MBD & CODE GENERATION: A COST-EFFECTIVE WAY TO SPEED UP HMI CERTIFICATION

Luc Marcil, HMI Product Manager, Presagis, Montréal, Québec (Canada)

Abstract

This paper compares the traditional Text-Based approach with Model-Based Design (MBD) approach and demonstrates the efficiency of adopting Model-Based Design when coupled with application simulation and automatic code generation.

Organizations can see gains not only in the reduction of the development cycle but also in the overall improvement of the DO-178B or DO-178C certification process; including reduction of schedule and costs, and improvements in the quality and reliability.

In the old school of thought, the methodology relies on textual specifications and physical prototypes. That is why the Text-Based Design approach is tightly associated with the waterfall methodology where all the textual requirements are manually coded, inspected, and tested on a real embedded system. In this method, changes in any part of the waterfall chain are very costly and time consuming, leaving almost no room to iterate on the design.

By contrast, in the Model-Based Design approach, the specifications are self contained in the HMI Model. The HMI requirements are defined in an unambiguous way and often captured in a formal definition language. The model owns many graphical representations, called model views, such as Structure Diagram, State Charts, Interaction and Sequence diagrams, etc. Model-Based Design incorporates the behavior and performance requirements to properly describe the overall avionics application. This model possesses graphical representations and can be used throughout the entire development cycle, from the requirement inception phase up to the final embedded application deployment.

Model-Based Design offers a collaborative approach to avionics development and allows engineers to inexpensively experiment with various concepts by involving hardware as late as possible in the development process. Correcting problems in the early simulation phase is undeniably the strongest argument in favor of the Model-Based Design approach for developing certifiable or non-certifiable avionics applications.

Company Culture Paradigm Shift

Aircraft builders and avionics providers continue to be reluctant to adopt a Model-Based Design approach. The majority of Aerospace companies[1] continue to write textual requirements; manually code embedded avionics applications and run the majority of their tests on real hardware.

This hesitation is understandable when considering that adopting a Model-Based Design approach can create a huge paradigm shift in a company's development culture. Additionally, the usage of a Model-Based Design Tool is often perceived as costly and risky as a third party is introduced in the mix.

Work Process Right Angle Turn

In addition to challenges associated with adopting new technologies and work processes, avionics engineering departments also need to adapt and refocus their activities to incorporate the Model-Based Design workflow. Adopting this approach needs to happen across departments and can lead to redefinition of global work processes.

With all the players involved in the change process, there are a plethora of established habits that can be difficult to surmount in order to deploy a new streamlined process built around Model-based design. The natural behaviors against the culture change revolve around the perceived risks of adopting a new process. Constructing a new work process requires a vision supported by the entire corporation, beginning with senior management, and including day-to-day users of the new process.

---

[1] VDC Research, page 35 (cf. References [1]), page 35: 54.3% of in-house developed lines of software code that will go into the final embedded design for program producing from 10,000 to 99,999 lines of code (LOC). The mean number of in-house developed LOC is 113,352.

New Tools

The adoption of the Model-Based Design approach requires new software tools to support and facilitate the new work process. Without proper tools, the Model-Based Design approach can face many road blocks towards a broad acceptance. Therefore, a parallel effort on the work methods and software tools has to be deployed to ensure the success of this paradigm shift. To capitalize on the positive outcomes from a changed approach, both people and technology need to change in tandem.

Role Redefinition

As the Model becomes the center of all activities, multiple domain experts are needed, and jobs have to be adjusted to fall in line with the new approach. Although significant changes are sometimes needed to adopt a Model-Based Design approach, there are many demonstrable benefits that can make this effort worthwhile; which is what this paper is exploring.

Designers and developers are the best advocates to promote a design that can ultimately meet better performance, even if it requires a deeper analytical process up front. With empirical results indicating improved workflow and performance it becomes difficult to justify departments working in silos, struggling to merge their efforts near the end of a program. Collaboration from the outset of a program is important to maintain interoperability when using the Model-Based Design approach.

Traditional Waterfall Process

Textual Requirements and Hand Coding

For many decades, avionics systems were developed using waterfall and Text-Based approach. The requirements are defined with natural languages such as English, and then manually translated into a high level design. Usually, the design is manually coded in a procedural language such as ADA or C[2] which often undergoes a manual code inspection.

