How Train Transportation Design Challenges can be addressed with Simulation-based Virtual Prototyping for Distributed Systems

F Corbier, L Kislin, E Fourgeau

How Train Transportation Design Challenges can be addressed with Simulation-based Virtual Prototyping for Distributed Systems

F. Corbier[1], L. Kislin[1], E. Fourgeau[2]

1: TNI-Software, 3 allée d'Auteuil, F-54500 Vandoeuvre les Nancy

2: TNI-Software, 174-178 quai de Jemappes, F-75010 Paris

Abstract: The needs to accelerate development of next generation train control technology, while meeting the quality standards and safety requirements demanded for the development of today's distributed, software dominated, train transportation applications, can only be achieved through a novel methodology that addresses analysis and integration validation of sub-systems, via simulation-based *virtual prototyping*.

The methodology must be able to address, much earlier in the development cycle, design goals (among others) such as:

- Specifications Validation and Quality of Service Analysis,
- Systems safety evaluation and commissioning of automated train control systems,
- Early Integration Validation,
- Fast assessment of derivative designs,
- Compliance with transportation standards (EN50128 - SIL2).

Such methods actually used in some current projects can be seen as exemplary; they are paving the way for other transportation industry domains to adopt *in confidence* new system design paradigm shifts.

Keywords: Transportation, EN50128, Model based Design, Virtual Prototyping, Progressive Integration, Validation, Software Quality, Certification.

1. Introduction

A train is a complex system where many professions and different engineering disciplines are involved: from boiler-making to passenger safety engineers, including Traction-Brake or Quality Services. In this context, electronics take a large place (*Figure 1*) on one hand in intelligent equipments (doors, traction, brake, light, fire detection…) on the other hand, in the train vertebral column (train control and maintenance system).

With the growth of electronic equipments and networks in the train, and the increase of customer quality services and certification constraints, transportation suppliers have been looking for both methods and development tools that are able to:

- Validate the functional specifications and requirements as soon as possible in the project,
- Analyse the quality of customer services,
- Validate the safety (equipments and passengers),
- Qualify electronic control units and electro mechanical equipments of suppliers with progressive integration,
- Easily manage all changes (requirement modification…)

while staying compliant with transportation rules & standards, like EN50128.

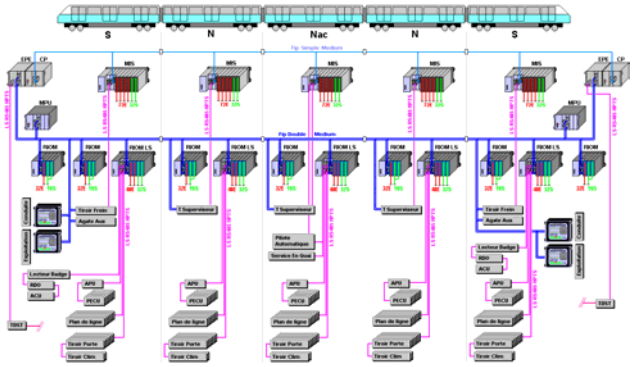

Figure 1: two hundred networked ECUs

There are today stiff requirements to become more efficient in designing and delivering on-board electronic devices, notably:

- To keep control on costs despite increasing challenges set by electronics complexity,
- To compensate by a better use of technology the organisational issues imposing less people to do more and faster.

In the following pages, we are illustrating how novel processes and tools can be successfully used on

today's most advanced train projects (intercity, tramway, metro, high speed train).

2. Historical development cycle

Traditionally in the train transportation industry - and in other industries alike – the design process for embedded electronic boards was built upon a certain number of elementary practices:

- Paper specifications exclusively, prior to the C code elaboration phase which was either generated manually or with programming tools,
- Many complicated and mandatory reviews for specification validation,
- A "Gate Review" which marked the final decision step before execution,
- Back and forth painful iterations between Business Units, Central Office and Sub-Contractors,
- Slow project starts due to a lengthy exchange process in the early stage of the project.

