An Automated Development Process for Interlocking Software that Cuts Costs and Provides Improved Methods for Checking Quality and Safety

Authors:

Claes Malmnäs
Prover Technology
Rosenlundsgatan 54
118 63 Stockholm
Sweden
clas.malmnas@prover.com
Phone: +46 8 617 68 13
Fax: +46 8 653 69 00

Gunnar Smith
Prover Technology
Rosenlundsgatan 54
118 63 Stockholm
Sweden
gunnar.smith@prover.com
Phone: +46 8 617 68 23
Fax: +46 8 653 69 00
**ABSTRACT**

When developing a safety critical application like a computerized railway interlocking, one major challenge is to maintain cost efficiency while meeting the strictest requirements on safe operation. This paper describes how this challenge can be overcome using the Prover iLock Process and its adaptation to North American interlocking systems based on AREMA signaling principles, the Prover iLock for American Railroads solution. The process significantly reduces development costs by providing automated tool support for simulation-based functional testing, formal safety verification, and automatic code generation targeting leading interlocking system platforms. The solution runs on a normal desktop computer, and makes it possible to configure, code, test and verify a system in less than a day.

The Prover iLock for American Railroads solution has been developed by Prover Technology in close collaboration with Canadian Pacific Railway, who will deploy it as its standard development process for interlocking software. The solution is based on the AREMA specification library, providing pre-defined, but adaptable, specifications of design, testing and safety principles. Both vital and non-vital software for the VHLC, ElectroLogIXS and Microlok II platforms are supported.
INTRODUCTION

In North America, the de-facto standard for railway interlocking system design is set by AREMA. The quality and safe operation of railway interlocking systems is ensured by developing them following the practices recommended by AREMA. In addition to these practices each railway operator typically has its own set of rules and guidelines adapted for their specific needs.

This paper describes the challenges typically faced when developing interlocking systems for the North American market and proposes a highly automated and cost efficient solution – the Prover iLock for American Railroads solution. This solution is based on Prover Technology’s standard Prover iLock Process, which has been adapted for the needs of North American railroads in collaboration with Canadian Pacific Railway.

CHALLENGES AND OBJECTIVES

It is an obvious objective of any railway operator to have interlocking systems that are well designed. While it might be difficult to define exactly what that means, it should be possible to agree on that a well designed system has the following characteristics:

- **Maintainability**: it is easy to maintain, for instance when the functionality needs to be updated
- **Consistency**: it is designed the same way as other systems for the same operator
- **Verifiability**: the requirements are well defined and it is easy to verify that they are met
- **Traceability**: should an error occur, it must be easy to track down its source, so that appropriate actions can be taken to ensure that the same error does not occur again

In order to produce well designed interlocking systems within reasonable time, it is necessary to have a good development process. The main ambition of such a process is to be a cost efficient way to produce systems that meet the highest quality and safety standards. In order to achieve that we believe that the process needs to:

- support early detection of design errors
- minimize the manual effort needed for acceptance and safety testing
- help the signaling engineers in their daily work, rather than being a burden
- make it easy to handle changes in the requirement specification

© 2011 AREMA ®
• minimize the dependency on undocumented know-how
• leverage from the fact that a potentially large set of similar systems with the same set of requirements are to be developed
• be well documented

The process described in this paper automates many of the manual tasks for coding and verification and validation (V&V) that are central in many traditional processes. Automation has two major benefits; it saves time and reduces the risk of manual errors. A key element in almost any development process is the management of requirements, since without clear and complete requirements it is difficult to verify that the system does what it is supposed to do. Clear and unambiguously defined requirements are also a key factor in order to achieve automation.