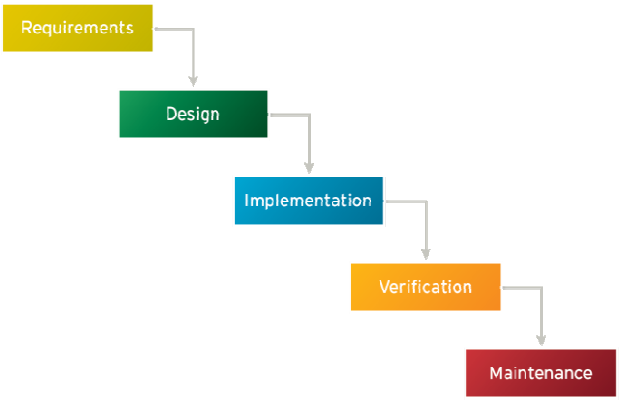

Figure 1 – Waterfall Approach

Hand Coded Prototype

The validation and verification are directly made on the handed coded prototype. Some automated tests (DO-178B/C or not) are also manually written and executed among other tests on the final hardware platform.

Without a tool that can simulate the textual design, it is very tedious to execute functional tests on the hand coded prototype to validate the requirements. By contrast, the modeling tool enables the developers to execute the specifications in a simulated environment without the need of hand coding the prototype and functional tests.

In the Text Based approach, any changes in any part of the waterfall chain are very costly and time consuming, leaving almost no room to maneuver and will certainly lead to a scheduling conflict.

---

[2] VDC Research, page 35 (cf. References [1]), page 38: Language C is used in 85.1% of the embedded programs and ADA in 10.6%. Assembly is still used in 36.2%.

Model-Based Design Components

Model-Based Design uncovers potential pitfalls and design mistakes that may be missed or discovered too late when using traditional methodologies. The HMI Model and its simulation are very close to real world operation and problems can be discovered while developing and simulating the model. Every step of the development is continuously tested against the requirements which results in identification of defects early on. It is no secret that it is easier and less costly to correct defects as they arise in the early stages of the development process, rather than rush all the required changes near implementation.

From a Waterfall to a Fountain

An analogy can be made from the water in a fall and fountain. The text-based approach can be seen as a waterfall where the water flows downward, from the step above, until no more water remains. The water cannot go back up unless it is conveyed by a mechanical means. That is why it is so difficult to go upstream and make modifications to work done in earlier phases.

The Model-Based Design approach can be compared to a fountain, where the water flows out from a central source, but then is recycled back through a central jet to be distributed in all directions. The HMI Model becomes the central point, accessible by the entire organization.

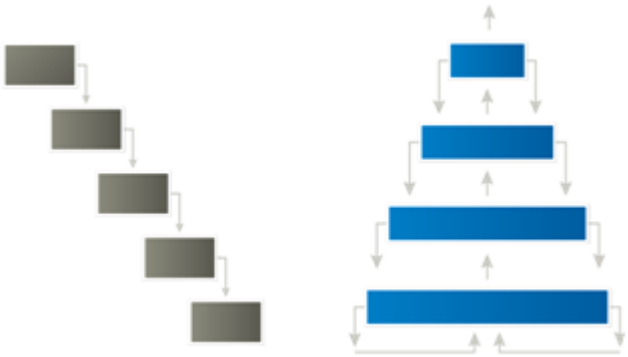

Figure 2 – Waterfall and Fountain

HMI Model

The DO-178C Standard defines a model as an abstract representation of a set of software aspects of a system that may be used to support the software development process or the software verification process. A model also carries relationships between selected properties, consciously constructed for the purpose of a requirement.

In contrast to the text-based approach, in the Model-Based Design approach, the specifications are self contained in the HMI Model. The model is defined in an unambiguous definition language, such as the Unified Modeling Language (UML), or any other modeling language.

Model-Based Design incorporates the behavior and performance requirements to properly describe the overall avionics application. This model possesses graphical representations and can be used throughout the entire development cycle: from the requirement inception phase up to the final embedded application deployment.