Consequently a certain number of productivity issues had to be dealt with:

- A lot of projects missed their delivery deadlines,
- Too much effort was spent on "Paperwork" rather than on development and integration,
- Too much time was spent in "meeting" specifications, even in sharing them and making them clear and understandable by all development groups involved,
- No "re-use" possible of high level design intellectual investment,
- Software Quality was becoming an important concern,
- Projects costs were difficult to keep within budget,
- Important managerial issues had to be addressed.

3. Introducing a drastic change

In early 2003, breakthrough decisions were made. Notably, one of the world leading transportation companies moved from a paper based specification approach to an executable specification process, in order to speed up development cycles and meet deadlines.

The process was first launched with Electronic Control Units for Train Control & Monitoring System projects. The ControlBuild technology from TNI-Software was chosen as the common tool platform to achieve new productivity related objectives:

- Use of open ECU hardware platforms,
- Design of a virtual reference prior to any implementation,
- Automated allocation of functions on targeted resources with model transformation and code generation,
- Scalability to the full train (traction for example),
- Risk mitigation with progressive implementation on HIL platforms (Factory Acceptance Tests),
- Automatic documents generation.

Since then, most of the ControlBuild productivity development technologies have been deployed; their use has been spread along the entire life cycle of the train and equipments, among architects, developers, and integrators.

4. Concept: Component Based Design

ControlBuild uniqueness is to cover design, validation, hardware integration and verification tests steps in a single integrated process, from components assembly to electronic devices design.

Using components: why?

Even though all trains may look different from one another, they have many elements in common, such as functional *components*, (i.e. the "*all doors are closed*" function …) and many others.

Hence there are sound benefits for a transportation company to manage disciplinary know-how in *components libraries*.

What is a *component*?

- A *component* prime attribute is to be *reusable* (notably in as many projects as possible).
- Therefore a *component* is not hardware platform dependent; ideally a *component* is not located in the train, for optimum reuse flexibility reasons.
- A *component* is being defined either as an atomic function (i.e. "select the active cab"), an equipment (i.e. one door sub-system) or the train itself.
- A *component* is described as a multi facetted element, according to several domains viewpoints (*Figure 2*): end-user, customer, designer, quality engineer or supplier.

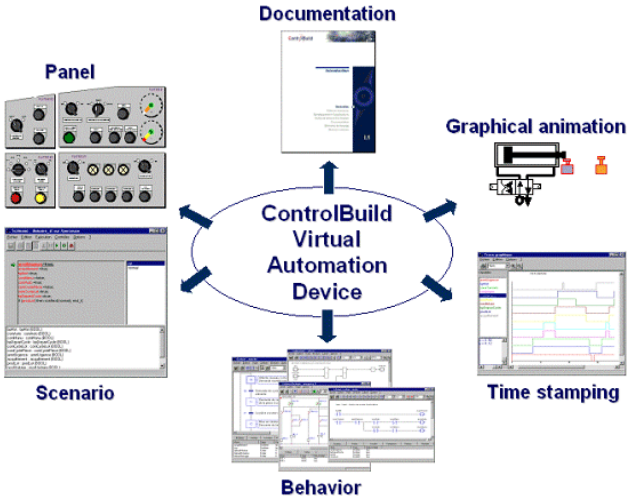

Figure 2: A Component with multiple Views

5. Compositions & Design Methodology

5.1. Functional & Requirements Specification

The functional composition approach, at train level, is structured hierarchically, in hierarchical trees of functions and sub-functions**.**

Defining the *Main Train Functions*: A train project consists of various sub-systems (doors, air conditioning unit, light, fire detection, traction, brake, driver display …). Theses sub-systems are classified in groups, called *Main Train Functions*. Each *Main Train Function* will be analysed and designed by an expert team.

Creating the functional hierarchy of a sub-system: The design methodology is based on a top down approach. The first decomposition level identifies all the specific parts of the sub-system (*Figure 3*). At each level, the expert analyst defines elementary sub functions, and for each one, its interfaces with other functions, the requirements covered, and a rich set of textual descriptions (generalities, usage, functional specification, fault detection, operating modes, availability, safety…).

