BAE SYSTEMS

Model-Based Techniques in the Development of Net-Centric Applications

June 20, 2007

Timothy A. Anderson

Basil C. Krikeles

BAE-Systems

Advanced Information Technologies

6 New England Executive Park

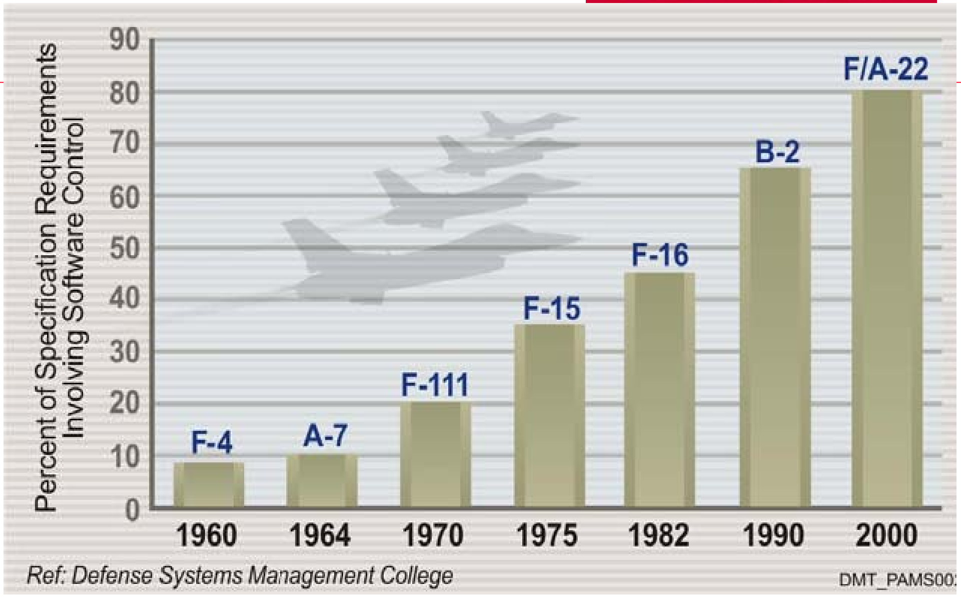
Burlington, MA 01803

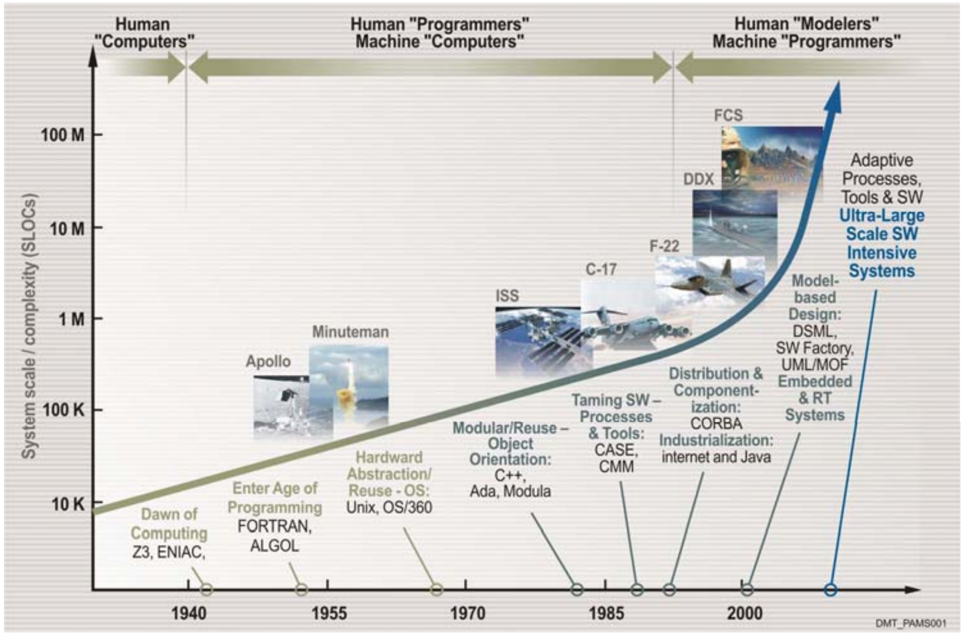
This work was sponsored by the Air Force Research Laboratory (Contract # F30602-01-D-0053 Order No. 0012, subcontract S06021). Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the United States Air Force.