The HMI Model becomes the center of gravity of all engineering activities. The HMI Model becomes the application specification and the unique medium through which all teams communicate. The system-level HMI Model is located in the center of these five major engineering activities:

1. Graphically specify the HMI Model – i.e. Model building
2. Research for new concepts – prototyping and refining the HMI Model
3. Simulate and stimulate the HMI Model
4. Continuously test and verify the HMI Model
5. Deploy the HMI Model with automatic code generation

Graphical modeling tools can be used in all aspects of creation of the HMI Model. These tools provide a very generic and unified graphical modeling environment often based on a standardized XML definition format such as ARINC 661. These modeling tools help reduce the complexity of model designs by breaking them into hierarchies of individual design blocks. For example, in the ARINC 661 standard, the graphical representation is broken into widgets, layers, and windows.

Models in DO-178C

The new DO-178C Standard addresses software modeling and the ability to use modeling to supplant some of the verification techniques required in DO-178B.

The DO-178C Model-Based Development supplement goes a step further in the usage of models and distinguishes two types of models: Specification Models and Design Models.

Specification Model

According to the DO-178C supplement, a Specification Model is an abstract representation of externally observable properties of a software component, such as interface, functional, performance, or safety characteristics. The Specification Model should express these characteristics unambiguously to support an understanding of the software functionality. It should only contain detail that contributes to this understanding and does not prescribe a specific software implementation or architecture except for exceptional cases of justified design constraints. Specification models do not define software design details such as internal data structures, internal data flow or internal control flow. Therefore a Specification Model may express High-Level Requirements (HLR), but not Low-Level Requirements (LLR) or software architecture.

Design Model

Again according the DO-178C supplement a Design Model prescribes software component internal data structures, data flow and/or control flow. A Design Model includes Low-Level Requirements and/or architecture. In particular, when a model expresses software design data, regardless of other content, it should be classified as a Design Model. This includes models used to produce code.

DO-178C Model Benefits

The use of models may bring the following benefits and capabilities to the DO-178C process:

1. Provide unambiguous documentation of requirements and architecture
2. Support the use of automated code generation
3. Support the use of automated test generation
4. Support the use of analysis tools for verification of requirements and architecture
5. Provide the capability to partially verify requirements and architecture using simulation

HMI Model Usage

HMI Views

The HMI Model owns many graphical representations called model views such as Structure Diagram, State Charts (Figure 3), Interaction and Sequence diagrams, etc. The view renders the model into a form suitable for user interaction, typically a user interface element. Multiple views can exist for a single HMI Model for different purposes. The different views allow users to focus on the visual appearance and functional aspects of avionics displays. For example, Figure 3 depicts logical states between display phases.

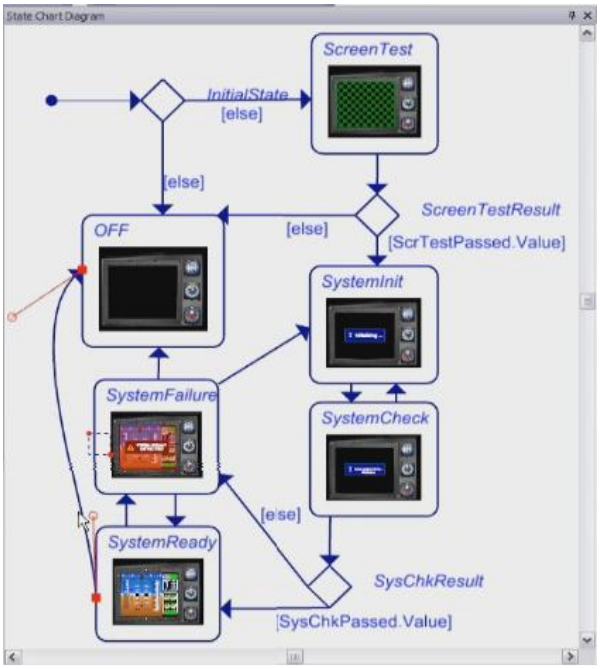

Figure 3 – UML-Based State Chart Display Logic

Virtual Prototype

In addition to the different views of the HMI Model, developers need to see the final avionics display appearance. Figure 4 illustrates a Primary Flight Display (PFD) example that can be simulated or stimulated from a graphical tool as the HMI Model contains all the parameters of the graphical objects to render the entire display.

