DataAccessTechnologies

Where Business Meets Technology

MDA Vision

For

Government and Defense

The Model Driven Architecture Opportunity

Government and defense systems are undergoing a shift from independent hardware solutions to integrated software solutions that enable network centric warfare and collaborative homeland security. Organizations able to deliver on these visions will be those that excel and prosper in the new millennium. The following is a introduces a plan to capitalize on this opportunity based on an emerging technology, Model Driven Architecture.

Driving Requirements

Complex and unique systems must be understood, designed, proposed, executed and delivered with factory-like efficiency and compelling results. Despite the improvements in technologies and processes, proposing and delivering solutions is still expensive and high-risk. Without the capability to deliver compelling solutions opportunities are being missed.

The typical government project of today combines several challenges;

- Large system complexity
- Changing requirements
- Fluid deployment platforms
- Short development cycles and limited budgets
- Nessesity to integrate with legacy technologies
- Leading-edge functionality embracing operational vision such as network warfare and global integration
- Expectation of long life-cycles in the face of changing requirements and technologies
- Evolution without re-design and re-implementation
- Controlled and reasonable total cost of ownership
- Ad-hoc partnerships with other suppliers
- Mature and repeatable processes

Current processes and techniques evolved from an assumption that requirements and supporting technologies would be relatively constant over the life of a project or a system

– this assumption is simply not valid today. A central theme of the above challenges is change – we must *design for change* with agile and adaptable tools, techniques and processes. Systems must evolve with changing requirements, designs, infrastructure and platforms. This evolution must be fast, efficient and inexpensive.

The evolution of software

Since the first symbolic assembler to the latest visual tool software has been evolving along a few fundamental directions;

- Higher levels of abstraction – the move of software away from the technology and closer to the problem domain. This can be seen in high-level languages, symbolic databases as well as modeling and knowledge based systems.
- Separation of concerns – the move from tightly coupled monolithic systems to loosely coupled systems where different components of and viewpoints on the system can be independently described and evolved.
- Industrial processes – the move from un-restrained creative approaches to every problem to well defined, reliable and repeatable processes.
- Reuse – the move from one-off solutions to systems as compositions of reusable components.

This transition from an infant technology to a mainstream industry is now converging in the form of the Model Driven Architecture with the supporting standards, tools, technologies, methodologies and processes.

The Model Driven Architecture (MDA) approach

The basic tenants of MDA are simple – to move the focus of systems development from software artifacts to high-level models. High level models are tuned to the problems being solved, resonating with domain experts. Provisioning processes are used to automate much of the process of producing the system in terms of technology artifacts, documentation and process support. These high-level models and provisioning processes are supported by open standards and COTS tooling. In MDA the heart of the system is models, not code.

The Choice of MDA as the System Development Approach

Government systems embrace a long-term strategy that must survive changes in objectives, approach and technology. In the short term the technical foundation of systems is frequently fluid – systems, interfaces and technologies will be changing as the system is developed. To support short-term fluidity and long term stability systems must be designed for change. Only by designing with the expectation of changing requirements and changing technologies can systems fulfill their goals now and into the future, and be delivered on-time with minimal risk.

To support design for change we will employ a strategy that separates technology concerns from domain concerns – since both will change over time independently. A Model Driven Architecture (MDA) has proven to be the best approach to supporting

design for change. MDA describes the system in terms of models and ontologies[1] at both the domain and technology levels with an automated process for creating system implementations, tests and documentation. MDA is based on a set of emerging standards from the Object Management Group (OMG)[2], leveraging the OMG's leadership in open systems, interoperability and UML.

To support design for change with MDA, systems will be specified in terms of high-level ontologically aware models and that describe domain problems in domain terms, independent of the implementation technology. Another set of models defines the technology (Application servers, Grids, Agents, and legacy systems) in technology terms. An automated provisioning[3] process is used to transform these high-level ontologically aware domain models into the executable and deployable components that execute the system. MDA the following advantages:

- Support for a fast-iterative spiral development process supported by automated provisioning
- Single point of change with automated propagation to and validation of all system artifacts
- An agile full life-cycle approach that ties together requirements, analysis, designs, implementation, tests, monitoring, documentation and the executing system
- Rapid, reliable and reproducible development due to the automated provisioning of systems from models
- Support for evolution and refinement of the technical architecture over time without re-implementing or re-designing the core domain logic – including runtime specification of rules, processes and policies
- Automated production of code, configurations, tests, validation and documentation
- Support for the automated integration of diverse new and legacy technologies
- Reduction in total lifecycle cost, time and labor due to automation and reuse
- Reduced risk of technology lock-in and obsolescence
- Increased ability to optimize the technologies and architecture before, during and after deployment
- Reduced risk of project failure due to the fast, iterative process and increased flexibility in technology choices and domain model evolution
- Use of standards integrates with existing processes supported by open and readily available resources