**Requirements Specifications**

The requirements on an interlocking system can be divided into two categories: *generic requirements* and *specific requirements*. A generic requirement applies to a set of interlocking systems, typically to all systems of a particular railway operator, authority or region. An example of a generic requirement is a condition that must hold whenever a route is cleared:

*A route must be cleared only if every switch in the route is proved in the position required by the route.*

A specific requirement applies to a single interlocking system. An example of such a requirement is the configuration of the actual routes in the system:

*Route 500S-499N requires that switches 1A and 1B are normal and that switches 2A and 2B are reverse.*

A clear and complete generic requirements specification gives a precise description of the functionality of a range of systems and can be used when validating that the developed system meets all requirements. Unfortunately, in reality the quality of generic requirements specifications is often unsatisfactory. Although requirements often are captured in documents or perhaps even in a requirements management tool, they are typically not complete, and the successful development relies on implicit requirements, existing only in the minds of the engineers. As a consequence, it is hard to guarantee that systems are designed in a consistent way. Important design choices are left
to the individual engineers, resulting in systems that may very well function correctly (possibly after extensive testing and debugging) and according to the specification, but that are undocumented and expensive to maintain.

So, what can be done to ensure that the generic requirements are precise enough and that the developed systems meet all of them? It is the aim of the following sections to provide an answer to this question.

**PiSPEC**

Generic requirements specifications are typically written in natural language (English, Spanish et cetera). Natural language may seem easy to understand at first glance, but experience shows that it tends to be interpreted differently by different people. An efficient way to resolve possible ambiguities in requirements is to instead express them in a formal language. A formal language is a mathematical language, in which every statement has exactly one, well defined, meaning, much as in any well-defined programming language.

PiSPEC is a formal language designed for signal engineering. Using PiSPEC, it is possible to develop a *Generic Application* that formalizes the requirements of a generic requirements specification. The PiSPEC language is built on language constructs that are well known within the computer science community, such as predicate logic and object orientation.

PiSPEC is used in the *Prover iLock Process* to develop three types of specifications:

- **Generic Safety Specification (GSS)**. This specification defines the safety requirements. Safety requirements are used to verify that the system is always safe, for instance that two opposing signals may never display permissive aspects at the same time.

- **Generic Test Specification (GTS)**. This specification defines the functional requirements. Functional requirements are used to test that the system operates correctly, for instance that it is possible to clear signals. These requirements are expressed as generic test cases.

- **Generic Design Specification (GDS)**. This specification defines the design requirements. Design requirements specify how application software shall be implemented and are expressed as generic equations.
The generic requirement example from the previous section, “A route must be cleared only if every switch in the route is proved in the position required by the route”, belongs to the safety specification; it would not be safe to clear a route over a switch which is out of position. Figure 1 illustrates how this can be formalized in PiSPEC.

\[
\text{ALL } rt: \text{ROUTE} \{ \\
\text{rt.clear } \rightarrow \\
\text{ALL } sw:rt: \text{normal\_switches} \text{ sw.detection\_normal } \& \\
\text{ALL } sw:rt: \text{reverse\_switches} \text{ sw.detection\_reverse }
\}
\]

**Figure 1**, a PiSPEC example, ‘\(\rightarrow\)’ denotes logical implication and ‘\(\&\)’ logical and.

The formalization is expressed as a requirement that most hold for all routes that are cleared. In order to make this example complete the predicates (or functions) `clear`, `normal_switches`, `detected_normal`, `reverse_switches` and `detected_reverse` also need to be defined.

While specifying requirements in a formal language has values in itself; for instance that it becomes necessary to resolve inconsistencies and ambiguities in the generic specifications, the real benefit with the PiSPEC specifications is that they can be processed by a computer and used for automatic generation of code, tests and safety requirements for an interlocking system; a Specific Application. In order to facilitate this, the specifications also include generic rules for how to instantiate the requirements for a specific application.

**PROVER iLOCK**

Prover iLock is a software product suite for engineering of interlocking systems. Application engineers input application specific data such as track layout and route and aspect charts in Prover iLock. Based on this high-level data and the generic PiSPEC specifications, Prover iLock generates application specific safety, test and design specifications. The application specific specifications are then used by Prover iLock for automatic safety verification, functional validation (simulation), and code generation.

