Knowledge-Based Analysis and Design (KBAD)

A methodology for rapid, cost-effective system engineering and architecture development

Presented October 22, 2008
Overview of presentation

• Why yet another “methodology?”
• What is KBAD?
• What theory underlies KBAD?
• What kind of tools work with KBAD?
• What process does KBAD implement?
• What kind of people do we need to execute KBAD?
• How do we move from drawing pictures to building a knowledgebase?
Why Yet Another Methodology?

• We have the DoD Architecture Framework …
  • But DoDAF isn’t a methodology, it’s just a description of necessary products

• We have UML …
  • But UML is only a software engineering technique. You have to come up with the process and tools for implementing it

• We now have SysML …
  • But SySML is just another technique and still needs more definition to create complete, executable designs

• What’s missing?
  • A complete, coherent technique, process, and tool set that results in a knowledge base that can be used for full lifecycle decision making
Knowledge-Based Analysis and Design

• KBAD combines system engineering and program management disciplines to enable the development of a knowledgebase that can enable cost-effective decision making

• KBAD spans the acquisition lifecycle enabling support for design, development, integration, test, operations and sustainment

• KBAD focuses on using a variety of techniques and tools, brought together in a common database using special software to migrate data between tools

• The KBAD process links the technique and tools together in an executable, cost-effective way to support decision making at all levels

KBAD reduces costs and increases speed of delivery by simplifying the data captured and focusing on the analyses needed for design. The result: a knowledge-base for decision making.
What makes up KBAD?

• **Technique**
  • Modified Model-Based System Engineering (MBSE)

• **Process**
  • SPEC’s Middle-Out Process for Architecture Development and System Engineering

• **Tools**
  • A variety of COTS tools tailored to the MBSE modifications and special needs of DoDADF

• **People**
  • Trained, experienced professionals who bring a wealth of different backgrounds and knowledge in architecture, system engineering, modeling & simulation, physics, computer science, test & evaluation, operations & support

KBAD was developed over the past 15 years and brings lessons learned from those years of experience.
The technique: refined MBSE

• Various forms of model-based system engineering have been developed

• SPEC uses the one developed by TRW in the late 1960s, which has been successfully used since then

• SPEC has refined this technique by simplifying the information collected (entities, relationships and attributes) and adding a number of key elements missing from the original development
1. Logical architecture (behavior) model
   - Functional sequencing
   - Data flow and size
   - Resource model
   - Evolution in time

2. Physical architecture (asset) model
   - Interface definition (bandwidth and latency)
   - Actions allocated to Assets
   - Data allocated to interfaces
## Models are based on language

<table>
<thead>
<tr>
<th>Language Elements</th>
<th>English Equivalent</th>
<th>KBAD Schema Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Noun</td>
<td>• Statement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Asset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ...</td>
</tr>
<tr>
<td>Relationship</td>
<td>Verb</td>
<td>• Statement is the <strong>basis</strong> of an Action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• An Action is <strong>performed by</strong> an Asset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ...</td>
</tr>
<tr>
<td>Attribute</td>
<td>Adjective</td>
<td>• Description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Type (e.g., Operational Activity is a type of Action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ...</td>
</tr>
<tr>
<td>Attribute of Relationship</td>
<td>Adverb</td>
<td>• <strong>amount</strong> of Resource consumed by an Action</td>
</tr>
<tr>
<td>Structure Enables Executability</td>
<td>Graphics/Drawings</td>
<td>• <strong>acquire available</strong> (hold partial) Resource for Action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Graphic Views: Behavior, Hierarchies, Physical Block</td>
</tr>
</tbody>
</table>
We modified Vitech’s schema