Why is it important to develop next-generation software technologies such as MDA?

BAE SYSTEMS

- Size and complexity of software systems rapidly increasing beyond our capacity to handle with current technology
- Functionality moving from hardware to software (flight control, radios)
- Networked – evolving constellation of systems to integrate with
- Underlying technologies and infrastructure rapidly evolving
- Requirements rapidly evolving
- For complex systems it may not be possible to accurately capture system requirements, especially in software with long lifecycles
- Need to protect large software investment by ensuring software can be managed, maintained, extended, modified, and adapted over a long period of time by teams other than the original developers
- MDA supports “Software Producibility”



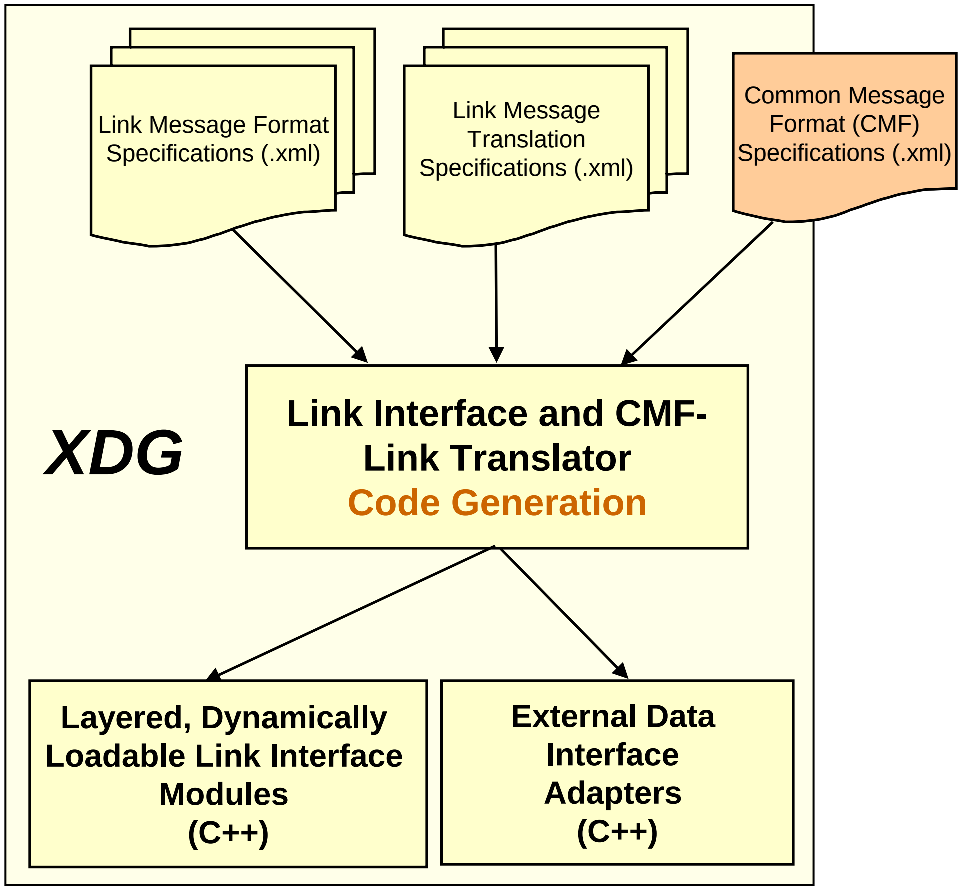


Model Driven Development, xUML, DSLs

- Model-driven development is focused on a representation of the functionality and behavior in of a system that is independent of the ultimate implementation technology, language or infrastructure
  - Relies on automated or semi-automated means of transforming the model into a platform specific executable
- Executable UML (xUML) comprises a subset of the UML standard with sufficiently precise semantics to be capable of being executable
  - XUML attempts to provide a generic MDA solution, i.e. one that works in any application domain
- Domain Specific Languages (DSLs) implement MDD in the context of specific problem domains, and can be optimized for those domains

