A White Paper *from* ARTiSAN Software Tools

ARTiSAN Real-time Studio Support For Model Driven Architecture (MDA)

ARTiSAN Real-time Studio Support For Model Driven Architecture (MDA) is published by ARTiSAN Software Tools.

Author: Matthew Hause

ARTiSAN Software Tools worldwide offices

The Americas	**International**	**Germany, Austria, Switzerland**
ARTiSAN Software Tools Inc Two Lincoln Center, 10220 SW Greenberg Road Portland, OR 97223 USA	ARTiSAN Software Tools Ltd Stamford House Regent Street Cheltenham, Glos, GL50 1HN UK	ARTiSAN Software Tools Gmbh Eupener Strasse 135-7 50933 Koln Germany
Tel: +1 503 245 6200 Fax: +1 503 244 1443 E-mail: info.us@artisansw.com	Tel: +44 (0)1242 229300 Fax: +44 (0)1242 229301 E-mail:info.uk@artisansw.com	Tel: +49 (0)221 4852260 Fax: +49 (0)221 4852261 E-mail: info.de@artisansw.com

www.artisansw.com

Overview

Unlike other modeling tools, ARTiSAN Software Tools' support for MDA goes beyond just code, allowing you to model all the different parts of your system. Additionally, ARTiSAN allows you to model your own profiles, patterns and code generation templates. All this, coupled with the fact that Artisan provides you with the UML real-time profile, adds up to MDA support that is second to none.

CONTENTS

TABLE OF FIGURES

Introduction

As industry, methods, and methodologies continue to evolve, the UML and the OMG will continue to evolve with them to stay relevant to the sorts of problems that developers face today and in the future. Because of the vast scope of problems to which computers are applied, there are countless languages, platforms, configurations, operating systems, networks and so on, that have been created to deal with them. Just as a carpenter has many different tools in his toolbox that are appropriate to the task he is trying to achieve, so to the computing industry has a variety of languages, operating systems, and protocols that it uses to solve its problems.

These platforms, OS's, and languages will continue to multiply. For instance, over the last 15 years we have heard how languages such as Fortran and Cobol, and Operating Systems such as Unix would soon disappear. This has not happened, nor will it any time soon. As a result, we need to build flexibility into our systems to cope with future integrations that our systems will encounter. This means that we need to agree on how different systems and the different parts of our systems interface and inter-operate in order to help future-proof our systems. We will then model the different aspects and levels of abstraction of our system as well as the inter-relationships between these systems.

Getting The Big Picture

The Object Management Group's Model-Driven Architecture (MDA) defines how our systems can work together and separate out the “what we want to do” from the “how we will achieve it”.

To accomplish this, we must first model the system in its environment by creating a Context Diagram using the Real-time Studio System Architecture Diagram (See Figure 1). This allows us to define a high level view of the system without constraining ourselves to implementation platforms and configurations.

