The DACS Software Development Tools & Technology Information Clearinghouse (SDTATIC):

www.SDTATIC.com

Robert Vienneau
Project Manager
DACS Technical Manager
rob.vienneau@itt.com
315.838.7118

Tom McGibbon, CSDP
DACS Director
tom.mcgibbon@itt.com
315.838.7094
Presentation Agenda

• Purpose
• What is the DACS?
• What is the SDTATIC Clearinghouse?
• SDTATIC Features
• Model Based Development Tools Example
• How You Can Help

• Conference Survey (Q. 1-3)
Purpose of This Presentation

- Make you aware of SDTATIC
- Get your Feedback on the Clearinghouse
- Getting you involved
- What Else is Needed?
The DACS technical area of focus is Software Technology and Software Engineering, in its broadest sense.

Central distribution hub for the latest software technology information sources.

Wide variety of Technical Services to support R&D, development, testing, validation, and transitioning of Software Engineering technology.

Administered by DTIC. Technically managed by AFRL

[www.TheDACS.com](http://www.TheDACS.com) or iac.dtic.mil/dacs
SDTATIC Clearinghouse

• SDTATIC provides DACS users, staff, Subject Matter Experts (SMEs) with a central and searchable source of information on software development tools and technology.

• At the clearinghouse, users will find a uniform description, characterization, and where available unbiased reviews of software development tools.

• These tools are categorized by a taxonomy

• Initial capability implemented
Software Development Tools

• A software development tool is an executable software product supporting developers during the software system life cycle.
  – A software development tool, as defined here, excludes defined manual techniques, procedures, and processes. It includes commercial as well as open and free tools.

• The focus of SDTATIC is on technology-oriented tools, as opposed to tools for managing and acquiring software.

• SDTATIC Strategy: Prototype with one tool category and expand to other categories
Sample Categories of Tools

- Architecture Tools
- Requirements
- Design
- Construction
- Testing
- Maintenance
- Open vs. Proprietary

- Embedded Development
- Model Driven Software Engineering
- Software Assurance

- These are tool attributes
SDTATIC Context

Taxonomy category weights, interests, tool classes, etc.

Searchers

Administrators

Oversight Personnel

Vendors

Tools DB

Search Results

Workshops, Conferences

Collaboration Technology

Reports, e.g. Gap Analyses, Tool Class SOARs

SDTATIC Gov’t Site

Gov’t Personnel, SPRUCE, SISPI, etc.

Analysts, SMEs

Tool Info, Experience

Searchers and Analyst Inputs

SDTATIC

Gov’t Personnel, SPRUCE, SISPI, etc.

Analysts, SMEs

Tool Info

Searchers

Vendors

Oversight Personnel

Administrators

Taxonomy category weights, interests, tool classes, etc.
Taxonomy Overview

• Available on SDTATIC web site
• Defined as three-level hierarchy. First level:
  – Life cycle process
  – Functionality
  – Host or running platform
  – Target platform
  – Input type or language
  – Output type or language
  – Availability
• Taxonomy entry includes definition. Maintained wiki-style
Taxonomy Development

- Synthesizes existing taxonomies
- Life cycle decomposition
  - Based on ISO/IEC 15288:2008(E)
- Functionality from:
  - SWEBOK, Chapter 10
  - INCOSE (for requirements functionality)
- Target Platform
  - Extends Software Development Tools Directory
  - Extensions include Web-based and Middleware
Profiles

- Profiles are associated with the SDTATIC taxonomy
  - Used to prioritize tool requirements
  - Assign an importance to an item in the taxonomy (not important, somewhat important, important, very important)

- Uses:
  - Define what is important to a user
  - Define what is important for a technology area (e.g., testing tools) or other grouping of tools
  - Identify stretch needs for gap analysis
Representing Tools in the Taxonomy

• Tools are evaluated against the taxonomy (not implemented, partially fulfilled, fulfilled)

• DACS will initially develop and maintain assessment
  – Inputs from users welcome
  – Inputs from SMEs welcome

• Side by Side Comparison

• Suggestions: Survey Q4
The SDTATIC Site
www.SDTATIC.com

SDTATIC Software Development Tools And Technology Information Clearinghouse

Welcome to the Software Development Tools And Technology Information Clearinghouse (SDTATIC). SDTATIC provides
DACS users, staff, Subject Matter Experts (SMEs) with a central and searchable source of information on software
development tools and technology. All the clearinghouse users will find a uniform description, characterization, and where
available unbiased reviews of software development tools. These tools are categorized by a taxonomy related to the SEI's
Technology Roadmap.

