IBM

Agile Model Development with the IBM® Rational® Software Architect

Daniel Leroux
Distinguished Engineer, IBM
dleroux@ca.ibm.com

Maneesh Goyal
Senior Software Engineer, IBM
mgoyal@ca.ibm.com

Rational. software

MAC03

Go to IBM

Agenda

- Introduction
- Maneesh’ team
- Agile Modeling and principles
- Demo
- Questions

What is a Model?

- A model is an abstraction of a physical system
- Typically, you will create different models for a physical system to visualize different points of view
  - Users
  - Developers
  - Graphic Artists
  - Database developers
  - Testers
  - Documenters
  - And on and on ……

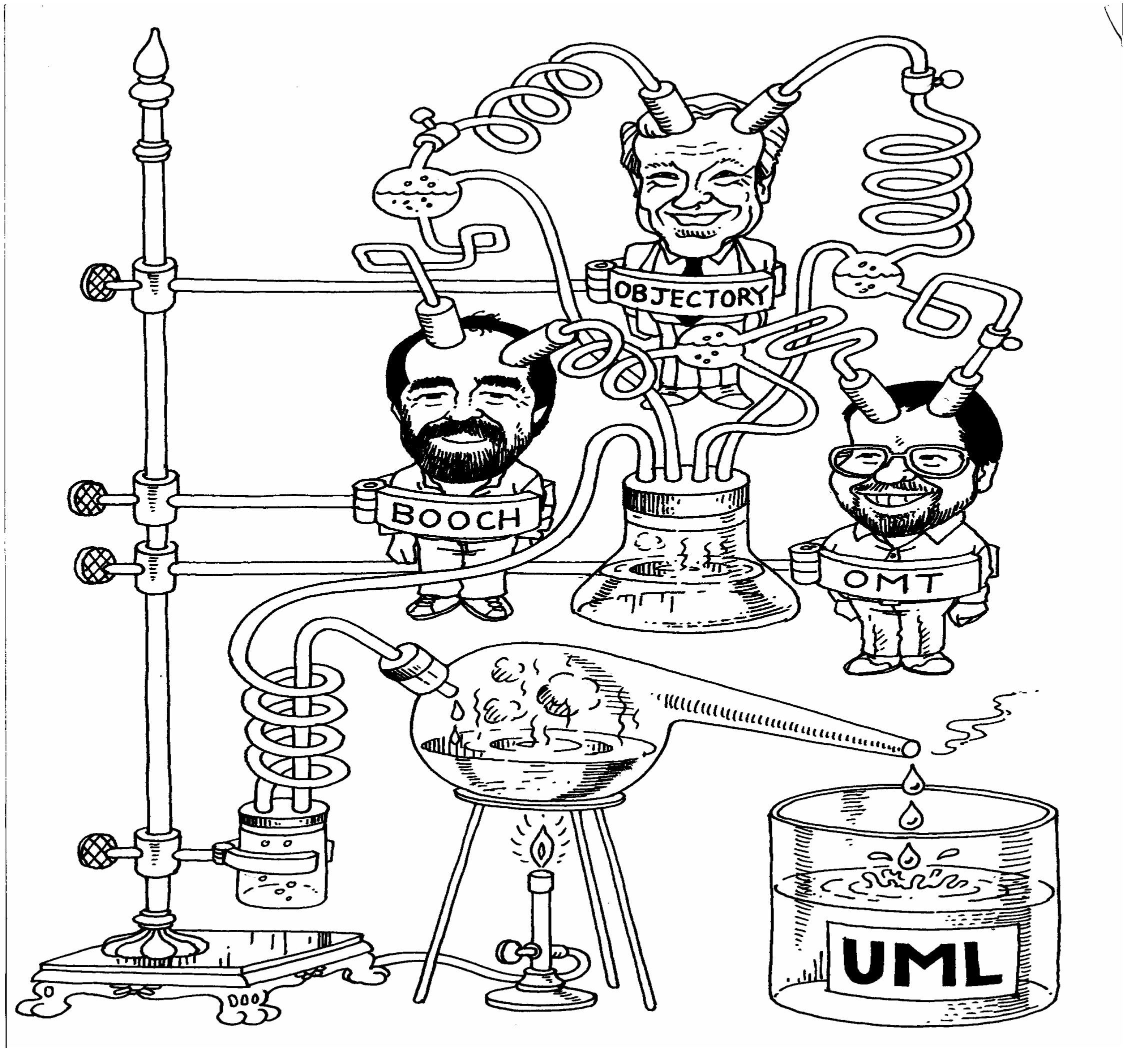
Why do we model?

- To manage complexity
- To detect errors and omissions early in the lifecycle
- To communicate with stakeholders
- To understand requirements
- To drive implementation
- To understand the impact of change
- To ensure that resources are deployed efficiently

Why do we model?