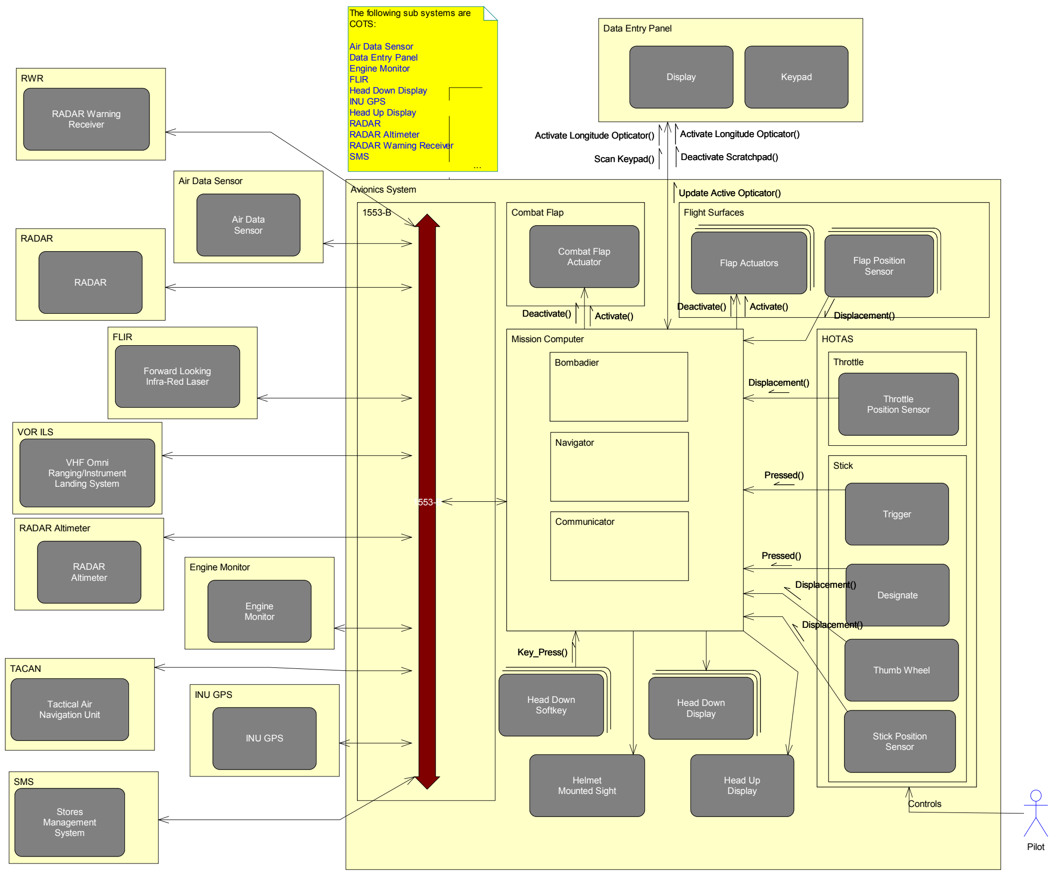


Figure 1 – Example Context Diagram

Later, when we have decided on the specific hardware configuration, we can use the System Architecture diagram for this as well (See Figure 2). Of the leading UML modeling tools, only ARTiSAN Software’s Real-time Studio provides these capabilities.