Interest Groups

- Model-Driven Software Engineering (MDSE), also known as Model-Driven Development (MDD) (Model-Driven Architecture (MDA) is the Object Management Group (OMG) implementation)


SDTATIC Actions
- Browse SDTATIC Taxonomy
- Review a Software Tool
- Search Tools
- Register as a Subject Matter Expert
- Suggest a Tool

Related Projects
- Systems and Software Productivity Collaboration and
User Capabilities for Finding Tools

- Browsing
- Searching
  - Near term: profile searching
  - Long term: natural language
    - “design tools that generate Java or C++”
- Ranking
  - Weighted rank order of tools based on profile priority
    - Similar to QFD Approach

Survey Q5

SDTATIC Actions
- Browse SDTATIC Taxonomy
- Review a Software Tool
- Search Tools
- Register as a Subject Matter Expert
- Suggest a Tool

SSTC
22 April 2009
Finding Technology Gaps

- **SDTATIC Gap Analysis Approach based on Quality Function Deployment (QFD)**
- Each column for each tool generates a weighted sum.
  - This weighted sum can be used to sort most relevant to least relevant tool
- Each row for each taxonomy category is summed.
  - Totals can be viewed as the extent to which the “market” addresses those features
  - Poorly scored features could be interpreted as “gaps”

- Survey Q6
Calling all SMEs

- Subject Matter Experts (SMEs) on Tool Technology Areas
- SMEs on Individual Tools
- DACS will work with SMEs for high quality assessments
  - Will contract with selected SMEs
- We will contact you with user questions
  - Provides you direct access to users

Survey: Q7

SDTATIC Actions
- [Browse SDTATIC Taxonomy](#)
- [Review a Software Tool](#)
- [Search Tools](#)
- [Register as a Subject Matter Expert](#)
- [Suggest a Tool](#)
Calling Software Development Tool Vendors

• SDTATIC will collaborate with tool vendors for high quality assessments
  – Tool vendor assessments will be shown separately
• We will either contact you or you can contact us.

• Survey: Q8 if you are a tool vendor

SDTATIC Actions
- Browse SDTATIC Taxonomy
- Review a Software Tool
- Search Tools
- Register as a Subject Matter Expert
- Suggest a Tool
Getting Your Input

- Capabilities exist to provide inputs/reviews on tools
- SDTATIC.com is a wiki
- SDTATIC Community Building
- Suggest Tools

- Survey: Q9

SDTATIC Actions
- [Browse SDTATIC Taxonomy](#)
- [Review a Software Tool](#)
- [Search Tools](#)
- [Register as a Subject Matter Expert](#)
- [Suggest a Tool](#)
SDTATIC Community Building

• Work with related projects, e.g.
  – DoD Best Practices Clearinghouse
  – International Council on Systems Engineering (INCOSE)
  – Software Assurance Metrics and Tools Evaluation (SAMATE)
  – Software Systems Stockroom (S3)
  – Systems and software Producibility Collaboration and Evaluation Environment (SPRUCE)

• Use collaborative technology (e.g., wiki)
• Surveys from DACS
• Sponsor workshops, conference tracks, etc.

• Survey: Q10
Other Services and Information From SDTATIC

- DACS/SDTATIC Team will Respond to Technical Inquiries on Software Development Tools, up to 4 hours, for Free
- Other Information
  - For Open Source, links to the source
  - Related documents
  - Conference links
  - Vendor links
Model-Driven Software Development

• Definition: Model-driven development is simply the notion that we can construct a model of a system that we then transform into the real thing... A model is a coherent set of formal elements describing something (for example, a system, bank, phone, or train) built for some purpose that is amenable to a particular form of analysis... Model-driven development automates the transformation of models from one form to another. (Mellor et al 2003)

• Synonyms:
  – Model-Driven Architecture (MDA)
  – Model-Driven Development (MDD)
  – Model-Based Development (MBD)
  – Model-Driven Software Engineering (MDSE)
MDD Process

- Identify Domain Abstractions
- Design and Implement Domain-Specific Modeling Language (DSML)
  - Design Patterns, Libraries, Etc.
- Implement model within DSML
  - Models, Product Lines
- Configure for target platform, application
  - Design Patterns, Libraries, Etc.
- Generate source
  - Applications