<table>
<thead>
<tr>
<th>KBAD Element</th>
<th>CORE Elements</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Function/Operational Activity</td>
<td>Provide overall class for actions</td>
</tr>
<tr>
<td>Artifact</td>
<td>Document</td>
<td>Recognized not just documents</td>
</tr>
<tr>
<td>Asset</td>
<td>Component/Operational Element</td>
<td>Provide overall class for assets</td>
</tr>
<tr>
<td>Characteristic</td>
<td>type of Requirement</td>
<td>Way to capture metrics and other characteristics of an element</td>
</tr>
<tr>
<td>Cost</td>
<td>attribute of Component</td>
<td>Broadens capture of costs</td>
</tr>
<tr>
<td>Input/Output</td>
<td>Item/Operational Information</td>
<td>Clearer name</td>
</tr>
<tr>
<td>Issue</td>
<td>Issue</td>
<td>Same</td>
</tr>
<tr>
<td>Link</td>
<td>Link/Needline</td>
<td>Provide overall class for transmission</td>
</tr>
<tr>
<td>Location</td>
<td>none</td>
<td>Captures geolocation information</td>
</tr>
<tr>
<td>Risk</td>
<td>Risk</td>
<td>Same</td>
</tr>
<tr>
<td>Statement</td>
<td>type of Requirement</td>
<td>Clearer name</td>
</tr>
<tr>
<td>Time</td>
<td>attribute of Function</td>
<td>Broadens capture of times</td>
</tr>
</tbody>
</table>

The goal was to simplify and clarify the language.
We related all the KBAD schema elements

<table>
<thead>
<tr>
<th>Action</th>
<th>Cost</th>
<th>Characteristic</th>
<th>Artifact</th>
<th>Asset</th>
<th>Input/Output</th>
<th>Link</th>
<th>Statement</th>
<th>Issue</th>
<th>Risk</th>
<th>Time</th>
<th>Location</th>
<th>CORE Equivalent</th>
<th>DoDAF Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>decomposed by</td>
<td>incurs by</td>
<td>specified by</td>
<td>documented by</td>
<td>performed by</td>
<td>utilization</td>
<td>inputs outputs</td>
<td>based on</td>
<td>generates</td>
<td>resolves</td>
<td>occurs</td>
<td>located at</td>
<td>Function</td>
<td>Operational Activity/System Function</td>
</tr>
<tr>
<td>incurred by</td>
<td>decomposed by</td>
<td>specified by</td>
<td>documented by</td>
<td>incurred by</td>
<td>performed by</td>
<td>time, location</td>
<td>based on</td>
<td>generates</td>
<td>incurred by</td>
<td>occurs</td>
<td>located at</td>
<td>New</td>
<td>N/A</td>
</tr>
<tr>
<td>specifies</td>
<td>specifies</td>
<td>decomposed by</td>
<td>documented by</td>
<td>specified by</td>
<td>specified by</td>
<td>specifies</td>
<td>based on</td>
<td>generates</td>
<td>causes</td>
<td>occurs</td>
<td>located at</td>
<td>New</td>
<td>N/A</td>
</tr>
<tr>
<td>documents</td>
<td>documents</td>
<td>decomposed by</td>
<td>documented by</td>
<td>documents</td>
<td>documents</td>
<td>source of</td>
<td>based on</td>
<td>generates</td>
<td>causes</td>
<td>occurs</td>
<td>located at</td>
<td>Document</td>
<td>N/A</td>
</tr>
<tr>
<td>performs</td>
<td>incurs by</td>
<td>documented by</td>
<td>decomposed by</td>
<td>connected by</td>
<td>based on</td>
<td>generates</td>
<td>causes</td>
<td>occurs</td>
<td>located at</td>
<td>Component</td>
<td>System Node</td>
<td>Component</td>
<td>Operational Node/Operational Node</td>
</tr>
<tr>
<td>input to</td>
<td>input to</td>
<td>output from</td>
<td>triggered by</td>
<td>transferred by</td>
<td>decomposed by</td>
<td>generates</td>
<td>causes</td>
<td>occurs</td>
<td>located at</td>
<td>Item</td>
<td>Item</td>
<td>Operational Information/Data</td>
<td>Operational Information/Data</td>
</tr>
<tr>
<td>specified by</td>
<td>documented by</td>
<td>decomposed by</td>
<td>basis of</td>
<td>basis of</td>
<td>basis of</td>
<td>decomposed by</td>
<td>generates</td>
<td>causes</td>
<td>occurs</td>
<td>located at</td>
<td>Requirement</td>
<td>Requirement</td>
<td>N/A</td>
</tr>
<tr>
<td>generated by</td>
<td>generated by</td>
<td>generated by</td>
<td>generated by</td>
<td>generated by</td>
<td>generated by</td>
<td>generated by</td>
<td>decomposed by</td>
<td>causes</td>
<td>occurs</td>
<td>located at</td>
<td>Issue</td>
<td>Issue</td>
<td>N/A</td>
</tr>
<tr>
<td>caused by</td>
<td>resolved by</td>
<td>documented by</td>
<td>basis of</td>
<td>basis of</td>
<td>basis of</td>
<td>decomposed by</td>
<td>generates</td>
<td>causes</td>
<td>occurs</td>
<td>located at</td>
<td>Risk</td>
<td>Risk</td>
<td>N/A</td>
</tr>
<tr>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>decomposed by</td>
<td>located at</td>
<td>New</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>decomposed by</td>
<td>New</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reduced number of elements from 21* to 12, while adding time, location, and cost