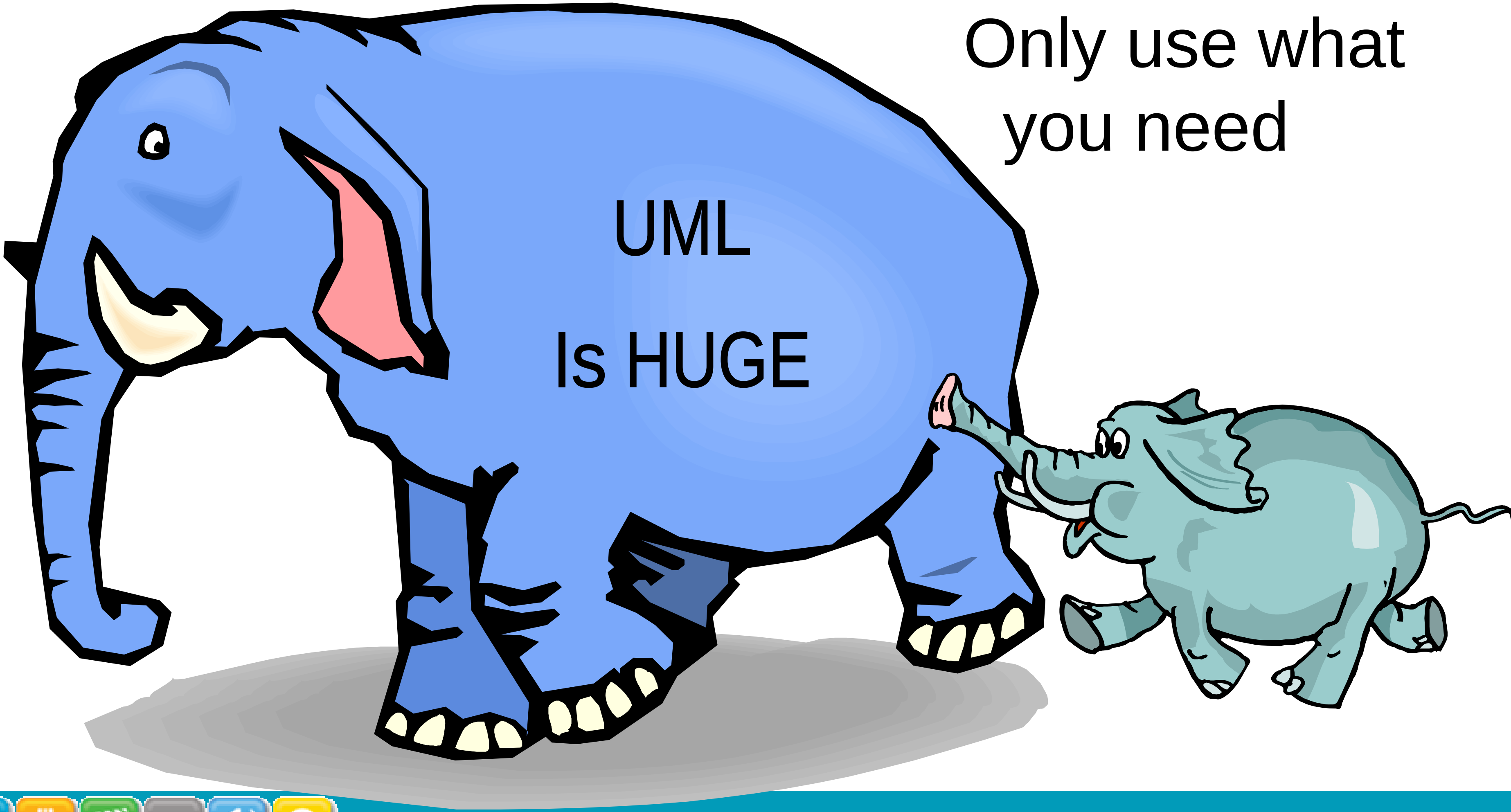
- To manage complexity
- To detect errors and omissions early in the lifecycle
- To communicate with stakeholders
- To understand requirements
- To drive implementation
- To understand the impact of change
- To ensure that resources are deployed efficiently

The Unified Modeling Language



- The UML is the standard language for visualizing, specifying, constructing, and documenting software and systems

