zeligsoft

Integrating the SBC Toolchain

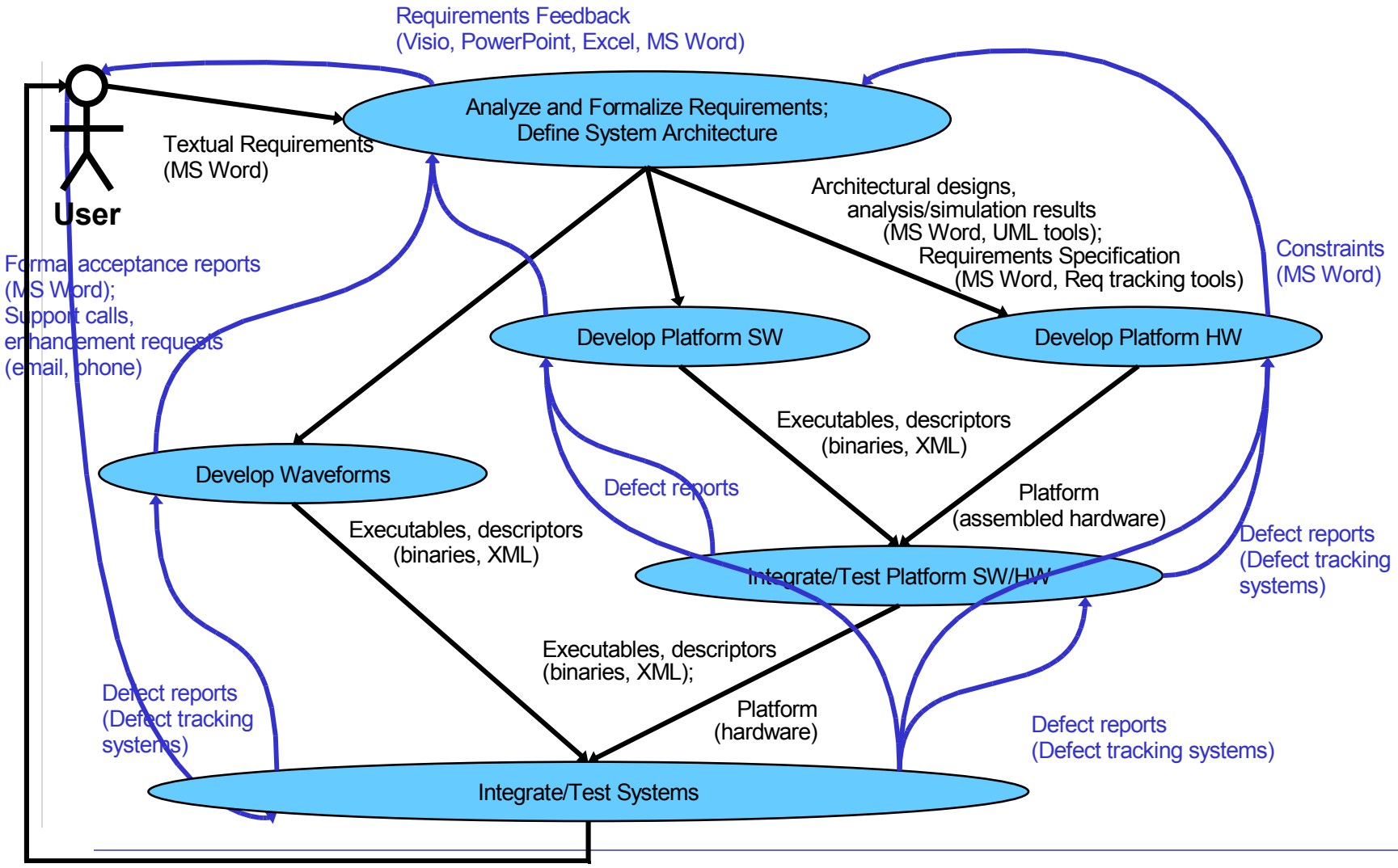
John Hogg
CTO
Zeligsoft

2007-03-06
V1.0

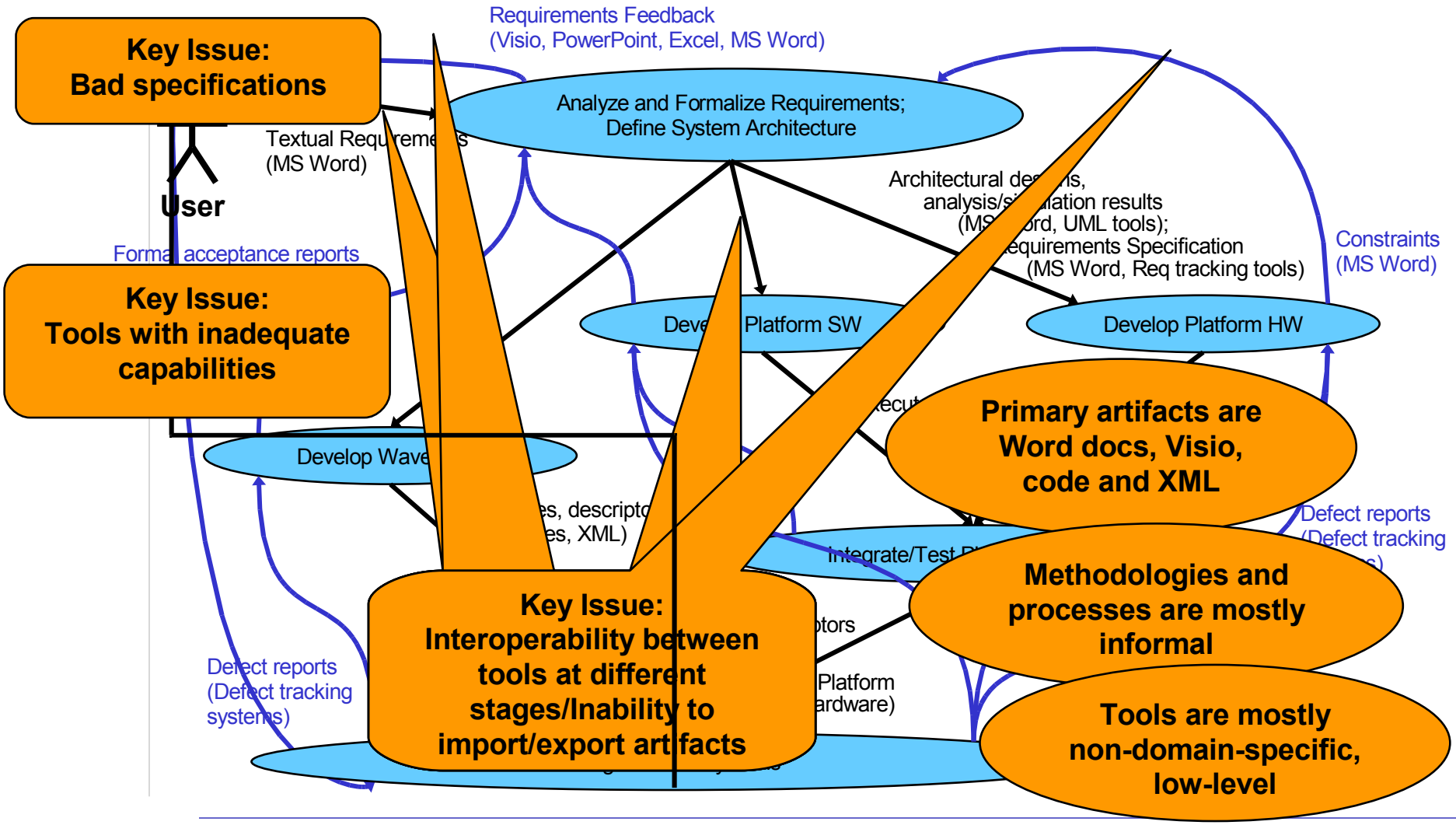
Agenda

- The problem
  - SDRF Design Processes and Tools Working Group Survey
- Integration strategies
- Summary

General System Design Process



General System Design Process



DPT-WG Survey Summary

- The most widely used tool suite in SBC technology development:
  - Microsoft Office

- Key Issues
  - Methodologies and processes are mostly informal
  - Lack of mature standards
  - **Development tools are generally non-domain specific**
  - **Development tools generally do not interoperate**
    - **Import/export of design artifacts between stages of development**
  - Tools do not easily support real time debugging and validation on systems supporting multiple parallel processing devices
  - Lack of a common language for communicating across interdisciplinary teams (SW, HW, HDL, System, etc)

