

Leveraging UML as a Standard Notation for Enterprise Architecture

The Open Group IT Architecture Practitioners Conference
July 19th 2006

Dan Hughes, *Principal Consultant*, Systems Flow, Inc

Matt Daniels, *V.P., Enterprise Strategy & Architecture*, Citizens Bank Financial Group

Andre Alguero, *Principal Consultant*, Digitas, Inc.

James Hosey, *Senior Consultant*, Systems Flow, Inc.



Abstract

UML is taking root as the standard notation of choice for communicating software designs, yet higher-level architecture diagrams frequently consist of clouds, flaming walls, and a myriad of other icons, colored boxes, and lines. In this presentation, we will present the benefits of adopting a standard notation, specifically UML, for architecture diagrams. We will offer a proposed set of UML notations for specific architecture views and present some standards and guidelines as well as share experiences rolling this out to an icon-laden enterprise.

Presentation Overview

- Genesis of this work
- Why UML?
- Review of the UML Architecture Views
- Implementation Experiences
- Questions

Genesis of this work

- Launched a “boots on the ground” enterprise architecture process at Citizens Bank
- Developed a standard approach for describing architectures with a set of diagram templates at its core
- Identified pain points in architecture communication including a need to capture “as is” architectures and publish reference architectures

About the Enterprise

About Citizens Bank Financial Group

8th Largest Commercial Banking Company in the US

40

States

26,000

Employees

\$155 Billion

In Assets

1600

Branches

3800

ATM's

1700

Call Center Agents

The Citizens "Credo"

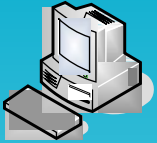
Customers. Treat the customer the way you would love to be treated all the time.

Colleagues. Do what it takes to make our company the best place in the world to work.

Community. Show that you care deeply about the community. Conduct yourself ethically at all times.

About the Citizens Bank Environment

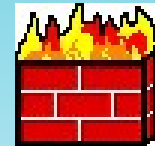
- Centralized Infrastructure Support
- Specialized Technology Groups
- Some Distributed Business Line IT
- Heterogeneous Computing Environment
 - Legacy Mainframe
 - Growing Distributed Activities
 - 25,000 Microsoft Desktops
 - 500 Unix Servers
 - 2100 Windows Servers
- Buy vs. Build with heavy customization (COTS)
- Introduced Enterprise middleware in 2002
- 200 Annual Technology Projects



Existing Diagram Landscape



- Lack of formal standards or even recommendations led to ad hoc diagrams with a limited level of consistency at the team level and no consistency at the enterprise level
- Diagram formats ranging from boxes and lines to flaming walls, miniature workstations, and an unlimited set of Visio icons
- Many diagrams included whatever information could fit, and then a bit more
- Multiple vendors came with multiple diagram formats



Rubrics for EA Diagrams

- Must clearly communicate architecture designs with consistent scope
- Must be tool agnostic
- Must maximize success rate for diagram producers
- Must be easily understood by diagram consumers
- Must be lightweight enough for a rigor-challenged enterprise, but heavyweight enough for success
- Must be whiteboard friendly

What is UML?

“The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system.

*Rumbaugh, Jacobson, Booch.
The Unified Modeling Language Reference Manual
2nd Edition.*

Why UML?

- Notation evolved out of 10+ years of refinement
- Industry standard diagramming notation.
Standards “come with”:
 - Books, articles, training, and online guidance
 - Tool adoption and templates
 - Pre-trained resources
 - Market value attached to the skillset

Why UML?

(continued)