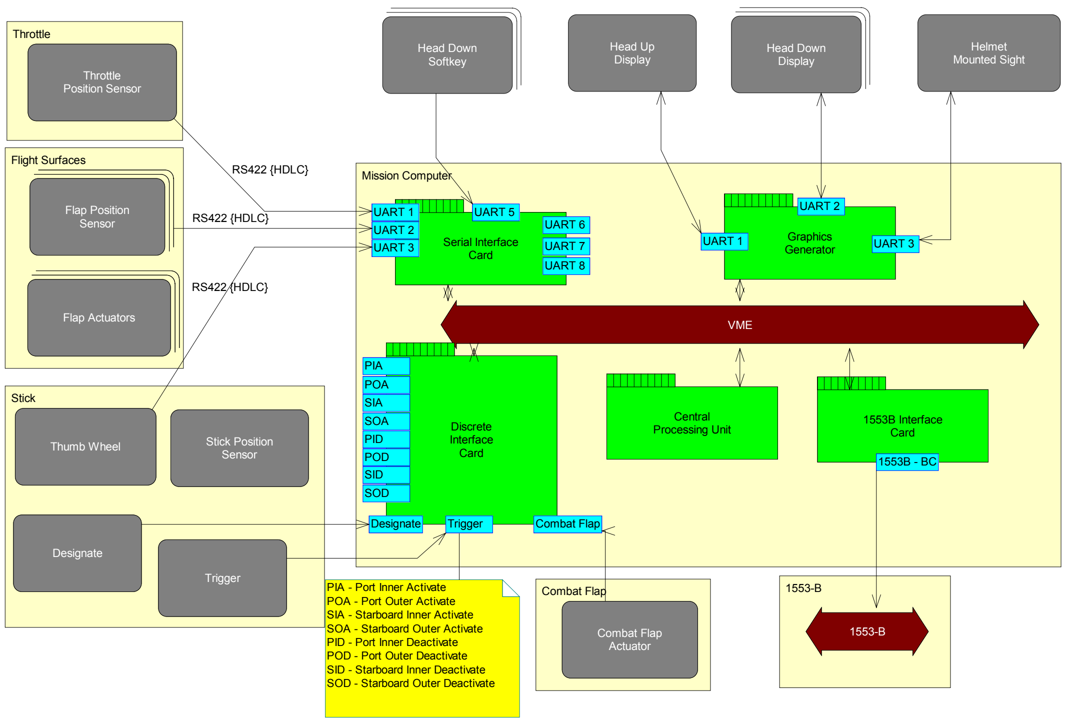
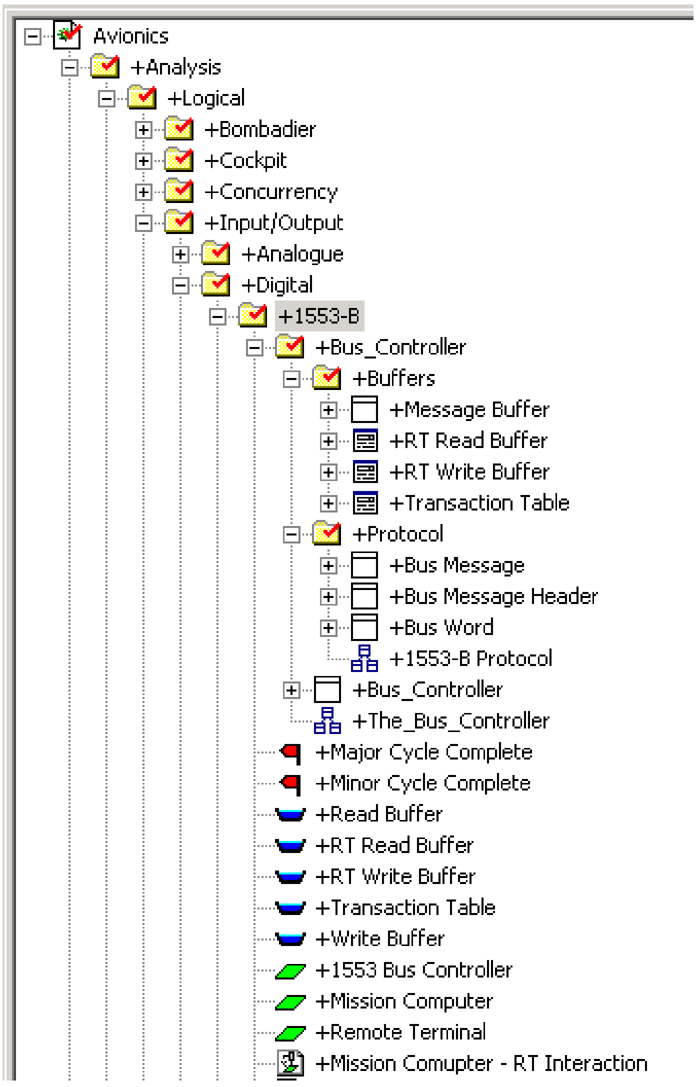


Figure 2 – Example System Hardware Diagram

A New Way To See Things

MDA is, in fact, a new way to build systems. It uses the UML, the state of the art modeling language, to express modeling concepts. MDA takes a wider perspective than most approaches to system design. Most only involve modeling how the system will function, whereas MDA takes the systems engineering viewpoint and looks at the entire system lifecycle, including new systems, that have not yet even been conceived. It looks at how the system will interact with others, and allows you to model the system separate from the implementation constraints. This will improve interoperability and facilitate porting to new platforms, protocols, and languages in the future when this becomes necessary. It helps to lower initial cost and maximizes Return On Investment (ROI) by clarifying requirements and taking reuse into consideration at the beginning of the project and not just at the end. It can also be applied to the programming language, network, middleware and operating system that you use.