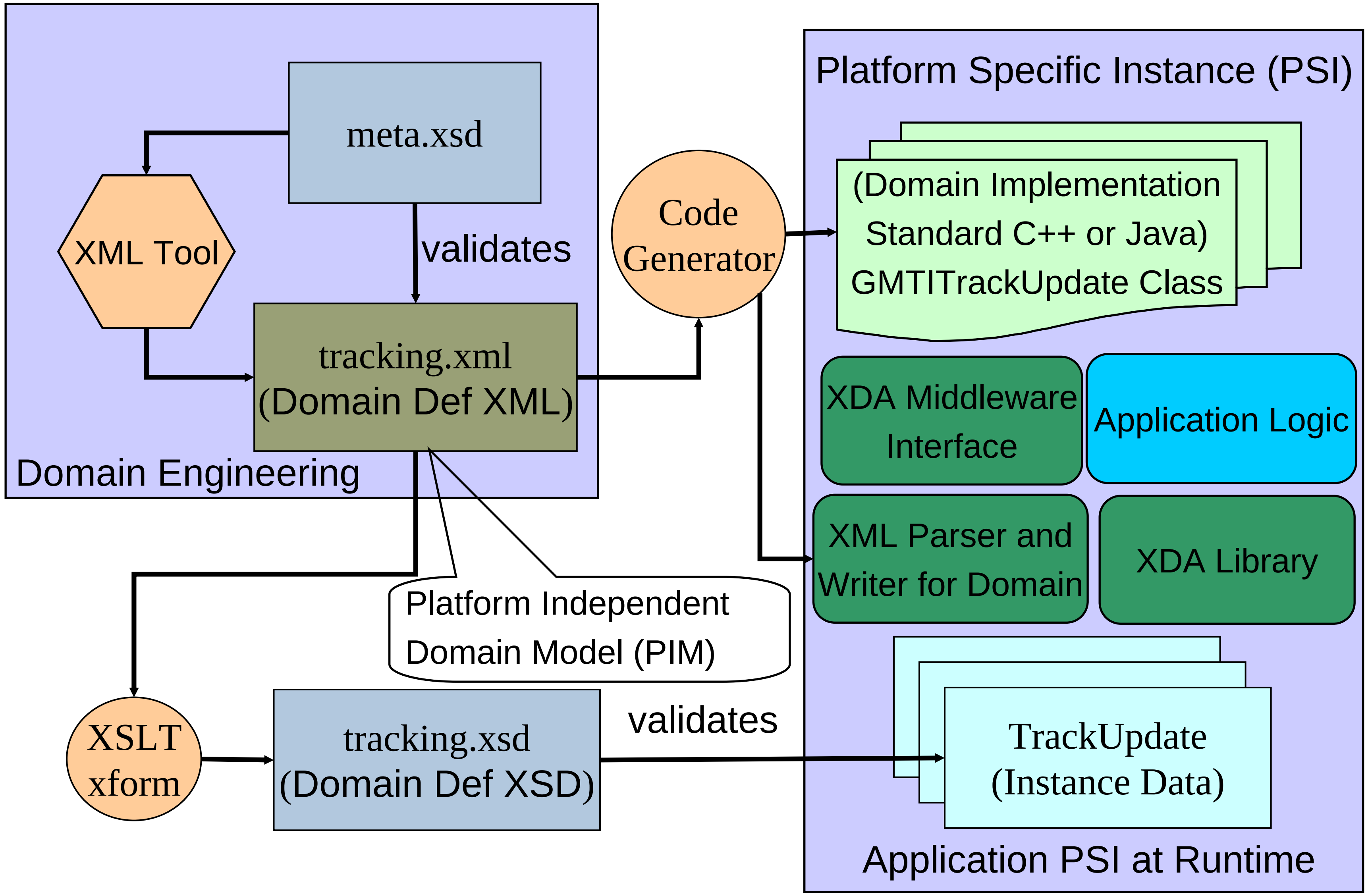
XML-defined Gateway for Tactical Data Links