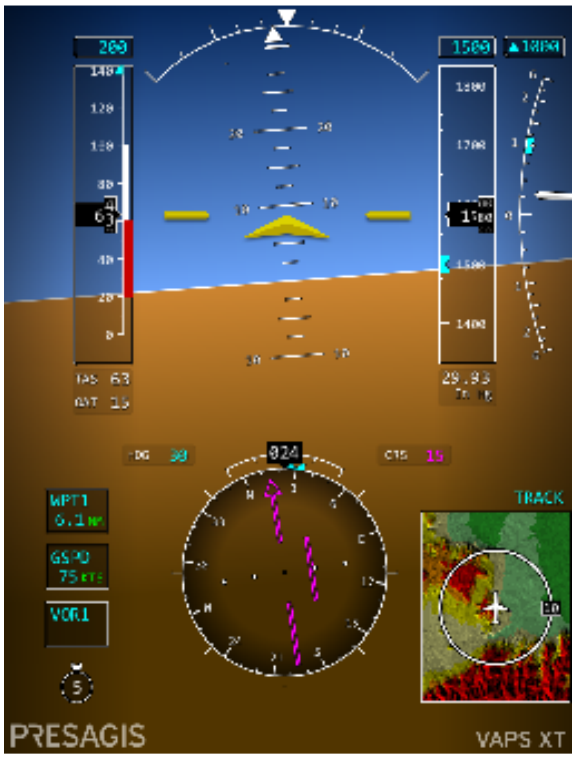

Figure 4 – Primary Flight Display

The virtual prototype is the lynch pin of the Model-Based Design approach. As the graphical model can be developed from the start in a graphical modeling tool, the tool does not only show the static aspect of the model but can also be used to dynamically simulate or stimulate the HMI Model. As the requirements are modeled, the user can visualize the model components in action; this is called a virtual prototype.

In simulation mode, the developer can manually or automatically (e.g. from a signal generator) drive some graphical parameters contained in the model. The display reacts according the behavior defined in the HMI Model (i.e. it reacts according its associated Finite State Machine). Moreover, the developer can even connect the input parameters with an external data source to stimulate the model.

Therefore, during the creation process, developers can regularly run the display in a test environment, i.e. the virtual prototype, to validate that the functionality they implemented works as expected. Developers can inspect the data that is either simulated by the tool itself or is being sent by external data sources. The virtual prototype instantaneously produces the same results as compiling the application in an executable form targeted for the physical prototype or final product.

DO-178B/C Requirement Management

Unambiguous Requirement

As opposed to textual requirements, the HMI Model is precise and leaves no room for interpretation. This approach is highly software driven and avionics developers can more easily explore new concepts without the overhead of extensive hardware investment.

Requirement Traceability

In the DO-178B and DO-178C processes, requirements traceability is aimed at ensuring completeness and correctness of the development and verification activities. It should be possible to trace back to the origin of each requirement. Therefore, every change made to a requirement should be documented in order to achieve this traceability. At any step of the implementation, requirements should be traceable in a bidirectional manner: forward and backward, i.e. from the initial requirement down to its implemented line of code and vice versa.

Software tools are especially useful to handle the numerous DO-178B/C requirements in avionics systems. A typical avionics displays such as a Primary Flight Display (PFD) in Figure 4 can contain hundreds of High Level Requirements (HLR) and thousands of Low Level Requirements (LLR).

When using an integrated set of software tools, a requirement can leave the following trace:

1. The requirement is create in a specialized requirement tool and has a unique identifier
2. The requirement is attached to a HMI Model component in a graphical modeling tool
3. The requirement has one or more visual representations in the graphical modeling tool

4. The requirement is automatically documented by a document generator
5. The requirement identifier is automatically generated into a comment in the code during the code generation process. The generated code carries the original requirement identifier.
6. The requirement is tested by a specific DO-178B/C test case created especially for the given requirement identifier
7. The result of the specific DO-178B/C test case is captured and shows compliance to the original requirement

Of course, all the above activities can be manually performed in a text-based approach. However, keeping all the DO-178B/C documents in synchronization with the code and test cases requires a large amount of work that can be counted in many man-months. The Model-Based Design approach offers some substantial saving because the design is centralized in the model and the code traceability is automatic.