One of the key concepts of MDA is the modeling of systems from different viewpoints. Each viewpoint is focused on particular concerns, an area of functionality or an area of interest to the system. These viewpoints can then be reused on different systems. An example of this would be the 1553B Bus. In UML, Packages are used to structure and group elements together. Figure 3 shows a Package hierarchy that contains text documents describing the background and test information, state diagrams documenting the protocol, class diagrams describing the data format, hardware diagrams detailing various hardware configurations using the 1553B, as well as other information. This viewpoint can then be used in other models. Viewpoints can also be concerned with all aspects of the system. For example, the system context, the system interfaces, the system internal design, system reliability to name but a few. To integrate these viewpoints, the relationships between the models must be clearly defined. A similar strategy can be used for components.



```
Avionics
  +Analysis
    +Logical
      +Bombadier
      +Cockpit
      +Concurrency
      +Input/Output
        +Analogue
        +Digital
          +1553-B
            +Bus_Controller
              +Buffers
                +Message Buffer
                +RT Read Buffer
                +RT Write Buffer
                +Transaction Table
              +Protocol
                +Bus Message
                +Bus Message Header
                +Bus Word
                +1553-B Protocol
              +Bus_Controller
              +The_Bus_Controller
            +Major Cycle Complete
            +Minor Cycle Complete
            +Read Buffer
            +RT Read Buffer
            +RT Write Buffer
            +Transaction Table
            +Write Buffer
            +1553 Bus Controller
            +Mission Computer
            +Remote Terminal
            +Mission Comupter - RT Interaction
```

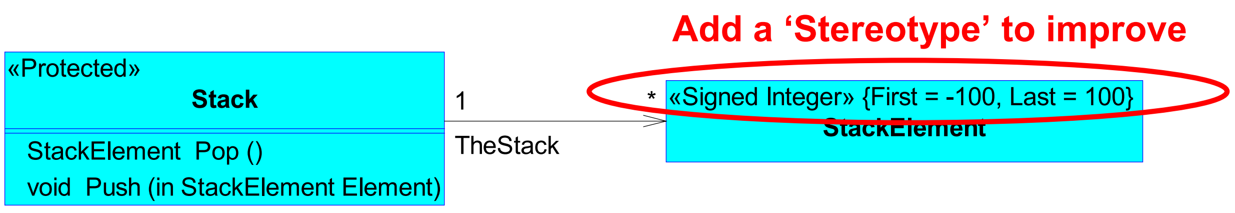
Figure 3 – 1553B Bus Packages Diagram

MDA Terminology

At the heart of the MDA are three models: the Platform Independent Model, (PIM), The Platform Specific Model (PSM), and the Generated Application. The Platform Independent Model expresses the system intent ***independently*** of any implementation technology. Regardless of the platform on which the system is implemented, this model will be valid. It will comprise the many different system viewpoints described earlier. The Platform Specific Model is created from the PIM and expresses functionality in a technology-***dependent*** fashion. Finally, the Generated Application is created from the PSM.

UML extension mechanisms can help to bridge between these various models. Figure 4 shows the very simple example of a stack as viewed by a systems engineer and a software engineer. Using the basics of the class diagram, it is simple for the systems engineer to model the stack. However, for the software engineer to implement it, he will need additional information. UML stereotypes can help clarify the stack requirements. A *stereotype* is a 'label' that can be used to 'characterize' a model item in some way. For example, an actor could be stereotyped as «Human», «System» or «Abstract». A *stereotype* is a model element that defines additional values (based on tag definitions), and additional constraints. These can be applied to any model element. In the previous example the <<Signed Integer>> Stereotype was applied to a Class called StackElement. The stereotype has two Tag Definitions 'First' and 'Last' which have corresponding Tagged Values of '-100' and '100' respectively. This additional information does not constrain the platform or language on which the stack will be implemented, but it does give additional valuable information to the software engineer. Without this information, the stack could be implemented incorrectly.