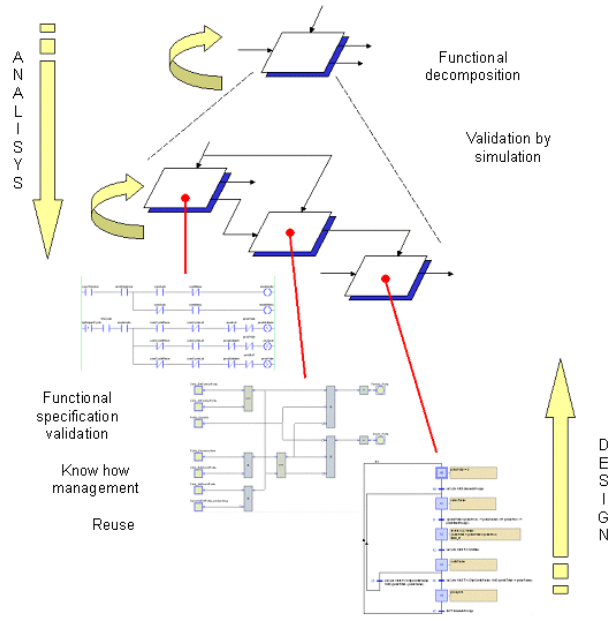

Figure 3: Mixed Top-Down & Bottom-Up methodology

Reuse of know-how: Inversely at each level of the system composition, system engineers have to determine if the functionality exists or not in the company libraries.

They will establish a link on the reusable components so that they don't have to be described again. Users can classify, archive, and reuse their company's know-how in design rule standards and component libraries, so as to improve the productivity and quality of their designs.

Reusability at this level is of significant benefit; it is a mean to win time and money and improve quality.

There are indeed many software tools on the market that allow for "code" reuse: but code reuse only! ControlBuild uniqueness is to enable reuse on all component facets: requirements, behaviour/know-how models, HMIs, graphical views, textual descriptions, qualification scripts, tests, execution reference...

5.2. Design Specifications

Each function can be described using given formalisms - standard languages are favored - that enable component descriptions to be *executed*. Hence, whatever its granularity, each function comes with an *executable model*; the model becomes an intrinsic part of the specification, enabling at any stage of the system composition, non ambiguous and interactive discussions between the various actors of the development chain, customers included.

Two *execution views* are possible, based upon two different use cases:

The first one, called *dynamic function execution* is

using the description language best suited to the function, from either:

- IEC61131-3: this applies to most sequential or combinatory processes,
- Low Voltage Schematics: when security levels are implemented on hardware,
- Physical models: to describe sensors and actuators, and also physical laws.

The second one is used to show the resulting function to a customer or end-user who is not necessarily a technical expert but owns the high level requirements and specifications of the function. In this case the execution is providing a highly abstract view of the system, using graphical representations and HMIs (*Figure 4*).

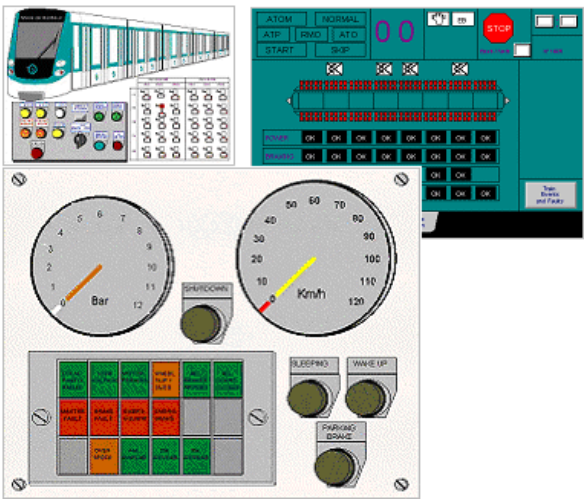

Figure 4: Graphical end-user view

This second view is called the animated *graphical prototype*. With this view, ControlBuild offers powerful means to verify as early as possible the system's conformance to the requirements.

Other *viewpoints* or component facets can be described, such as specifications and design documents, qualification scripts, formal tests, maintenance profiles, integration guidelines,…

5.3. Hardware-Software Integration

At this step of the development cycle, all functionalities and requirements have been taken into account and validated by engineers from both customers & suppliers sides. Now comes the time for the Transportation Company to change scope and start mapping the functional model on the target networked hardware architecture.