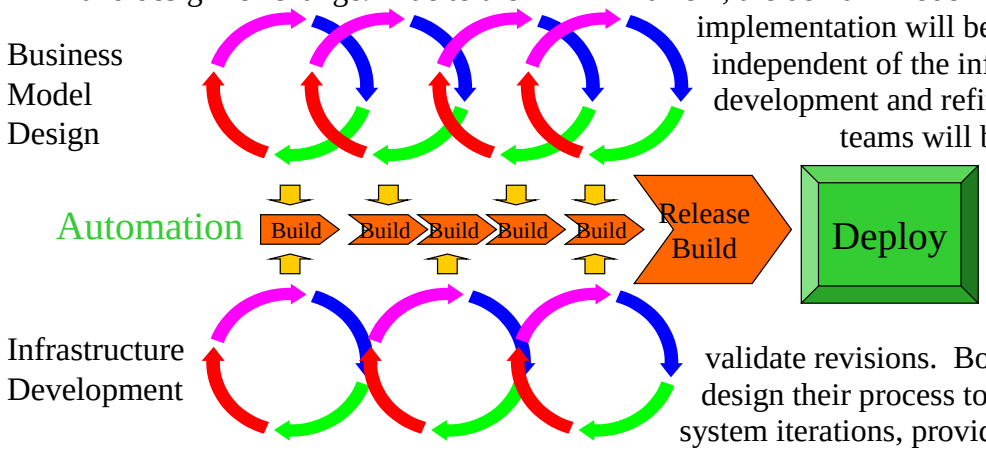
[1] An ontology defines the terms in a domain and how those terms relate to each other
[2] Object Management Group (OMG) – www.omg.org
[3] Provisioning, in this sense, is the process of transforming models into systems

- Reduced lock-in to specific vendors or tools by using standards-based models, IDEFx and UML
- Increased ability to evolve and adapt domain models, rules, and processes

For the above reasons, a Model Driven Architecture is central to a plan to address the requirements for a **high degree of flexibility while reducing cost and risk**. The combined leverage of **early and incremental implementation** combined with **automated and repeatable testability** provides a **profound and lasting benefit** to the effectiveness of the entire system for its entire lifetime.

Iterative spiral process

Systems will be developed in an iterative spiral process allowing for maximum flexibility and design for change. Due to the MDA "Buffer", the domain modeling and implementation will be largely independent of the infrastructure development and refinement. Both teams will be able to iterate on their own schedule, using the provisioning capability to build and validate revisions. Both teams will design their process to converge at major system iterations, providing early visibility of results to the team and to the customer. This iterative spiral process will take place before, during and after initial deployment – allowing systems to improve over time.

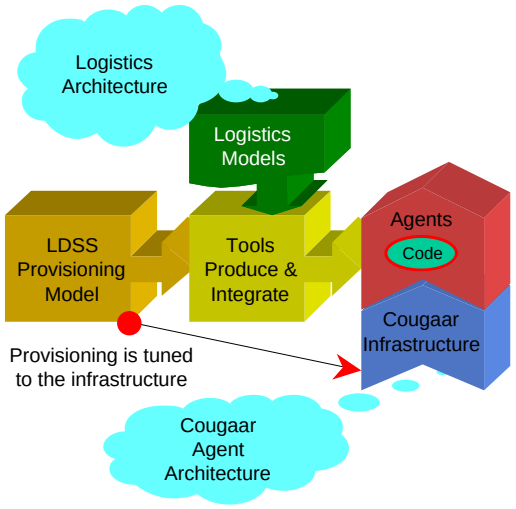


How MDA works

MDA is used to create and maintain systems based on high-level models of the subject domain. MDA separates the concerns of the operational domain architecture from the technical infrastructure.