Change Request

With the text-based approach, a lot of documents, pieces of code, and tests have to be manually rewritten and re-validated. Using the Model-Based Design approach greatly reduces the burden of tracing the impact of a Change Request as requirements are centralized in the modeling tool. It becomes easier to delineate the boundaries of the Change Request's impact. First, some graphical modeling tools are capable of comparing the original and updated models allowing the delta to be insulated. Second, the document generator can capture this delta and re-generate only a minimal set of documentation. Third, the code generator can produce only the software components, e.g. executables and runtime shared libraries which have changed. The other components remain untouched. It becomes then easy for the tester to only re-execute a small sub-set of test-cases on the delta.

Furthermore, the DO-178B/C standard provides the necessary process to keep the unchanged components and reuse them in an updated version of the current avionics program, or for an entirely new avionics program. The specific process is called: Reusable Software Components (RSC).

Cost Reductions and Benefits

Foremost among aircraft builders and avionics providers' concerns is cost. How much will this avionics program cost? How long will it take to be DO-178B/C certified? It is widely believed that DO-178B/C certification is inherently expensive. Unfortunately, this belief is completely true.

Avionics development programs often run in the millions[3] of dollars. These are large and complex programs involving many engineering teams. The cost of the DO-178B/C process represents a fair portion of the overall program costs. That is why it is important to use an approach that can maximize the use of resources. The Model-Based Design approach coupled with the proper software tools provides a way to allocate the human and material resources more effectively.

Time Savings

The traditional Text-Based approach involving hand-coding and manual certification can result in delays and cost overruns in both the development and certification processes. Using a Model-Based Design Tool together with repeatable, proven processes reduces risk and facilitates better time to market. As every development step revolves around the HMI Model, the information does not need to be reshaped and manually translated at every phase. Moreover, with the virtual prototype, code, and document generator many manual steps are reduced to a minimum.

The result is a reliable process that saves a significant amount of time in both the design and verification phases of embedded development, as well as reduces dependencies on specialized internal skill sets.

---

[3] VDC Research, page 35 (cf. References [1]), page 25. The mean development cost of aerospace embedded project is $4,564,888

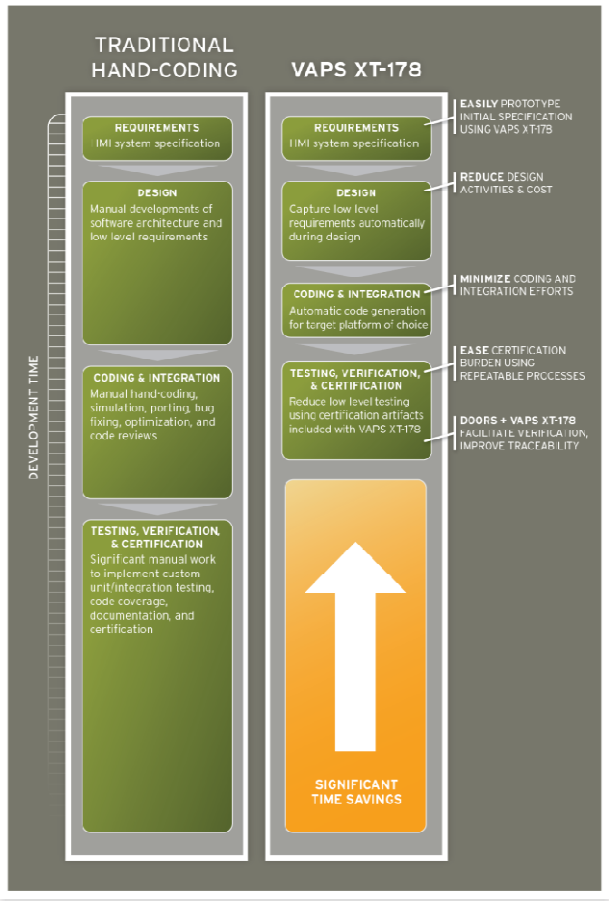

Figure 5 – Time Saving

V-Model Concept

The V-model concept depicts the development lifecycle of a complex system. It summarizes the main actives to be executed from the system inception to its final delivery and maintenance. The horizontal axis represents time, the left segment of the 'V' represents the project definition and right side represents the project testing and integration.