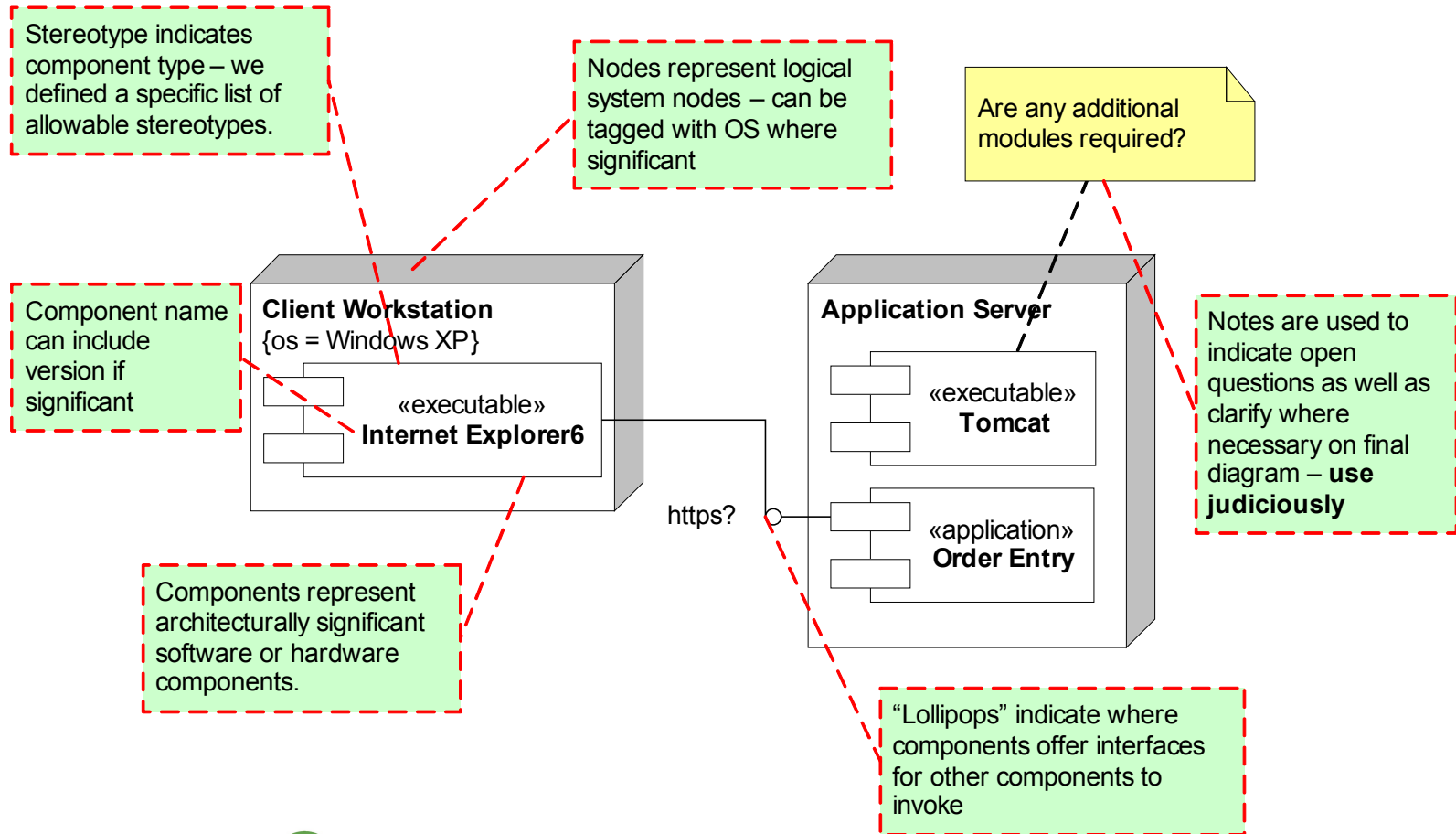
- Precise structure assists with consistency, completeness, and scope
- “Fit for purpose” diagrams force a separation of concerns
- Icon use is allowed, but not typically employed
 - Diagram creation time-saver
 - Consistent readability
- Core notation is simple and consistent – allows a focus on content vs. form

Architecture Views

- Architecture views have stabilized after about a year of iterative refinement
- Core views necessary to capture the right level of detail for most logical architecture designs are:
 - Conceptual Overview (non-UML “Powerpoint View”)
Very high level with internal standards around format
 - Logical System Overview
 - Data Context
 - Data Collaboration
- A physical deployment diagram instantiates the logical system overview

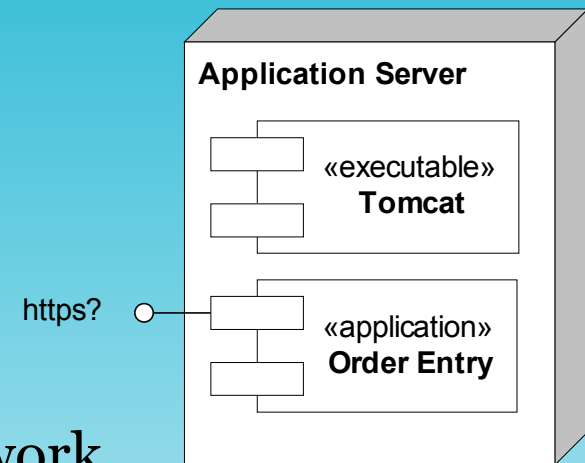
Logical System Overview

- ✓ Logical view of the architecturally significant system nodes , components, and relationships
- ✓ Created using UML Component Diagram Notation