BAE SYSTEMS

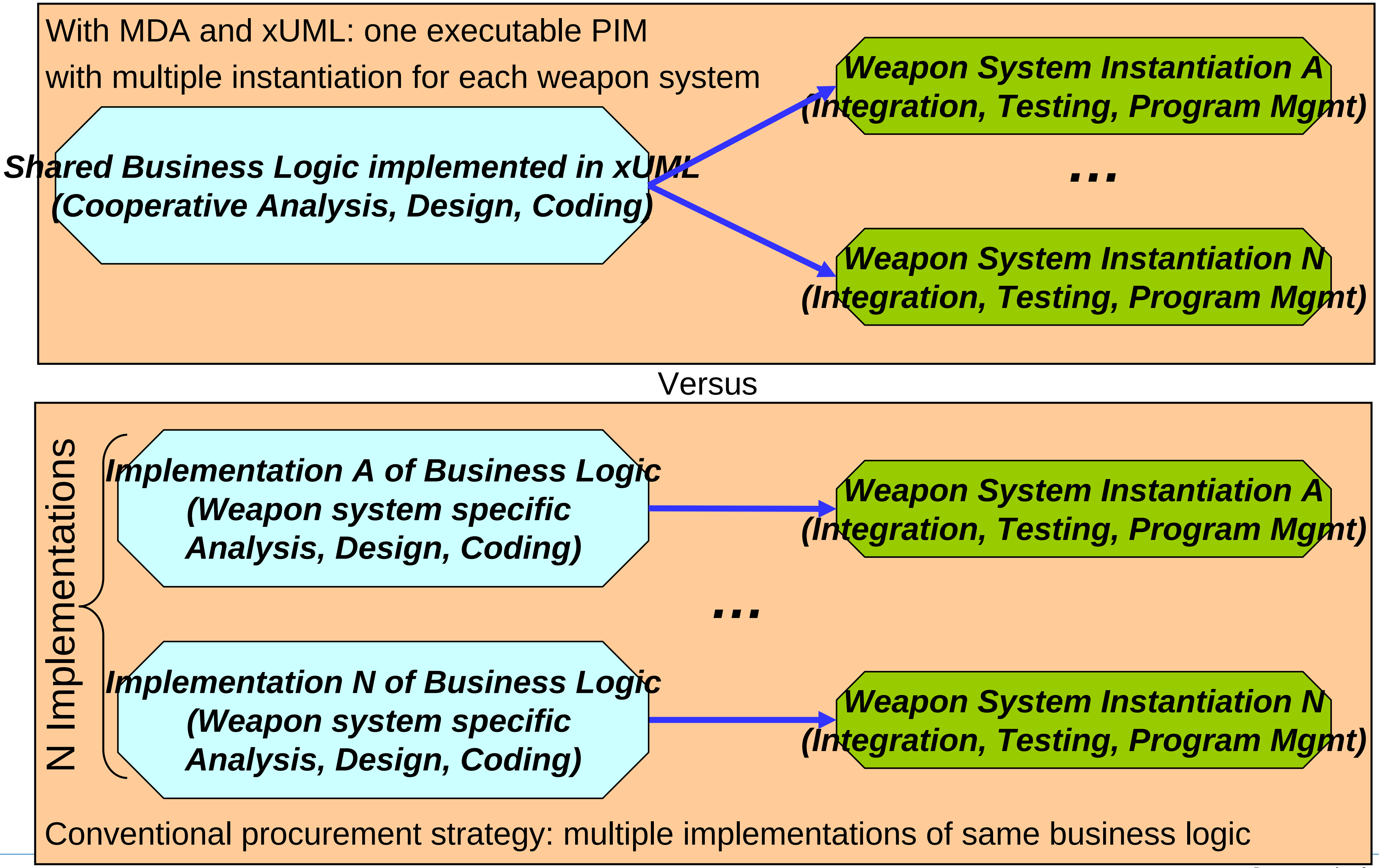

Link Message Format Specifications (.xml)
Link Message Translation Specifications (.xml)
Common Message Format (CMF) Specifications (.xml)
XDG
Link Interface and CMF-Link Translator
Code Generation
Layered, Dynamically Loadable Link Interface Modules (C++)
External Data Interface Adapters (C++)

- TDL specifications are generally human-readable, making implementation V&V difficult.
- Several data links (Link-11, Link-16, etc.) are in active use. Each one requires robust, high-performance support.
- Message forwarding among a variety of TDLs can be an N^2 implementation problem: each pair of network types can require a module to handle translation.
- XDG represents TDL data formats and rules in XML, from which it generates a common data representation and link-specific translators and forwarders.