*CORE’s DoDAF schema
A key attribute – *type*

• We added a “type” attribute to all classes
• Each “type” attribute contains different designators for the parent class

• Examples:
  • Assets can have types that include:
    • Operational Node, System, Component, Resource, Subsystem, System of Systems, Component, …
  • Actions can have types that include:
    • Operational Activity, System Function, Task, Mission, …
  • You can expand these lists to characterize anything in that class
  • When we display the element, we use the type
Benefits of the KBAD Schema

• Reducing the number of primary data elements means less complexity for analysts to deal with
  • Less complexity enables quicker capture and presentation of the information for analysis and decision making

• Covers programmatic, as well as technical, elements of information
  • Enables the trade off between cost, schedule and performance necessary for good design and decision making

• Eliminates overlap between similar data elements
  • Reduces potential for duplication of information which cuts the time and cost of data gathering

The result is a more cost-effective means for describing an architecture or system design.
• Typical data/activity modeling only works in the data dimension (e.g. IDEF0 or Data Flow Diagrams)
• For simple systems with sequential flow, this is sufficient
• However, for more complex systems, which all architecture are, it can be very misleading
• We need to be able to predict how system will behave
Why is sequencing important?

• In software the mantra is: data, data, data
  • Why? Because a tremendous amount of software programming has to do with input/output, hence the need to understand the data very well
  • The functional sequencing for individual software modules is relatively simple and many algorithms exist for complex methods (e.g., sorting algorithms)

• In architecture development (or system engineering or business process modeling …) sequencing is actually more important than the data
  • We want to know how the data affects the functional sequencing – we call these triggers
  • We want to control the behavior to avoid having significant failures
  • We also need sequencing for the human side

Hence the real answer is we need both if we are to develop systems and services with predictable behavior.
MBSE provide a robust set of constructs

- **SERIAL**: Action A → Action B
- **ITERATIVE**: Iteration Set → Action A
- **MULTIPLE-EXIT**: Exit 1 → Action B → Exit 2 → Action C
- **PARALLEL**: Action A → Action B
- **SELECTIVE**: Action A / Action B / Action A
- **LOOP**: Exit 1 → Action A / Exit 2 → Action A
- **REPLICATE**: Domain Set with coordination → Action B
One diagram gives many products

EFFBD: complete and executable

FFBD: lacks data

IDEF0: lacks constructs

N2 Chart: lacks constructs

Hierarchy: only parent to child

OV-6c; SV-10c

OV-5; SV-4
MBSE also diagrams the physical elements

Physical Hierarchy

- MCVS Subsystem
  - Subsystem
  - Roadway Subsystem
    - Subsystem
  - Vehicle
    - System
  - Maintenance and Construction Vehicle
    - System

Physical Block Diagram

- 13: Roadway Subsystem
  - Subsystem
    - RS-MCVS Interface
  - OV-2; SV-1; SV-2
- 83: Maintenance and Construction Vehicle
  - System
    - VS-MCVS Interface
  - 20: Vehicle
    - System

<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friday, February 08, 2008</td>
<td>Administrator</td>
<td>MCVS Subsystem</td>
</tr>
<tr>
<td>Number: 0x0</td>
<td>Name: MCVS Subsystem</td>
<td></td>
</tr>
</tbody>
</table>

© 2008 Systems and Proposal Engineering Company. All Rights Reserved
Traceability is a key to success
Most Programs Require Tools in All These Domains, but …

… they do not interoperate well together.