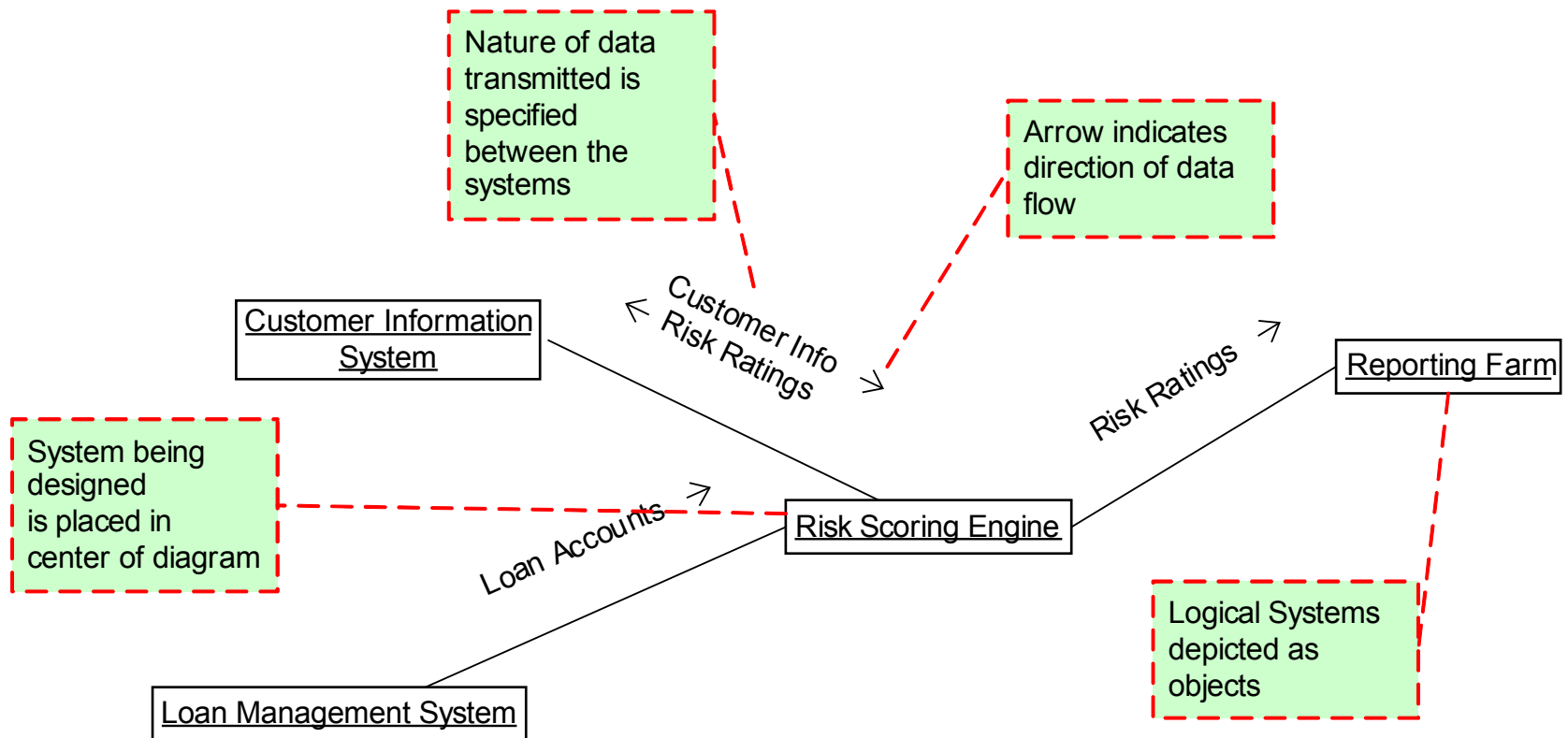
About the Logical System Overview

- Provides clear static view of systems involved and their relationships
- Logical nodes inventory hardware and software needs - physical nodes instantiate logical nodes
- Nodes and components identify support tasks and responsibilities
- Protocol identification useful for network and information security stakeholders
- Roadmap artifact for most stakeholders including development, infrastructure, and support teams
- Basis for many other stakeholder specific artifacts – queuing views, code deployment views, etc.
- Makes a great poster!