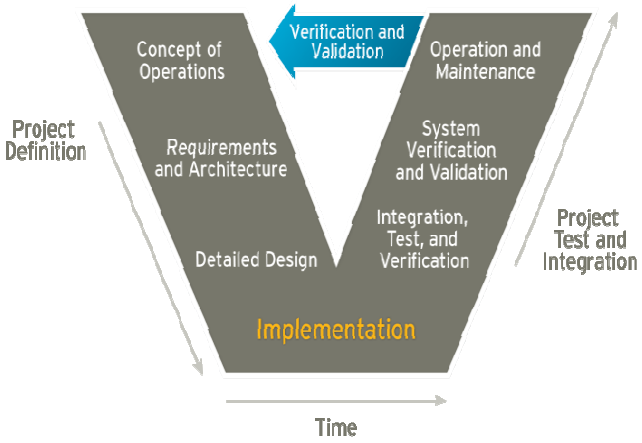

Figure 6 – V-Model Concept

DO-178B/C V-Model for Text-Based Approach

In the Text-Based Design approach (Figure 7), the developers start the HMI development process by writing, in a natural language such as English, textual System Requirements Allocated to Software (SRATS) and/or High Level Requirements (HLR) in order to fulfill the objectives of DO-178B/C (table A-3). From these requirements, the developers manually derive the Low Level Requirements (LLR) and manually capture them in documents. Then, the developers hand code the application. The DO-178B/C process requires manual code review and structural code coverage to traverse 100% of the statements/decisions/conditions in the code as applicable. For every HLR and LLR, a test case has to be written and executed to prove the compliance to the initial requirements.

A simple rule of thumb suggests a minimum of 1 LLR per 10 LOC's. There will typically be more than a single test case per requirement. The amount of work is therefore considerable for a program containing over 100,000 LOC's. It can equate to writing and executing thousands of test cases in addition to all the manual document edition and hand coding.