TQ's Golden Rule

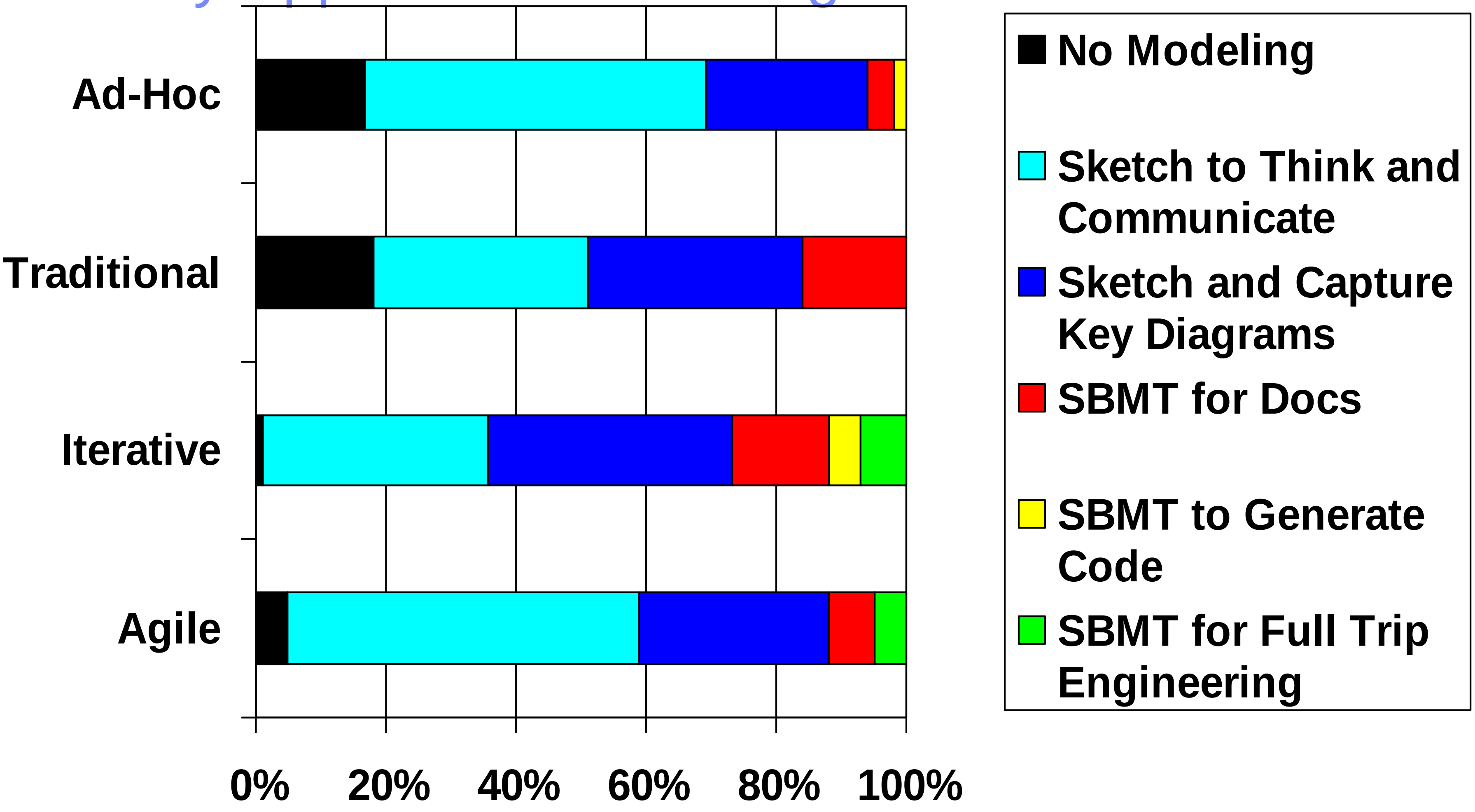


Only use what you need

Agile Modeling

- Agile Modeling is a collection of values, principles and practices for modeling software that can be applied on a software development project in an effective and light-weight manner

Primary Approach to Modeling



Common Misconceptions About CASE Tools

- Agile modelers don’t use CASE tools
  - Agile modelers use the simplest tool, and if the simplest tool for the job is a CASE tool, then that is what will be used
- UML requires CASE tools
  - Not true. UML drawings are often done by hand
- You start modeling with CASE tools
  - Typically modeling is started with a simple tool (e.g. flip charts) and then you migrate to a CASE tool if needed (e.g. to create persistent models)
- The CASE tool is the master
  - Not true. Once code is either generated from the model or written by hand, the code is the master. One tough decision that has to be made is should the model be updated to reflect the code?