Data Context

- ✓ A high-level view of the external systems with which the system being designed interfaces and the nature of the data transferred.
- ✓ Created using a simple UML collaboration diagram notation



About the Data Context

- Inventories real-time interfaces and batch data feeds to/from other systems.
- Frequently identifies external stakeholders that need to be included from a development and support perspective
- Highlights external system recovery dependencies

Customer Information
System

Customer Info →
← Risk Ratings

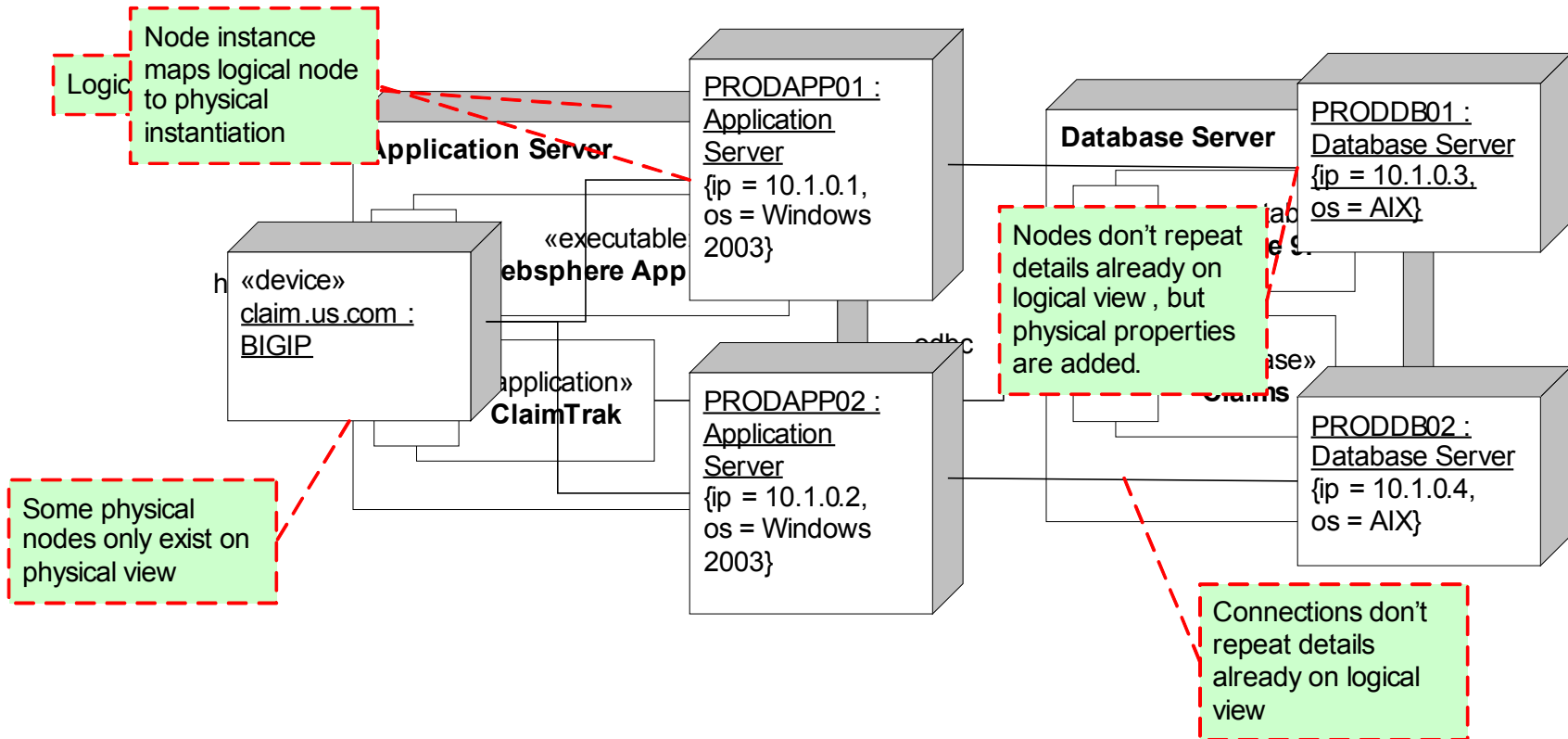
Risk Scoring Engine

Data Collaboration

- Similar in notation to the Data Context, but scoped to capture communications between the logical nodes that make up the system being designed
- Typically of specific interest to Information Security stakeholders, so data descriptions include classification details
- If data collaboration ordering is important, message sequence information is depicted

