells us how to implement the system, but experience shows that such a distinction is often missing, which makes it much more difficult to formulate the safety argument.

Also, we need to be able to test and evaluate the requirements as early on as possible, to avoid spending time on requirements that will later on, perhaps during system level tests, be discovered to be badly chosen. In order to accomplish this we propose to formalize, or implement, the requirements in a language that can interpreted automatically by software tools.
In the Prover Trident process, this language is PiSPEC and the software tool suite is Prover iLock. Doing this formalization in parallel with the work of defining the specifications will enable the teams to quickly try out the impact of different choices in the design and to show this to the end-customer. The use of formal specifications and automation tools is not only a way to simplify the specific application development, it is also a great help when defining the generic application.

Prover iLock enables, and encourages, a test-driven approach to develop your specifications:

- Define, and formalize, object model, design, test and safety requirements in parallel
- Use reference applications to validate the specifications, with simulation, formal verification and code generation

Using Prover iLock during the development of the specifications in this way will also help you ensure that you stay on the right track, sticking to the proposed structure, and that requirements are always kept up to date. You will get a method to do automatic regression testing on your specifications, not only on the final applications. Often specification engineers working with the process for the first time are struck by not only how easy it makes their work, but also how much more interesting and rewarding it becomes when you already from the very start of the project get to try out the requirements and see how they work together.
Specific Application Engineering

With the requirement specifications developed as outlined in the previous section, the task of developing the application software for specific applications (interlockings) is reduced to simply configuring the application and invoking the tools.

Configuration
For a typical interlocking system the configuration starts with the track layout, including objects such as signals, track circuits, switches and level crossings. This data is either entered into Prover iLocks graphical user interface, or imported from a CAD file. Additional configuration data may be needed depending on the specifics of the application, as described in the Generic Application Configuration specification, and is typically entered in a tabular format. Examples of such data include:

- Approach locking limits
- Control Lines
- I/O assignments and zoning

Prover iLock
With the configuration in place, Prover iLock will create the specific interlocking requirements from the generic specification, including:

- Logic, to be exported in a source code format for the chosen interlocking platform, using Prover iLock Coder
- Safety requirements, to be formally verified with Prover iLock Verifier, creating a safety verification report
- Test cases, to be executed with Prover iLock Simulator, creating a test report

In addition to the above, the Prover iLock Documenter can be used to export additional application documentation, such as control tables and test instructions for installation tests.
Safety Approval

The process for the safety approval consists of two main parts:

1. Approving the Generic Safety Specification, GSS
2. Approving each Specific Application

The GSS is approved by reviewing that all safety aspects have been captured correctly by requirements in the natural language specification, and that these requirements are formalized correctly. This is a manual task, though some tool support is provided, that relies on traditional signaling skills, but it is a non-recurring task, meaning that the total review effort compared to a traditional process is reduced.

The approval of each Specific Application is a more automated process, where the key part of the safety evidence is the formal verification, proving with 100% coverage that the safety requirements in the GSS are satisfied by the generated application software. This verification is done outside of the Prover iLock tool, using a CENELEC SIL 4 certified tool chain based on Prover Certifier. The input to this sign-off verification is the application software, the GSS, and the safety critical part of the application configuration. The only manual step in this process is to review that the application configuration is correct.
With a signaling system developed using a process like the one proposed here, you will find yourself in a good position for cost efficiently maintaining your system for a long time. You will have easy access to detailed documentation not only on how the system functions and operates but also how it is designed, and why it is safe. This can be invaluable when you are to make an update some 10 or 15 years down the road, when the principal designers of the system may have moved on from your organization.

With the automation tools it is easy to implement approve and updates even when you have a large install base of interlockings to maintain. They will also help you evaluate proposed changes, and quickly analyse if a proposed design change has the desired effect. For example, will a reduction of headway give the desired increase in capacity for your metro or railroad.
Summary and Comparison with a Traditional Process

Traditionally development of rail control software has been a mostly manual process, where requirements often are detailed and documented by the use of typical designs and templates. This originates in the design of the relay-based rail control systems, an environment in which many of the rail control software engineers often have their background. The concept of a generic application that defines reusable requirements from which specific applications can be derived has been winning ground, as has formal safety verification techniques, and the processes may be partially automated, but still most projects and processes relies heavily on manual engineering efforts and skills.

The key benefits of the process presented here are:

• Predictable development and delivery schedule
• Reduced calendar time and manpower needed to produce the deliverables
• Safety evidence based on formal verification with 100 % coverage
• Requirements are properly documented and defined, easily accessible throughout the system life cycle
• Reduced costs for maintenance and upgrades
If you want to learn more about the Interlocking Design Automation process, you can download several other white papers about it on Provers website www.prover.com. There you will also find information and testimonials from both suppliers and infrastructure managers that are using this process. For more information about the economic benefits with Interlocking Design Automation, consult Provers white paper “The Business Case”