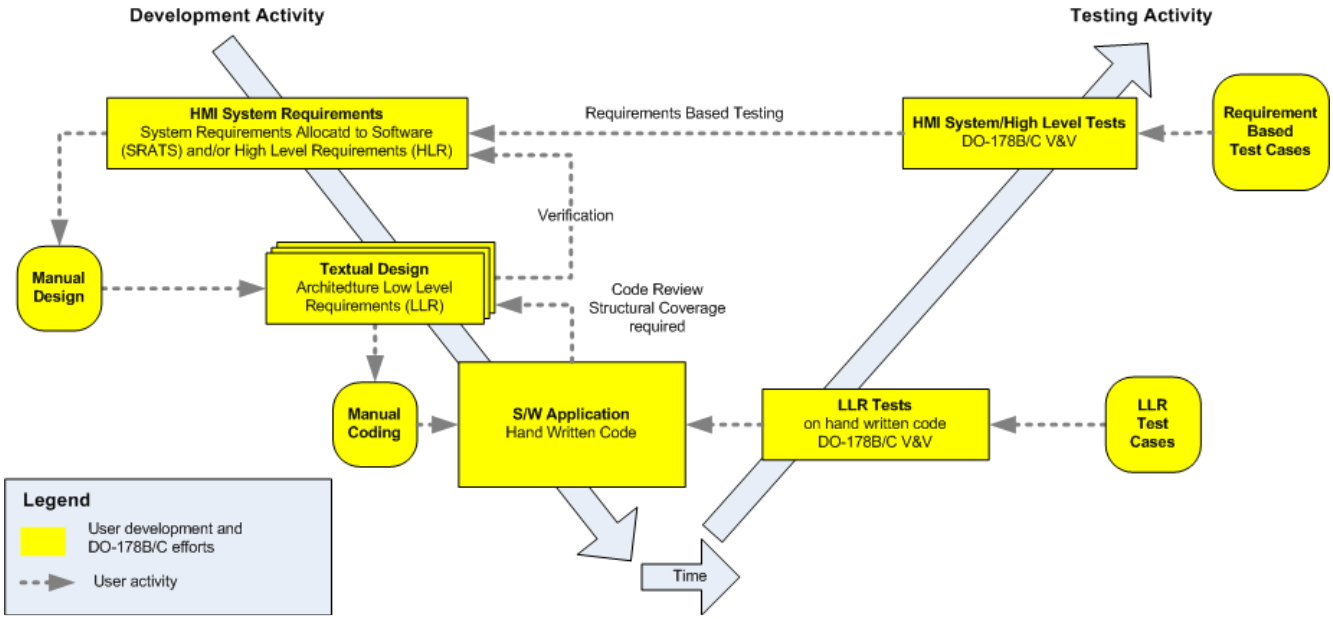

Figure 7 – Text-Based with respect to V-Model

DO-178B/C V-Model for Model-Based Approach

Figure 8 illustrates the V-Model activities needed by the Model-Based Design approach. The developers and testers can claim DO-178B/C credits for activities colored in green. The next sections explain how the Model-Based Design approach coupled with the proper DO-178B/C Qualifiable Development tools can eliminate or greatly reduce the DO-178B/C efforts.

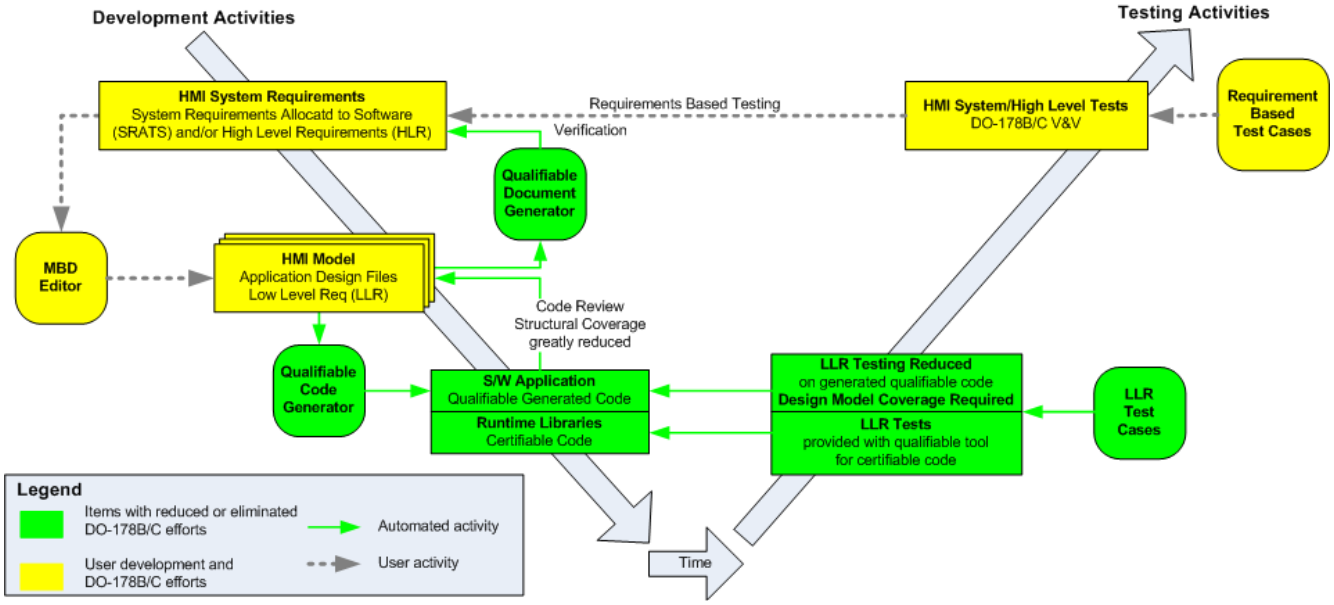

Figure 8 – Model-Based Design with respect to V-Model

Business Case for DO-178B/C Qualifiable Tools

The required up-front investment in the new work process around a graphical modeling tool can easily be paid off during the course of the first avionics program. It is wise to build a business case based on the past programs using a Text-Based approach to come up with a realistic Return on Investment (ROI). Table 1 provides an example of a ROI business case.