Physical Deployment

- ✓ A physical implementation diagram
- ✓ Created using UML deployment diagram notation which instantiates logical nodes as physical infrastructure



Other UML Diagrams Employed

- Some additional UML artifacts are used during the architecture design when additional clarity is required:
 - Use Case Diagrams for “backfilling” requirements
 - Sequence Diagrams for complex system integrations
 - Activity Diagrams for logic-driven system orchestrations and critical business processes

Diagramming Strategies

- Choose precision over completeness - only depict elements significant for the architecture under discussion
- Separate concerns onto targeted diagrams
- Treat clearly understood, stable systems as black boxes
- Limit typical use to basic elements of the notation
- Establish guidelines - Scott Ambler's "Elements of UML Style" is a good reference for this
- Promote important guidelines to standards as appropriate .
- Scope diagrams to fit on one page - some strategies include:
 - Identify some classification that lets you break the domain into multiple sub-domains (e.g. batch vs. real-time)
 - Manage scope and detail: Have more detailed diagrams with less scope roll up to a high-level diagram with a broader scope but less detail.

Implementation Recommendations

- Start with a small set of diagrams that employ a simple set of notation
- Provide specific guidance regarding which UML notations are in and out of scope
- Host formal and informal (e.g., “brown bag”) training, marketing the industry skills aspect
- Perform incidental training by reviewing notation as part of architecture reviews
- Resource for a level of mentoring services
- “Diagram First” to leverage the “tool” aspect of the notation
- Confirm a UML notation will not meet your need before crafting a non-standard diagram

Implementation Experiences

- From the outset, consumers quickly understood and embraced the notation
- Once introduced to diagrams in the UML notation, stakeholders would request the same views on other projects
- Non-EA resources producing diagrams were slow to let go of previously-employed techniques and move to the new notation
- Teaching the notation is less of a challenge than teaching the art and science of properly scoping diagrams
- Vendors need governance, but be aware of the learning curve

Questions?



Dan Hughes (dh@sysflow.com) is a principal consultant with Systems Flow, Inc., www.sysflow.com. Systems Flow helps organizations dramatically improve their competitive advantage through the practical, effective application of best practices in enterprise architecture and software development. He is engagement lead at Citizens Bank where he guided the launch of the enterprise architecture practice and is currently enterprise architect for Citizens Bank's Basel II implementation. Dan has 15 years of software engineering experience spanning a broad range of technologies and techniques. Startup to enterprise, he has launched, managed, and executed all aspects of both product and enterprise life cycle for clients in industries ranging from industrial automation to banking and insurance. He maintains a blog on software engineering at www.xengineering.com.

Matthew Daniels (matthew.daniels@citizensbank.com) developed, launched, and currently manages the Enterprise Strategy and Architecture department at Citizens Bank Financial Group. Prior to his six year career at Citizens, Matt previously held senior technology and leadership positions with Alltel Information Systems, Gartner, and eunetcom and has over 11 years of experience in the technology arena, specifically in the areas of Network and Data Security, large scale heterogeneous networking, technology integration and engineering, and application architecture.

André Alguero (aalguero@digitas.com) is a software architect with 15 years of experience building software solutions and consulting with enterprises to build and execute technology strategies. André has a passion for software engineering best practices with a focus on using models to build robust software solutions. André is a VP/Architect at Digitas, Inc. where he designs enterprise marketing solutions and technology strategies for Fortune 500 companies with a focus on content management. He previously was a Principal Consultant at Systems Flow, Inc.

James Hosey (jh@sysflow.com) is a senior consultant with Systems Flow, Inc. Over the course of his 16-year career, Jim has managed and executed all phases of the software life cycle and has delivered a wide variety of technology solutions for both commercial resale and internal use in domains that include banking, insurance, distribution, marketing, communications, and management training & development.