Prover iLock consists of a front end which reads the input (generic PiSPEC specifications and configuration of specific applications), a core that produces instantiated specifications for the specific application, and a number of additional modules. The most important modules are described below.
**Prover iLock Verifier**

Prover iLock Verifier lets the signaling engineer prove that the implementation of the specific application meets the generic safety requirements by the push of a button. This is achieved using a technique known as *formal verification*. It uses automated mathematical reasoning to prove that there is no possible scenario for which a safety property is violated, or if this is not true, it produces a scenario that demonstrates when and how the safety property is violated. Running on a standard laptop, the formal verification typically takes only a few minutes to complete for interlocking systems of average complexity.

Formal verification may be considered a novel technique in the context of railway signaling systems, but it is actually quickly gaining recognition as a cost efficient and reliable way of providing evidence of that signaling and train control systems meet their safety requirements. It is today strongly recommended by the European standardization committee CENELEC for SIL 4-compliant systems, and is required or recommended as part of the safety assessments by a number of leading railway operators such as New York City Transit, Paris Metro, Swedish National Rail and Norwegian National Rail.

**Prover iLock Simulator**

Prover iLock Simulator is an efficient desktop simulator which executes the automatically instantiated test specification to validate that the implementation of the specific application meets the generic functional requirements.

Prover iLock Simulator provides an easy to use graphical debugger which can be used to inspect failing test cases or to simply gain understanding of how the interlocking systems executes. The debugger includes a ladder logic view of the application software which can be used to browse the generated programs and examine the values of individual variables at given points of the execution.

**Prover iLock Coder**

Prover iLock Coder is used to export the equations that have been instantiated for the specific application, along with additional configuration for the target platform, such as IO assignments. This output is then read by the compiler for the chosen target platform and is, after compilation, ready for execution in the field. The exact format of the output differs depending on the target platform.

© 2011 AREMA ®
THE PROVER iLOCK FOR AMERICAN RAILROADS SOLUTION

The Prover iLock for American Railroads solution is a complete off the shelf offering for design and V&V of railway interlocking software targeting the North American market. It is designed to address the challenges and objectives discussed in this paper as well as overcoming the shortcomings of traditional design processes. Using the solution, coding, test and verification of application software for an interlocking system of medium complexity can be done by one engineer in less than one day. Perhaps more important is the reduced time needed to train new staff; this can be crucial for instance when a senior signaling engineer retires or when an operator wants to use in-house development teams.

The solution consists of generic specifications for safety, test and design together with the Verifier, Simulator and Coder modules of Prover iLock. The specifications are defined in the English language and formalized in PiSPEC. Code generation, as well as V&V, supports commonly used target platforms such as VHLC, ElectroLogIXS and Microlok II.

Generic Specifications

The generic specifications capture requirements that are based on the practices recommended by AREMA and can be used directly as is with Prover iLock to produce, test and verify fully functional application software. However, as each railway operator also has its own set of practices and requirements, it is also possible to adapt the specifications to the individual needs of a particular railway operator.

The generic requirements are divided into three specifications, the GSS, the GTS and the GDS, specifying generic safety, test and design requirements, respectively. Each specification is given in a text document, written in English for human readers, and as a PiSPEC implementation, which removes all ambiguities. The PiSPEC specifications can be read and understood both by trained signaling engineers and by Prover iLock.

The generic design specification defines generic equations that will be instantiated in the generated application software. An example of the PiSPEC definition of one such generic equation, RouteValidation, is given in Figure 2. This equation is, among other things, responsible for making sure that the switch position requirement discussed above is met, by checking that the switches of the route are in the required positions.
Specific Application Configuration

A specific application is configured in Prover iLock by drawing the track layout in the built in graphical layout editor and configuring the interlocking objects. Typical configurations include signal aspects, track codes and hardware properties. This configuration is then used by Prover iLock to instantiate the generic requirements for this specific interlocking.