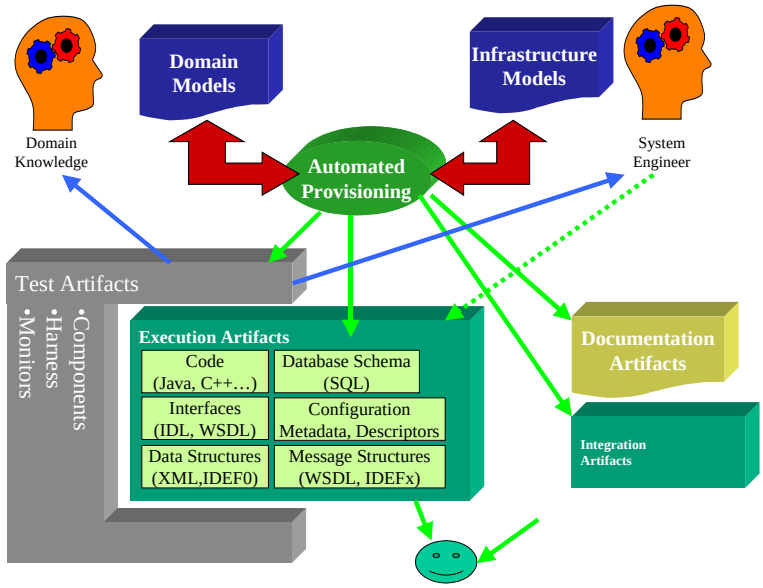
The operational domain architecture defines the concepts and structures necessary to specify and understand the domain (such as domain or strike architectures). This architecture includes the terms and concepts of processes, rules, policies, resources, units of action, and capabilities of the domain so that models may be defined and refined in high-level terms. The core domain models will be provided with the system along with tools to maintain and extend these models as the system matures – including field specification and refinement of policy and processes. These models represent the metadata that governs the system at design time and runtime.

A "Provisioning model" will also be provided with the system, this provisioning model will define the processes and transformations required to implement the domain using the systems infrastructure. The provisioning model is used to configure provisioning tools that are able to produce executable system components based on the domain models. This automated MDA process will produce a majority of the manual implementation and provide templates into which programmers place program code for complex algorithms that cannot be easily generated by the automation.



The domain models, infrastructure and provisioning model are able to evolve independently, providing the separation of domain concerns from technology concerns and reduction of risk that are the cornerstones of MDA. In addition to producing implementation components, the same provisioning technology will be used to automate the production of test cases and documentation. Requirements, models, documents and the system are always synchronized, traceable and consistent.

MDA Automated provisioning is based on models provided be both domain experts (with easy to use user interfaces) and systems engineers providing the technology and infrastructure specification. Automated MDA provisions most of the execution artifacts, test artifacts, documentation, integration specifications and runtime configuration policies and processes that produce and drive the system.



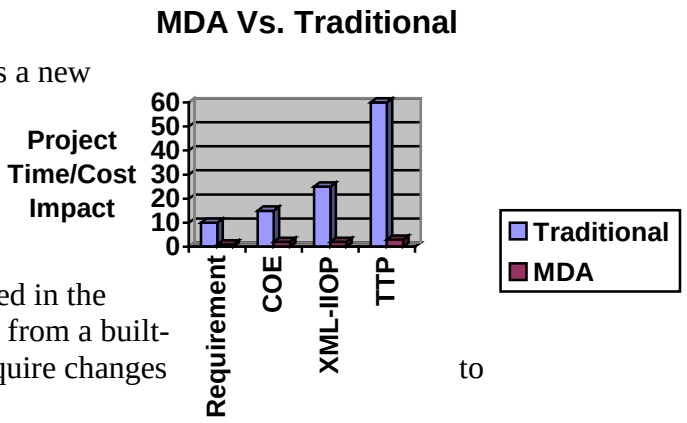
The domain architecture will based on the adopted OMG standards of UML, the "Meta Object Facility" (MOF[4]) and the "Enterprise Collaboration Architecture" (ECA). UML provides for general modeling capability and is extended with the ECA to provide for flexible and composable distributed components representing the interactions between actors in the operational system. Multiple existing MDA tools and technologies will be leveraged and extended for the needs of complex systems.

[4] A "MOF" (Meta object facility) holds models and meta-models in a shared and versioned repository. Tools and agents are able to maintain and query the models in the MOF repository at design time, test time and runtime.

MDA over the system life-cycle

A primary advantage of MDA is facilitation of change before, during and after deployment. The separation of concerns between the domain view and the technology view allows each to evolve independently. A change to either a domain model or a technology (provisioning) model will automatically be propagated across all systems artifacts. Compare this to the manual case where such a change could well cause re-development and testing of thousands of artifacts. Blanchard has estimated that a requirement introduced at each subsequent lifecycle step increases the cost of a change by a factor of 10. MDA largely eliminates the Blanchard effect.

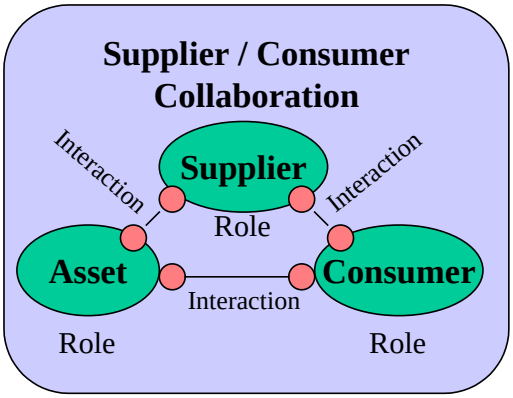
The following example is based on a typical project; consider a system where the following occurs;



- Half-way through the development process a new requirement is added for context sensitive security, affecting the implementation of existing components. System design and 200 artifacts require change, testing and re-integration.
- After the 4th build a new service is identified in the operating environment, requiring a change from a built-in service. 500 code and other artifacts require changes to use the new service.
- During system performance validation the use of XML is found inappropriate for a core service and a change is made to IIOP. Every agent requires a minor change testing and re-integration.
- After deployment a change in the Tactics, Techniques and Procedures (TTP) causes a re-design of a logistics service. System design and 900 system artifacts require changes. Other systems that have built-on the system being developed also require changes, testing and re-integration.