Changing scope from software to hardware centric is not an easy task; in fact ControlBuild is of great help in this process, as described step by step in the methodology detailed hereafter.

Hardware description: First, hardware architecture has to be described in terms of physical resources (virtual targets) and tasks.

Mapping software on hardware architectures: Each function of the virtual train model should be mapped on one task of the hardware architecture (*Figure 5*). Eventually the functional models are mapped on the individual hardware resources available and transformed into tasks on the target.

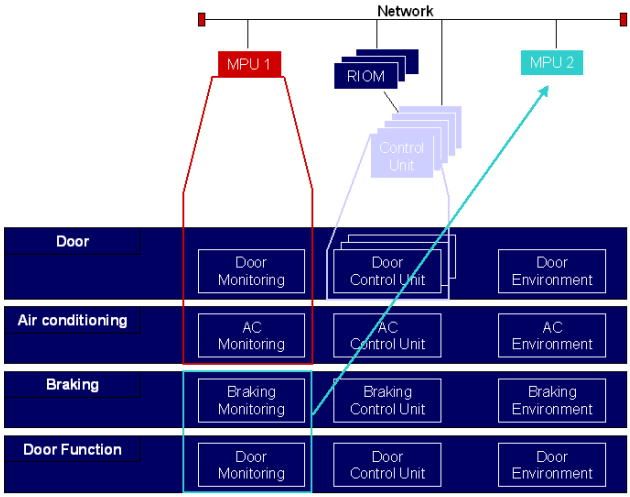

Figure 5: Mapping functions on hardware

Hardware Interface configuration: The *assistant* technology provided in ControlBuild can identify different interface levels:

- Exchanges between two tasks inside a given ECU. They are automatically managed and addressed by ControlBuild *code generator,*
- Exchanges between two ECUs: the designer has to manually define: support signal (train or car network) or group.
- Input and Output from one ECU to physical actuators and sensors: the designer has to assign each signal on a Fieldbus, an I/O card or a remote I/O equipment.

Code generation: once each physical target is declared and configured (re: operating system, network address, cycle time of each task…), the software code of each one (*Figure 6*) will be generated and linked with the dedicated environment (make, libraries …).

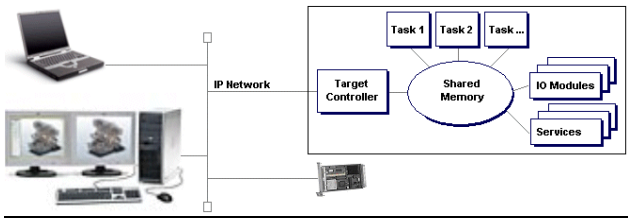

Figure 6: Code Generation multi-target/multi-CPU

Requirements and specifications: the output of this step is a Requirement & Specification Document for the equipment manufacturer. The elaboration of this document is entirely automated.

5.4. Progressive Integration

The integration of all train functions is *progressive*. By this we mean first integration and validation of individual ECUs and equipments, then integration and validation of ECUs contributing to a train function, finally integration and validation of all train functions.

Each integration step is implementing real equipments. It is obviously much simpler to integrate single equipment from a given supplier, than to proceed to the complete integration of all electronic equipments of the train, in one go - not only for costs reasons, but primarily since it is quite impossible to get all of the 200 ECUs of the train simultaneously!

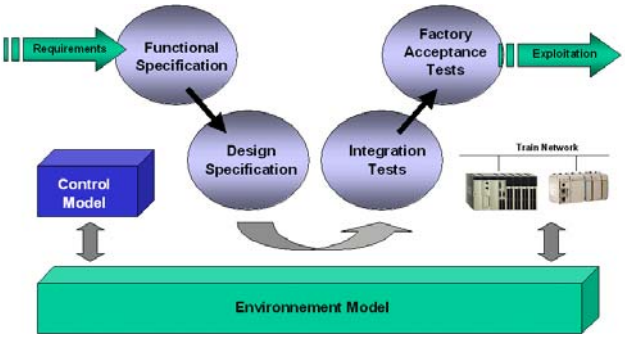

Figure 7: Reuse environment model

The risk mitigation approach enabled by ControlBuild at this stage is unique : starting from the virtual model of the complete train architecture, the tool provides mechanisms to declare physical equipments as they become available for integration. Equipments available for integration are then connected to the integration test bench and disabled in the virtual model.