Equations:

""" | REF GDS REQ_3.6.1.3) True if the route is requested and clear, so that the signal may display a proceed aspect. """

RouteValidation :=

SOME csi:SELF.signal
  csi.request_final_repeater &
ALL rt:SELF.route {
  ALL csi:SELF.conflicting_signal csi.approach_stick &
  ALL bl:rt.os_tracks bl.vacant_repeater &
  ALL sw:rt.normal_switches sw.normal_correspondence &
  ALL sw:rt.reverse_switches sw.reverse_correspondence &
  ALL sw:rt.switches (~sw.lock & ~sw.lock_stick) &
  ALL lx_bcp lx_bcp.battery_check_repeater
};

Figure 2, the PiSPEC formalization of the generic RouteValidation equation.

Figure 3, a track layout in Prover iLock, showing a crossover location.

A lot of the specific configuration data is generated automatically by Prover iLock from the track layout and definitions in the Generic Specification, such as the set of valid routes and their properties. For instance our earlier example of a specific requirement “Route 500S-499N requires that switches 1A and 1B are normal and that switches
2A and 2B are reverse” is generated from the track layout in Figure 3 as a route object named “500S-499N” with the parameter normal_switches set to \( \{1A, 1B\} \) and reverse_switches set to \( \{2A, 2B\} \).

**Generated Application Software**

Even though there seldom is any real need for manual inspection of the generated application software it still may feel reassuring to many engineers if they can inspect and understand the code. To facilitate this, the generated code is designed so that it looks like it might as well have been written by a human, albeit an unusually careful and structured one. It also comes complete with documentation so that it is possible to trace individual equations and variables back to requirements in the PiSPEC and natural language specifications.

An example of what an instantiated equation looks like in Prover iLock before being exported to the chosen target platform is displayed in Figure 4, which shows the RouteValidation equation from Figure 2 instantiated for route 500S-499N in the track layout of Figure 3. The name of the equation in the code is \( 1W_2EHZ \), where \( 1W_2E \) is a shorter name for the route and \( HZ \) the suffix used for the RouteValidation equations.

![Figure 4, the RouteValidation (HZ) equation instantiated for route 500S-499N (1W-2E).](image)

**Verification and Validation**

The verification and validation is done using the Verifier and Simulator modules of Prover iLock, by the push of a button. The result of the V&V activities includes the instantiated test cases that have been simulated along with the simulation results and the instantiated safety properties along with the verification results. These activities remove the need for any manual testing to verify the correctness of the application software in itself, though it is of course still necessary to test that the complete system is installed correctly.

To improve the process for verifying the installation of the system it is possible to use Prover iLock to generate test plans containing detailed test instructions. These plans can then be used in the field by the test engineer. The main
benefits of generating the test plans, compared to producing them manually, are that it saves time and improves the quality by eliminating the risk for manual errors or omissions in the test plans and guaranteeing consistency.

**Process**

The Prover iLock Process on which the Prover iLock for American Railroads solution is built is designed with the characteristics of a well designed development process, as discussed above, in mind. Hence they are inherent in the process and by using the solution you get them for free:

- Requirements are captured rigorously in the generic specifications
- Documentation and requirement traceability is generated automatically
- Manual effort is minimized by the use of automatic code generation, simulation and verification
- Should requirements need to be updated it is easy to re-generate the code and re-run simulation and verification
- Errors in the specific application configuration are detected directly on the desktop by the development engineer during simulation and safety verification

**CONCLUSIONS**

The presented solution offers a complete and automated process for the development of interlocking software that meets common North American expectations and practices. The key benefits are:

- the development is quicker and more cost efficient
- consistency in the quality of the interlocking software is guaranteed
- a process and requirements that are fully documented

This is achieved by introducing:

- a formal way of dealing with requirements
- automated code generation, simulation and formal verification, based directly on the requirements

© 2011 AREMA ®
Table of Figures

Figure 1, a PiSPEC example, ‘→’ denotes logical implication and ‘&’ logical and

Figure 2, the PiSPEC formalization of the generic RouteValidation equation

Figure 3, a track layout configured in Prover iLock, showing a crossover location

Figure 4, the RouteValidation equation instantiated for route 500S-499N
Automated Development of Interlocking Software

Gunnar Smith
Prover Technology
Overview

A complete development process for interlocking software that

- Reduces development time and cost
- Gives consistent and well documented results
- Provides improved methods for checking quality and safety