Figure 4 – Diagram Showing Example Stack Mechanism


Add a 'Stereotype' to improve
«Protected»
Stack
StackElement Pop ()
void Push (in StackElement Element)
1
TheStack
*
«Signed Integer» {First = -100, Last = 100}
StackElement

UML Profiles

Profiles provide a mechanism for managing stereotypes. In Real-time Studio, Profiles are created using UML packages. Because Real-time Studio allows packages to be exchanged between models (either directly or via a Configuration Management tool) the same Profile can be used across many projects. Using Profiles in this manner will ensure a standardized approach to problem solving within your organization.

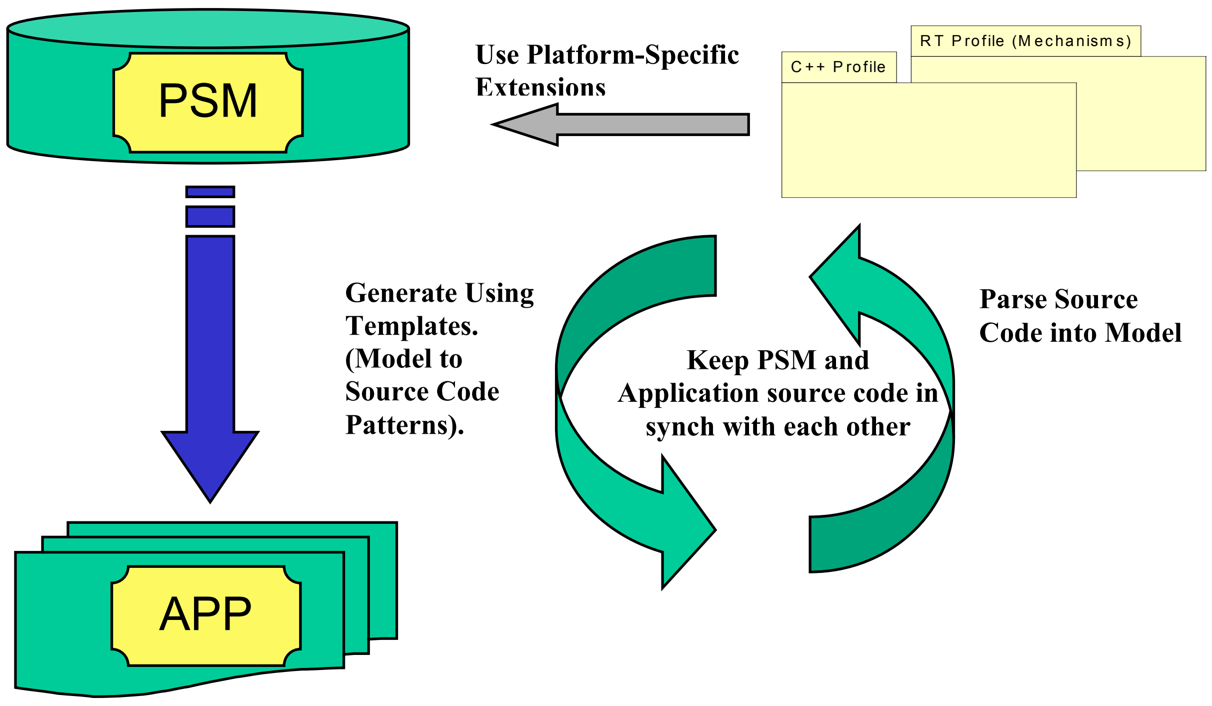
An example of an official OMG profile is the UML Profile for Schedulability, Performance and Time. It contains sub-profiles for: Time, Concurrency, Resources, Performance and Schedulability Analysis, and Real-Time CORBA. It also contains core concepts like Concurrency and Synchronization and mechanistic concepts like Clock and Queue. Alan Moore, ARTiSAN's Vice President of Product Strategy, is the chairman of the finalization task force for this profile. Most importantly, ARTiSAN Software Tools' Real-time Studio is the **first** tool on the market that fully supports this real-time profile, and includes it with the tool.