Role Based Collaboration

Systems of systems require people, organizations and systems to collaborate for planning and execution. A high-level view of such systems of systems can be understood in terms of these collaborations, the roles played by people, systems and organizations within these collaborations and the interactions between them. The collaboration of components playing roles can be understood and modeled using existing standards - the Object Management Group (OMG) has adopted the "Enterprise Collaboration Architecture" (ECA) as part of the "Enterprise Distributed Object Computing" (EDOC) standard. ECA specifies how to model collaborative processes in a way that is sufficiently well defined to drive the execution of systems. In addition to the modeling of collaborations, ECA describes how collaborations at one



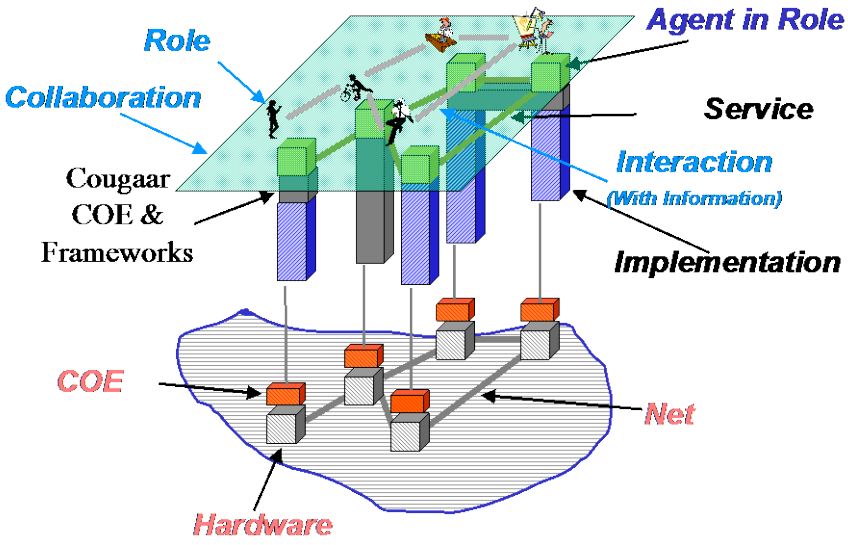
level “drill down” into sub-collaborations. Each role in each collaboration becomes a component in the system, separately implementable and deployable in components.

Collaboration modeling for systems of systems will include the specification of collaborative processes, roles within these processes and the interactions between them. Interactions specify message formats, sequencing, timing, security and quality of service. Collaboration roles will also be able to include compositions of components filling other roles so that processes can “drill down” into sub-processes.

Components representing people, platforms, organizations, automated systems are described in terms of their roles within the logistics processes. These roles have ports that are their communication points to other roles. In the systems environment each role has corresponding agents, which plays that role in terms of the systems environment.

In the collaborative process model roles interact in a well defined way to implement the logistics processes. In the systems environment, each port on an agent’s role (corresponding to a communication channel of the role) corresponds to a service for two-way conversation between agents. This interaction may be implemented across a variety of local and distributed communication media. Components representing roles in collaborations will be provisioned as components as the execution environment. The execution environment can embrace Grid, Agent or application server technologies.

Communications between components may be either point-to-point services or event based.



Roles to systems

Roles are performed by the components configured to play a given roles. The implementation of agents able to perform a given roles is provided by the MDA provisioning environment. These implemented and configured agents playing roles are deployed on systems platforms in the executing system. It is important to recognize that components, like people and organizations, play multiple roles at the same time.

The mix of organizational, platform and technology components

While the high-level picture shows only logistics roles, the executing systems environment will include other components as well. These will be system services, repositories, transformations, adapters, transaction coordinators, business objects, messaging queues and other components to support the domain roles.

In a systems sense the technology components and business components have the same “shape” in the system, but they are designed at different layers.

Roles are also defined for legacy systems, providing an integration point for components and legacy to co-exist under a common environment and paradigm.

Summary

MDA well be a necessary ingredient for achieving the goals of network centric warfare and integrated homeland security. The combination of enterprise architecture, role based collaboration and component reuse provides solutions early while supporting long term strategies.