This is achieved by

- Generic Application (GA), defined using formal Specifications
- Tool-supported development and V&V of Specific Applications (SA)
The Prover iLock Process

Generic Application for interlocking software
- Formal Requirement Specifications – PiSPEC and natural language
- Target platform adaptations
- Specific Application configuration adaptations
- Reused for each Specific Application

Automation and tool support
- System Configuration – Prover iLock Base
- Code Generation – Prover iLock Coder
- Functional Testing – Prover iLock Simulator
- Safety Verification – Prover iLock Verifier
Generic Applications

- Defines generic rules and requirements
  - Applies for a range of systems, e.g. all interlockings for a given railroad
- Generic Application Configuration
  - Defines how to configure each SA
- Generic Design Specification (GDS)
  - Defines how to implement each SA
- Generic Test Specification (GTS)
  - Defines how to test the functionality of SA
- Generic Safety Specification (GSS)
  - Defines how to verify the safety of each SA
GA Development

- Signaling principles
  - Gather
  - Document
  - Formalize

- Specify interfaces
  - SA Configuration Data
  - Target platform

- Validation
  - Using test configurations
  - Reviews

Diagram:
- Gather railway signaling principles, whether documented or not
- Document signaling principles into separate specifications
- Formalize informal signaling principles
- Generic Application (in PiSPEC)
Prover iLock for American Railroads

- Adaption of the Prover iLock Process for the North American market
- Complete Generic Application
  - Specified in collaboration with Canadian Pacific
  - Signaling principles based on AREMA guidelines
- Target Platforms:
  - VHLC, ElectroLogIXS, Electro Code
  - MicroLok II, Microtrax
Requirement Specifications

- Complete and well defined requirements is a key to successful development
  - What to implement
  - How to validate the implementation

- Traditionally expressed in natural language
  - Often ambiguous, i.e. may be interpreted differently by different people
  - Cannot be interpreted by computers
Formal Specifications

- Defined in a mathematical language with well-defined semantics
- Reduces risk of
  - Misinterpretations
  - Omitting requirements
- Improves quality of natural language specifications
  - Errors are resolved during formalization
- Can be interpreted by computers
  - Enables automation
PiSPEC

- A formal specification language tailored for railway signaling applications
- Based on established language constructs
  - Predicate logic
  - Object orientation
- Contains support for defining generic
  - Equations (GDS)
  - Test cases (GTS)
  - Safety requirements (GSS)
“A route must be cleared only if every switch in the route is proved in the position required by the route”

sw_pos_req :=
All rt:ROUTE ( 
  rt.clear ->
  All sw:rt.normal sw.is_normal
  All sw:rt.reverse sw.is_reverse);
Development

Generic Application
- Formal specification in PiSPEC
- Reused for each specific application

Specific Application
- Configuration (RAC)
- The GA is automatically instantiated for the SA
- Prover iLock Base

Output
- Automatically instantiated code ready for compilation
- Automatically instantiated requirements (safety & test)
- V&V Reports
- Documentation and Tables
Prover iLock

Prover iLock Coder
- Exports the instantiated interlocking software
- Ready for compilation, complete with IO configuration

Prover iLock Simulator
- Executes the instantiated functional test cases
- Simulates the instantiated interlocking software
- Test cases can be exported for use in FAT

Prover iLock Verifier
- Formally verifies the instantiated safety requirements
- 100% coverage using automated mathematical reasoning
Summary

- Good requirement specifications is a key to successful development
- Formal specifications
  - Improves quality of specifications
  - Enables automation
- Automation
  - Cuts cost
  - Improves consistency
  - Reduces dependency on undocumented know-how
Benefits

Efficient Development
- Deliver system in 1 day
  - Code generation
  - Formal verification
  - Simulation
  - Documentation
  - FAT/SAT test sheets
- Efficient change support
  - System configuration
  - Generic Application

Efficient mass production
- Deliver large no. of systems
  - Multiple target platforms

![Graph showing cost comparison between Application-centric process and Prover iLock Process. The graph plots Number of interlocking systems developed against Cost.]