Standard development DO-178B/C activities	Project Effort	Savings Coefficient	Saved Effort
HLR development and reviews	10%	0%	0%
Design and LLR development and reviews	15%	50%	8%
Code Development and reviews	25%	80%	20%
HLR Test Case Development and Review	15%	0%	0%
LLR Test Case Development and Review	25%	80%	20%
Structural Coverage Analysis	10%	75%	8%
Total	**100%**	**n/a**	**55%**

Table 1 – Cost Saving Coefficient[4]

The first column of Table 1 lists the most important activities to be executed during a DO-178B or DO-178C program. The second column captures the typical effort distribution deployed throughout the entire DO-178B project. The third is the saving coefficient that can be applied when using a DO-178B Qualifiable Development Tool. And finally the last column corresponds to the percentage of the saved effort. In this case, Table 1 shows an effort saving up to 55% for the whole program when a DO-178B Qualified Development tool is utilized with the Model-Based Design approached. For instance, a potential saving of 55% applied to a $4M project does not necessarily translate into a schedule cut in half and as well as a budget also cut in half. The saved effort can be reused to further refine the final application.

In addition to the reduced costs, other benefits have to be considered: a faster time-to-market, higher quality product, Reusable Software Components (RSC), and reusable tools for other projects, etc.

Code Generation

The HMI Model can be directly translated into code by an automatic code generator. By contrast, with the hand coded application, also created from hand written requirements, the auto-generated code is completely synchronized with the model. The entire coding process to generate a fully functional executable, can take just a few minutes as compared to months of hand coding.

A DO-178B/C Qualified code generator eliminates the need to inspect and perform dynamic analysis on generated code.

Over an entire DO-178B project, Table 1 considers a 25% effort with an 80% saving coefficient for the "Code Development and reviews" activity. The combination of this effort and saving leads to a 20% saving over the whole project.

Document Generation

Providing the ability to automatically generate documentation from the Model-Based Design is another appeal of using graphical modeling tools. The document generator allows users to extract images and information about all aspects of the avionics design, including custom user comments, and output them in a user-configurable format. Once a document template is configured, specifications and technical documentation can easily be re-generated from the current model.

Detailed documentation describing the HMI specification must be written in the initial stages of development and must be kept up-to-date as the product evolves. A document generator outputs documents detailing all aspects of the HMI Model and the user has complete control over the appearance of the document by configuring the document templates. In addition, once the template is configured, if the application needs to be updated or

[4] This table is based on the data accumulated by Presagis over the past 10 years supporting more than 30 DO-178B programs from Level D to Level A.

changed in any way, the user simply needs to re-run the document generator in order to automatically create an updated document in a matter of seconds. Because the document generator could be a DO-178B/C Qualifiable Verification Tool, it can be used to review the HMI Model and DO-178B/C Low Level Requirements (LLR) against DO-178B/C High Level Requirements (HLR).

Quality and Reliability Improvement

Cost and time-to-market are important competitive benefits that the Model-Based approach can provide to aircraft builders and avionics providers. However, in addition to these financial advantages, it is also worth considering the gain on the end product quality.

The Model-Based Design approach enables the developers to spend more time on iterating on the HMI Model and its associated virtual prototype. Therefore, instead of spending time fixing the DO-178B/C process and synchronizing the documentation, the developers can spend more time polishing the system requirements. More iteration of the HMI Model can only lead to a higher quality product as the qualifiable development tool enables the user to automatically retest each iteration.

Summary

Model-Based Design offers a collaborative approach to the avionics development and allows engineers to inexpensively experiment with various concepts by involving hardware as late as possible in the development process. Correcting problems in the early simulation phase is undeniably the strongest argument in favor of the Model-Based Design approach for developing DO-178B or DO-178C certifiable avionics applications. This also applies to non-certifiable programs.

Graphical modeling tools alone cannot be the only ingredient in the recipe to successfully adopt the Model-Based Design approach. The engineering practices have to match this new approach. Without a culture change, the graphical modeling tools can certainly speed the evolution to more inclusive and Model-Based Design practices but the aircraft builders and avionics providers need to show a real will to change their traditional development culture.

References

[1] Steve Balacco, Chris Rommel, Jared Weiner, December 17, 2010, 2010 Service Year, Track 3: Embedded Systems, Market Statistics, Volume 5: Military/Aerospace, VDC Research, pp. 25, 35, and 38.

Email Addresses

The author can be reached at the following e-mail address:

Luc.Marcil@Presagis.com