SPEC’s KBAD methodology uses CORE and MD Workbench to provide the underlying tool interoperability.
Tools used: CORE

- CORE's system engineering tools maintain an integrated design repository that provides traceability between requirements, functional models and system design elements.
- CORE's database schema may be modified to customize the tool to support customer needs and facilitate tool integration.
-Executable diagrams
- Special schemas and reports
- Powerful scripting language for your own report generation

Version 5.1 released with updated schema and reports
Tools used: MD Workbench

Eclipse-based IDE for code generation and model transformation, devoted to implementing MDA/MDE strategies. It provides:

- code generation (via text template engine and optionally Java)
- model manipulation through dedicated languages
- (imperative rules, declarative ATL modules to support QVT transformations, Java)
- model and metamodel management, including UML support
- customizable model connectors (XMI 1.0 to 2.1, XML, Hibernate, COM, etc.)

http://www.mdworkbench.com

2.1 is a major release! New features of MDWorkbench 2.1 includes:

Microsoft Word® Document generation

Doc Templates enable you to easily generate Microsoft Word® documents (Word 2003 or 2007).

A doc template is edited in Word, using all the functions, formatting, and styles available, with the addition of MDWorkbench constructs to access model data. A doc template brings the power and ease of use of text templates into Microsoft Word®.

You can generate rich documents to describe models, to report metrics, to report QA validation results, etc…

A great way to move data between different tools.
Tools used: NetViz

• Personnel all over the world use netViz to graphically depict operational architectures and logistical scenarios

• With NetViz you can create the SV-1 and SV-2 diagrams, with its intuitive graphical workspace, drill down capability, and connectivity views

• You can use the data embedded in your netViz projects to create other critical elements of a comprehensive C4I documentation project, like OV-1s (Operational Concept Diagrams) and OV-3s (Information Exchange Matrices)

Version 7.1 released; Available in Client Server or Enterprise Web editions as well
SPEC processes – full lifecycle

- Current Operations and Maintenance
- Future Operations and Maintenance
- Demolition and Disposal

- Architecture Development
- Operational T&E and Transition
- Integration & Verification

- System Design
- Integration and Test
- Integration & Verification

- Hardware/Software Acquisition
- Program Management

- Design & Analysis
Design and analysis phase

Current Operations and Support

Future Operations and Support

Demolition and Disposal

Architecture Development

Operational T&E and Transition

Integration and Test

Hardware/Software Acquisition

Program Management

System Analysis and Control

Functional Analysis and Allocation

Synthesis

Requirements Analysis

Design & Analysis

System Analysis and Control
SPEC’s middle-out process

Requirements Analysis

Functional Analysis and Allocation

System Analysis and Control

Top Down

Best Use: "Classical SE"

Middle Out

Best Use: Architecture Development (To-Be)

Bottom Up

Best Use: Reverse Engineering (As-Is)

Adapted from EIA-632
Middle-out timeline with products

1. Capture and Analyze Related Documents
2. Identify Assumptions
5. Develop the Operational Context Diagram
6. Develop Operational Scenarios
7. Derive Functional Behavior
8. Derive System Elements
9. Allocate Functions to System Elements
4. Capture Constraints
3. Identify Existing/Planned Systems
10. Prepare Interface Diagrams
11. Define Resources, Error Detection & Recovery
12. Perform Dynamic Analysis
13. Develop Operational Demonstration Master Plan
14. Provide Options
15. Conduct Trade-off Analyses
16. Generate Operational and System Architecture Graphics, Briefings and Reports

Requirements Analysis
Functional Analysis
Synthesis
System Analysis and Control

The middle-out approach has been proven on a variety of projects.
People Considerations

• Large teams make organization and focus on a vision very difficult
• You need people with a wide variety of skills and personalities
  • Someone with vision
  • Someone who can perform the detailed system engineering
  • Someone who understands the domain
  • Someone familiar with the technique and tools
  • Someone who understands the process
• They need to be trained as a team – including the government personnel
How Do We Move from Drawing Pictures to Building a Knowledgebase?

• Apply a proven, model-based technique that results in executable diagrams
• Use a process that implements the technique
• Use industrial-strength system engineering tools
• Make sure the personnel who use the methodology have the proper knowledge, skills and abilities to implement the approach
Questions & Discussion