Maneesh’ Team

- Maneesh: software development manager responsible for managing the architecture of a Rational Modeling product and leading a team of engineers developing this product
- Maneesh’ team
  - 10 brilliant software engineers
- Maneesh’ extended team
  - Product Manager
  - Release engineering team
  - Product delivery team
  - User Experience team

Team Process – Agile Modeling

- Review requirements
  - New requirements arrive from the product manager
  - Ask clarifying questions to the product manager and provide feedback
  - Review the updated requirements
- Manage SW architecture
  - Create new SW design models
    - Use case diagrams
    - Class diagrams, sequence diagrams, …
  - Review the SW designs created by the team members
  - Share and collaborate within the team and extended team

The Problem

- Information overload
  - Email discussions
  - Meeting minutes, whiteboards
- Collaboration and communication
  - Globally distributed teams
  - Knowledge transfer – The whys of architecture

Agile Modeling Principles

- Requirements envisioning
- Prioritized requirements
- Architecture envisioning
- Multiple models
- Just barely good enough models
- Model storming
- Active stakeholder participation

Requirements envisioning

- Create new requirements
- Modify existing requirements
- Use Rational Requirements Composer
- Stakeholder participation
  - Product managers
  - User experience team

Prioritized requirements

- Define criteria for prioritization
- Review and prioritize requirements
- Stakeholder participation
  - Internal stakeholders – software development engineers, user experience team
  - External stakeholders – product managers

Architecture envisioning

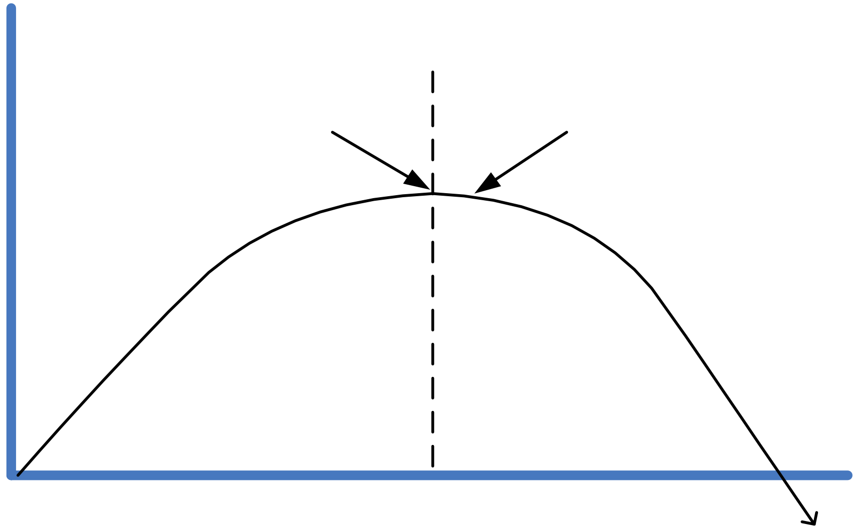
- Create and modify architecture models
- Associate the models with the requirements
- Stakeholder participation
  - Software architects
  - Software development engineers

Multiple models & Just barely good enough models

- Similar to whiteboard models
  - Use case diagram
  - Class and Component diagram
  - Sequence diagram
  - Deployment diagram
- Stakeholder participation
  - Software architect
  - Software development engineers

What Are Agile Models?

- Agile models:
  - Fulfill their purpose
  - Are understandable
  - Are sufficiently accurate
  - Are sufficiently consistent
  - Are sufficiently detailed
  - Provide positive value
  - Are as simple as possible



Agile models are just barely enough!

Active stakeholder participation & Model storming

- Review the requirements and architecture with all stakeholders
- Remove the barriers to participation
  - The needs of the globally distributed team
  - Provide the right tools

DEMO

Conclusion

- You are engaged in Agile Modeling if:
  - Your customers/users are active participants
  - Changing requirements are welcomed and acted upon
    - You work on the highest priority requirements first
  - You take an iterative and incremental approach to modeling
  - Your primary focus is the development of software, not documentation or models themselves
  - You model as a team where everyone’s input is welcome
  - You actively try to keep things as simple as possible
  - You discard models as development progresses
  - Customers/business owners make business decisions; developers make technical decisions
  - The model’s content is recognized as being significantly more important than the format/representation of that content
  - How you test what you describe with your model(s) is a critical issue continually considered as you model

References and Recommended Reading

- www.agilealliance.com, www.agilemodeling.com, www.agiledata.org, www.ambysoft.com, www.databaserefactoring.com, www.enterpriseunifiedprocess.com
- Ambler, S.W. (2002). Agile Modeling: Effective Practices for XP and the UP. New York: John Wiley & Sons.
- Ambler, S.W. (2003). Agile Database Techniques. New York: John Wiley & Sons.
- Ambler, S.W. (2004). The Object Primer 3rd Edition: AMDD with UML 2. New York: Cambridge University Press.
- Ambler, S.W. and Sadalage, P.J. (2006). Refactoring Databases: Evolutionary Database Design. Reading, MA: Addison Wesley Longman, Inc.
- Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Reading, MA: Addison Wesley
- McGovern, J., Ambler, S.W., Stevens, M., Linn, J., Sharan, V., & Jo, E. (2003). The Practical Guide to Enterprise Architecture. Prentice Hall PTR.

Questions

Thank You