Languages, Metamodels
MDSE Raises Level of Abstraction

(Based on Kelly and Tolvanen 2008)
Origins

• Knowledge-Based Software Assistant (KBSA)
  – AFRL project
  – Project meetings became KBS Engineering (KBSE) conference
  – Now IEEE Conference on Automated Software Engineering

• Computer Aided Software Engineering (CASE) tools
  – Often Object-Oriented
  – Often with diagrams for user interaction
  – Functionality: Documentation, prototype simulation, code generation

• Object Management Group (OMG) and Unified Modeling Language (UML)
  – UML created by the “Three amigos”: Grady Booch, Ivar Jacobson, and James Rumbaugh
  – Model-Driven Architecture (MDA) is OMG project
Example MDD Tools

- AndroMDA – OMG MDA-compliant
- ArcStyle - OMG MDA-compliant
- Borland Together
- CA Gen
- CA Plex
- Generic Modeling Environment (GME)

- MetaEdit+
- Oslo
- Rational Software Architect
- Rational Software Modeler
- Telelogic Tau
- Telelogic Rhapsody
Metamodeling Hierarchy

M3: Metamodel (aka Metametamodel)
    defines

M2: Domain-Specific Language (aka Metamodel)
    defines

M1: Model
    describes

M0: Instance

(Based on Stahl and Volter 2006)
OMG Standards for MDSE

- Model Driven Architecture (MDA)
- MetaObject Facility (MOF)
- Unified Modeling Language (UML 2.0)
- Object Constraint Framework (OCF)
- Query/View/Transformation (QVT)
- XML Metadata Interchange (XMI)
- Common Warehouse Metamodel (CWM)
  Metadata Interchange Pattern (MIP)
Twelve UML Diagram Types in Three Categories

<table>
<thead>
<tr>
<th>System Structure</th>
<th>Model Management</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
<td><strong>Package</strong></td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td><strong>Subsystem</strong></td>
</tr>
<tr>
<td><strong>Component</strong></td>
<td><strong>Model</strong></td>
</tr>
<tr>
<td><strong>Deployment</strong></td>
<td></td>
</tr>
</tbody>
</table>

An innovation of UML 2.0
## Twelve UML Diagram Types in Three Categories (Cont’d)

<table>
<thead>
<tr>
<th>System Behavior</th>
<th>Use Case</th>
<th>Relationships and the flow of events between actors and a sequence of related transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sequence</td>
<td>Object interactions in a sequence</td>
</tr>
<tr>
<td></td>
<td>Activity</td>
<td>Flow of control (e.g., business workflow or between methods of a class)</td>
</tr>
<tr>
<td></td>
<td>Collaboration</td>
<td>Object interactions organized around objects and their links</td>
</tr>
<tr>
<td></td>
<td>State Chart</td>
<td>For a given class, states and events that cause a state transition</td>
</tr>
</tbody>
</table>

An *interaction diagram* is a combination of a sequence and a collaboration diagram.
MDD Input Languages Example

4.4.7 Spring
4.4.8 Struts
5.0 Input Type or Programming Language
5.1 Metamodeling Framework
5.1.1 MetaObject Facility (MOF)
5.2 Domain Specific Language
5.2.1 Unified Modeling Language
5.3 Interchange Format
5.3.1 XML Metadata Exchange
5.3.2 QueryView/Transformation
5.3.3 Object Constraint Language
5.3.4 Atlas Transformation Language (ATL)
5.4 Programming Languages
6.0 Output Type or Language
6.1 Metamodeling Framework
6.1.1 MetaObject Facility (MOF)
Further Information

Other Suggestions: Q11

SDTATIC Web Site:
http://www.SDTATIC.com/

Robert Vienneau
Project Manager
DACS Technical Manager
rob.vienneau@itt.com
315.838.7118

“Just because it’s SDTATIC, doesn’t mean things don’t change”
Backup
Metamodel Hierarchy Example

M3 (MOF)
- Class
  - Attribute
    - Video
      +title: String
      title="2001: A Space Odyssey"
  - Instance

M2 (UML)
- Class
  - Attribute
    - Video
      +title: String
      title="2001: A Space Odyssey"
  - Instance

M1 (User Model)
- Video
  - +title: String
  - : Video
    title="2001: A Space Odyssey"

M0 (Run-time instances)
- aVideo
  (Based on UML 2.0 Infrastructure Specification 2003)
OMG Model Driven Architecture (MDA) Process

1. Build the Computational Independent Model (CIM)
2. Build the PIM
3. Transform the PIM into the PSM
4. Generate code from the PSM