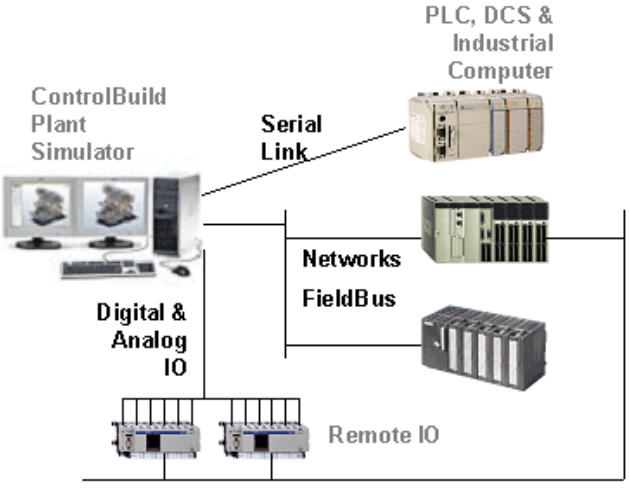

Figure 8: Multiple levels of connexion

Hence the virtual model progressively integrates the real physical elements, at the same time as it provides a virtual environment model for these same physical elements (*Figure 9*).

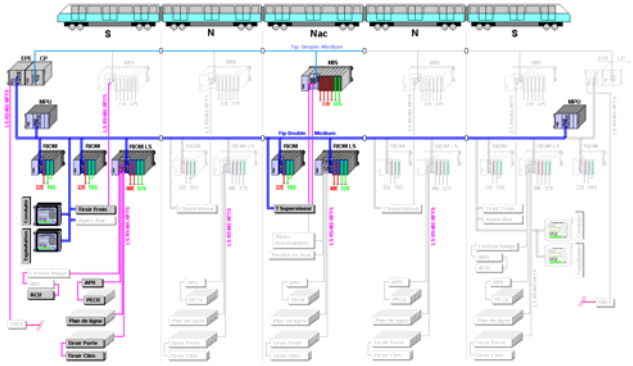

Figure 9: Progressive Integration

This approach brings high value benefits: it allows a productive support of system evolutions over the complete train lifecycle, i.e. 30 to 40 years. This is notably the case with all non regression tests mandated by the progressive obsolescence over time of some ECUs.

6. Documentation management

The various steps of a given train transportation development project are producing an incredible amount of documents (re: EN50128).

Previously this was a huge burden on designers, who had to keep focus on writing these documents. With the novel ControlBuild processes, designers are spending time on design and much less on paperwork : "*make it work*", "*make it work according to customers requirements*" is of essence.

During the composition, modeling and validation phases, each component at each level receives textual information (functions, operation modes, safety, …) as well as requirement coverage scores.

This is enabling a fully automated document generation step, capable of covering the entire project needs, notably:

- Software Requirement Specification
- Software Design Specification
- Software Architecture Description
- Software Modules Design Specification
- Software Modules Tests Procedure…

Centralizing/binding all project information in a global model, ensures that all project related documents always stay consistent and cohesively updated,

whichever modifications or upgrades are made to the project.

7. Conclusion

The quantitative value proposition of such an approach in terms of productivity gains and organizational effectiveness is promising – unfortunately it is not public information.

Intangible benefits are not to be left behind either:

- Executable specifications allow:
  - Easy validation and design reviews with both internal and external non experts groups
  - Robust implementation with virtual simulation.
- Integration/simulation/test platforms can be created at a fraction of time compared to before.
- Full Functional HIL validation is possible, with on time delivery.

8. Glossary

ECU Electronic Control Unit
ERTS Embedded Real Time Software
HIL Hardware In the Loop
SRS Software Requirement Specification
SDS Software Design Specification
SAD Software Architecture Description
SMDS Software Modules Design Specification
SMTP Software Modules Tests Procedure
TCMS Train Control and Monitoring System