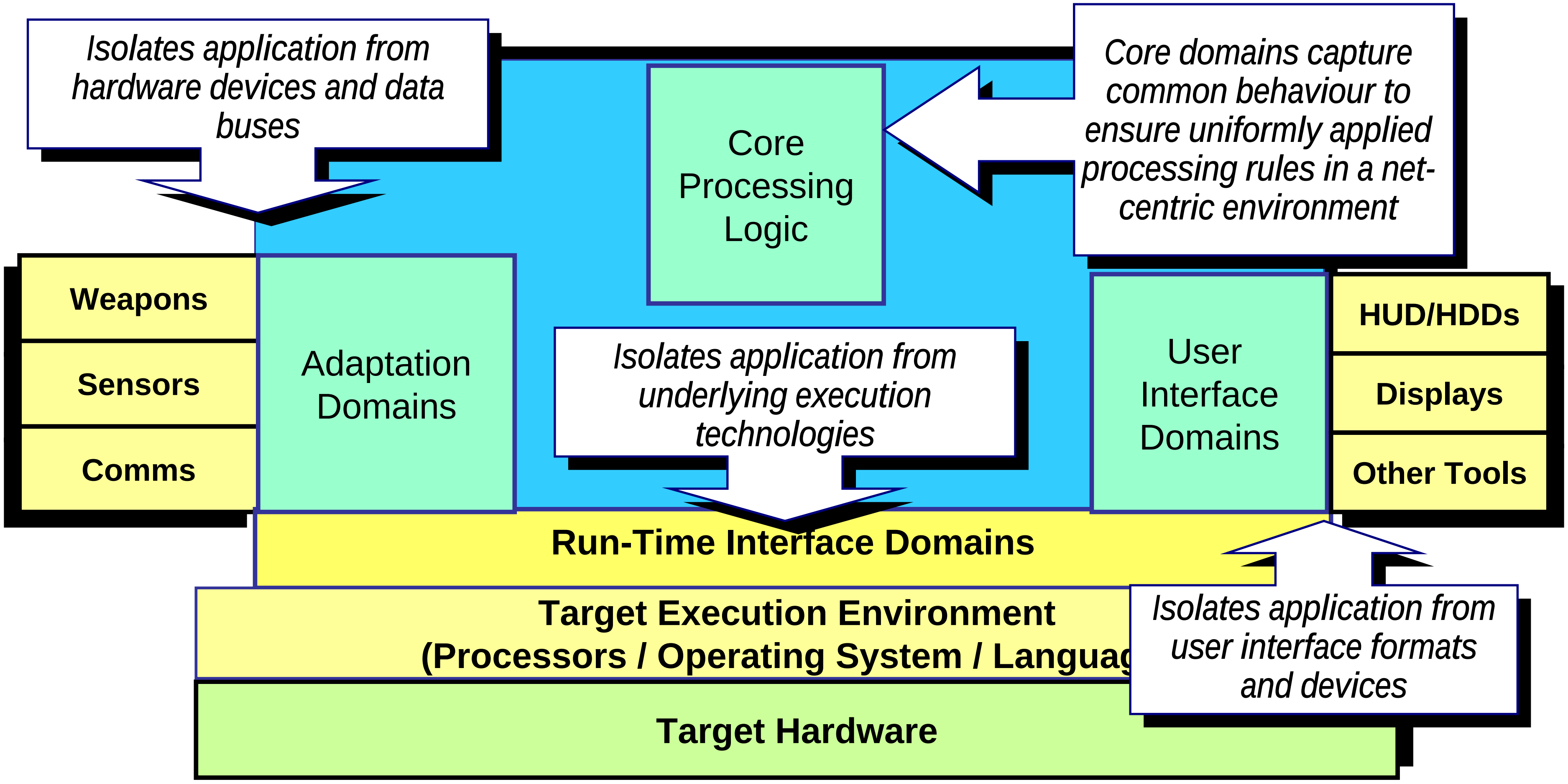
XDA DSL: A Model Driven Framework for Domain Engineering with Auto-generated XML parser



Our case study and its value proposition


With MDA and xUML: one executable PIM
with multiple instantiation for each weapon system
Shared Business Logic implemented in xUML
(Cooperative Analysis, Design, Coding)
Weapon System Instantiation A
(Integration, Testing, Program Mgmt)
...
Weapon System Instantiation N
(Integration, Testing, Program Mgmt)
Versus
N Implementations
Implementation A of Business Logic
(Weapon system specific
Analysis, Design, Coding)
Weapon System Instantiation A
(Integration, Testing, Program Mgmt)
...
Implementation N of Business Logic
(Weapon system specific
Analysis, Design, Coding)
Weapon System Instantiation N
(Integration, Testing, Program Mgmt)
Conventional procurement strategy: multiple implementations of same business logic