An example of a proprietary profile is the ARTiSAN C++ profile. As there is no official OMG C++ profile, ARTiSAN has created a C++ profile, which is used by the Real-time Studio code generators and reversers to capture C++-specific properties. ARTiSAN is currently developing profiles for Ada, C, CORBA, and Java. Unlike other tools, Real-time Studio allows you to access these profiles. You can customize them to your particular needs by modifying and adding to them. You can also create your own profiles. Real-time Studio also allows you to modify the code generation templates. This results in code generated to **your** company standards, and suited to **your** specific applications. With other vendors, you can have your code generated any way you want, as long as it's **their** way. Additionally, Real-time Studio does not limit you to one programming language per model. Integrating legacy systems could not be easier.

The Final Result

The Platform Specific Model is created by combining the solution profiles, including the language, hardware, specific components and the system profiles. For the Generated Application, the developers will use platform-specific tools to build the application from source. These will include code generators, compilers, debuggers, and test platforms as shown in Figure 5. Most modeling tools cannot show the Platform Specific Model; platform specific elements such as the code generation or operating system are either built into the platform independent model, or the code generators do not allow access to this information. Real-time Studio, however, allows you to model the PIM, PSM, Profiles and Generated Application independently. With Real-time Studio, you can even make changes to the Generated Application and have these changes reflected all the way back to the PIM using Real-time Studio's Round-Trip Engineering. Most other tools provide only a one-way trip from model to code, a real handicap to your developers.

Figure 5 – Real-time Studio and MDA


PSM
Use Platform-Specific Extensions
RT Profile (Mechanisms)
C++ Profile
Generate Using Templates. (Model to Source Code Patterns).
Keep PSM and Application source code in synch with each other
Parse Source Code into Model
APP

Future Developments

ARTiSAN will continue improving its support for MDA with an exciting new addition called the Pattern Wizard. ARTiSAN's Pattern Wizard goes beyond the application of just profiles… it allows you to apply patterns as well. In addition to the Pattern Wizard, we will also be providing many of the most well known patterns in the Real-time industry, helping you to leverage the knowledge base of the entire real-time community. As these patterns will be modeled using Packages, Real-time Studio allows you to populate your project with the patterns that you need with just a few mouse clicks. Unlike other tools, the results of the application of the pattern will be visible in the diagram instead of hidden inside a proprietary framework. As a result, if you find that pattern is not suitable, the Pattern Wizard can undo the applications of the pattern. You can also create your own patterns and you will be able to safeguard and reuse your intellectual property now and in the future. This makes for an authentic ROI for your company. With the ARTiSAN Pattern Wizard, the realization of MDA could not be easier: apply the necessary Profiles to the PIM, thus creating the PSM, and generate your code. Change platforms, components, languages and operating systems almost as easily as waving a magic wand - truly magical!

Conclusion

ARTiSAN's support for MDA will continue to be one of the best in the industry. Real-time Studio goes beyond simple code generation; it models the whole system, allowing a complete view of all aspects of the system from requirements to test. Our configuration management, and repository-based product architecture means that your modeling investment is truly safe. Can your company afford to experiment with tools that don't?

References

Model-Driven Architecture and Integration Opportunities and Challenges Version 1.1
Desmond DSouza, Kinetium

Model Driven Architecture:An Introduction
Richard Mark Soley, Ph. D.

Developing Real-time Systems using Object Technology
Alan Moore, ARTiSAN Software Tools

Rebuilding the Tower of Babel - The Case for UML with Real-time Extensions
Matthew C. Hause, ARTiSAN Software Tools