Platform-specific deployment


Isolates application from hardware devices and data buses
Core domains capture common behaviour to ensure uniformly applied processing rules in a net-centric environment
Core Processing Logic
Weapons
Sensors
Comms
Adaptation Domains
Isolates application from underlying execution technologies
User Interface Domains
HUD/HDDs
Displays
Other Tools
Run-Time Interface Domains
Target Execution Environment
(Processors / Operating System / Languag
Isolates application from user interface formats and devices
Target Hardware

Configuration Management Issues

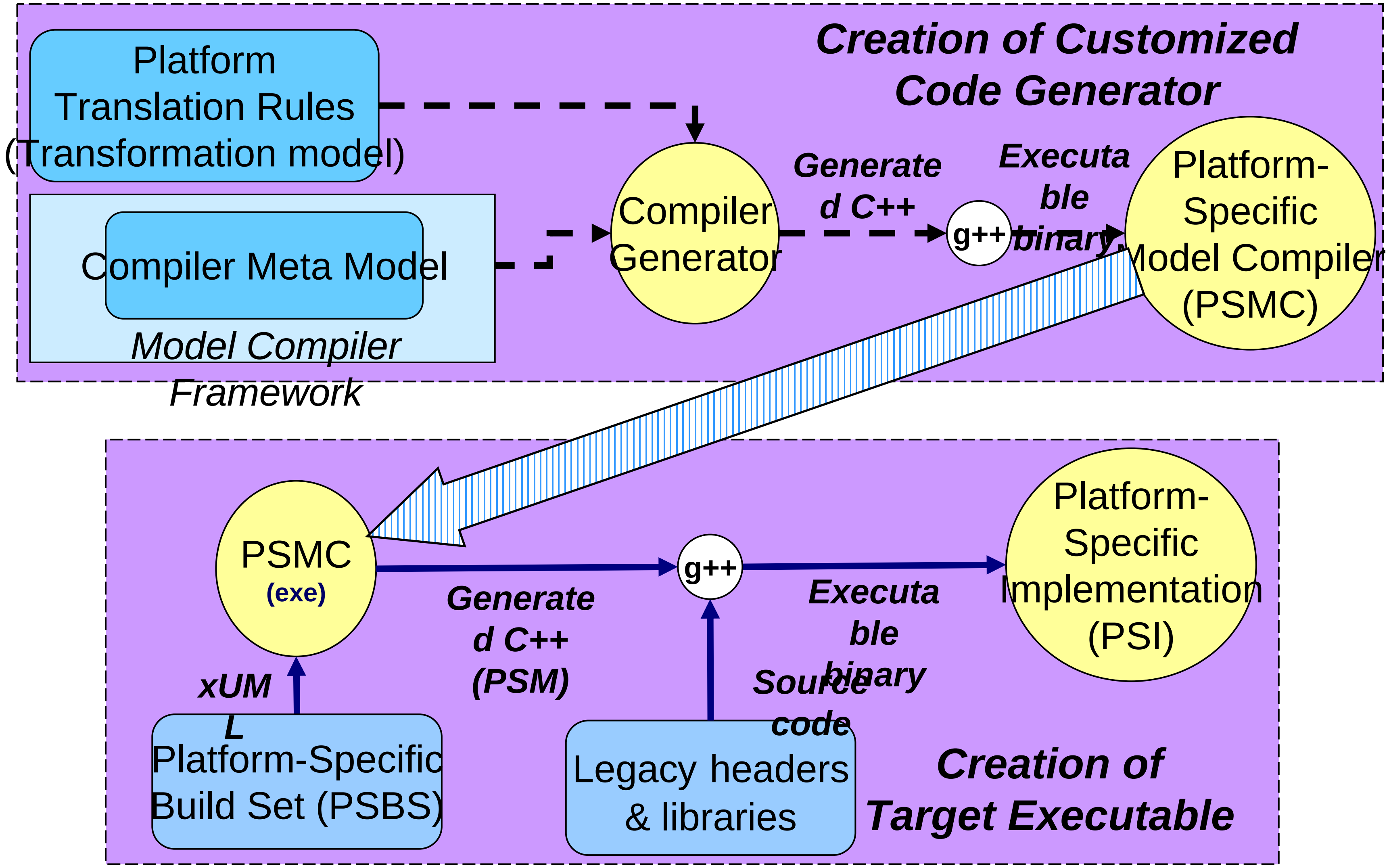
BAE SYSTEMS

- Effective tools have been developed to support text-oriented software development, for example
  - Concurrent modification of the same source file by multiple developers
  - Source code differencing
  - Semi-automated merging of different versions of the same source file
  - Automated patching of released versions
  - Branching of the source tree and merging of different branches
- None of these tools exist in usable form for Model Driven Development, resulting in hard to overcome problems for large MDA projects
  - Development must be centralized
  - One small team of two to three developers must be assigned to each unit of re-use (domain, or package or set of classes) in order to serialize modifications and to de-conflict access by direct communication
  - Patching a release of the model is virtually impossible; significant defects typically force a re-release of the modified model
  - Domain interactions happen in “Bridges” across domain boundaries and are even harder to manage
- The next generation of MDA technologies must address this issue

Unintended Consequences

- Use of the relational data model and specialization hierarchies rather than inheritance reduces performance and developer effectiveness
- Expressivity of the Action Specification Language is not adequate for many algorithms
  - An abstract language may be adequate for describing an algorithm, but not expressive enough to capture optimizations and other implementation-specific details needed for system performance
- Use of abstract state machines and event queues can yield fragile, inefficient code

Model Compilation and PSM Generation


Creation of Customized Code Generator
Platform Translation Rules (Transformation model)
Compiler Meta Model
Model Compiler Framework
Compiler Generator
Generated C++
g++
Executable binary
Platform-Specific Model Compiler (PSMC)
PSMC (exe)
xUML
Platform-Specific Build Set (PSBS)
Generated C++ (PSM)
g++
Source code
Legacy headers & libraries
Executable binary
Platform-Specific Implementation (PSI)
Creation of Target Executable

Lessons learned from current experience with MDA

BAE SYSTEMS

- MDA technology is in its infancy but holds tremendous promise
- Standards are being developed
- MDA has been successfully applied in the well-scoped area of Domain Specific Languages
- Effective general purpose MDA tools will require the following:
  - A next-generation standard for a subset of UML that includes execution semantics and effectively addresses all aspects of software modeling including runtime model, threading/concurrency, state machines and multi-processing
  - Ability to represent different levels of detail within a model according to the user’s interest
    - Subject matter experts can see a view appropriate to them, while developers and modelers can drill down to implementation details
  - Industry accepted standards enabling cross-vendor interoperability
  - Additional research to address version control and configuration management in the context of MDA and to support concurrent model modifications and distributed model development

Suggestions for improving the next xUML standard

- Include a new approach to domain interaction that will eliminate the tight coupling induced by bridges and inline code, truly hide the internal implementation details of domains
- Enable a heterogeneous approach to modeling to allow for efficiencies and optimizations at different levels of representation
- Do not relegate platform-specific and optimization issues to the model compiler. The modeling language, if appropriate for the granularity of the model, should be able to abstract and capture both platform issues and optimization strategies.
  - The strategy of relying on the model compiler is both unrealistic and brittle
- The runtime architecture should not be drastically different from current state of the art. The Relational Database Model used by the current version of xUML is confusing, error-prone and inefficient
- Include interoperability standards. The market place will force tool vendors to comply allowing developers to mix and match MDA tools and to create customized modeling environments with the tools best suited for their task
- Try to emulate at the MDA level the utility of the Unix constellation of file-based tools and the success of the Eclipse “eco-system” of development tools

Conclusion

BAE SYSTEMS

- As software projects continue to grow in size and complexity, MDA can be a valuable approach to managing the development
- With current technology, MDA is especially effective when more narrowly focused and used in conjunction with domain-specific modeling languages.
  - Standards are still being developed
  - General tools are either limited in expressive power or lack platform independence
- Cross-tool interoperability standards are critical, in order to allow competition, evolution, and use of best-of-breed tools
- New technologies for configuration management are required to support effective development
- It is extremely difficult to provide a single tool that supports all levels of development
  - Different levels of detail could require different modeling paradigms
  - Interoperability will be required between tools at different levels of detail on the same project
- Final thought: a model is an artifact that can be used beyond code generation, to produce documentation, complexity or producibility statistics, etc.