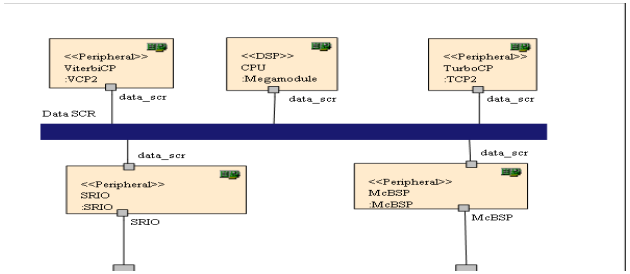
- These issues are mostly “worked around” with varying degrees of success

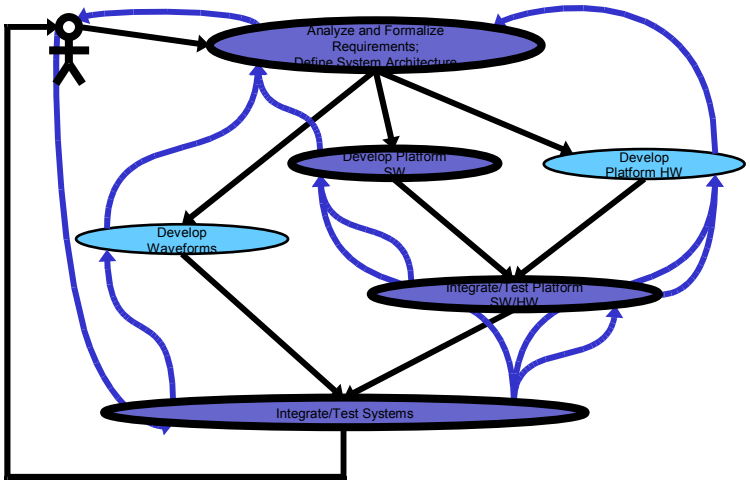
The Claim

- There is life beyond Word
- Tools are available for SBC tasks
- Integrations between these tasks exist
  - Through shared tools
  - Through tool APIs
  - Through artifacts
  - Through model transformation
- Existence proof

Integration by Shared Tools

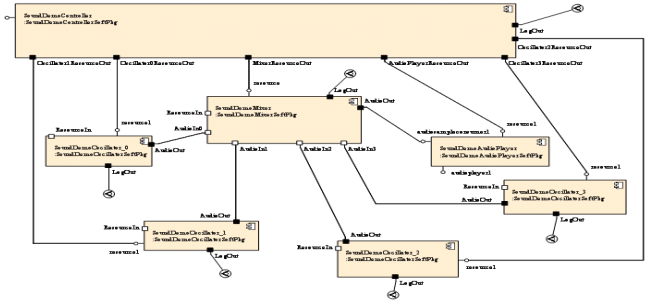
- Common artifacts of activities:
  - Platform

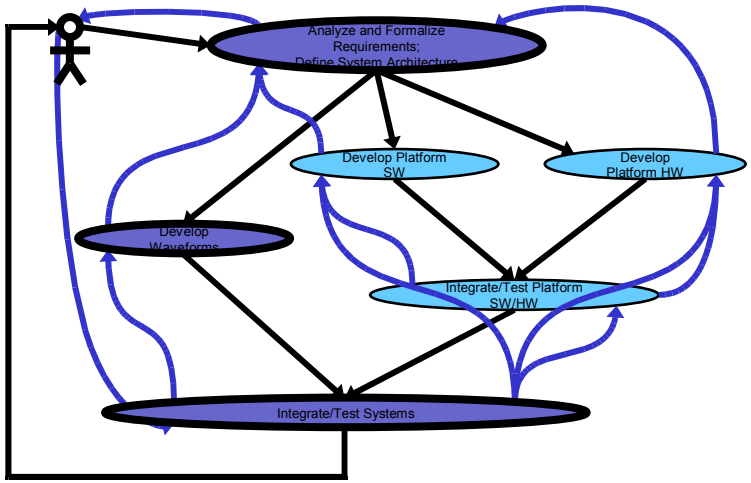




Integration by Shared Tools

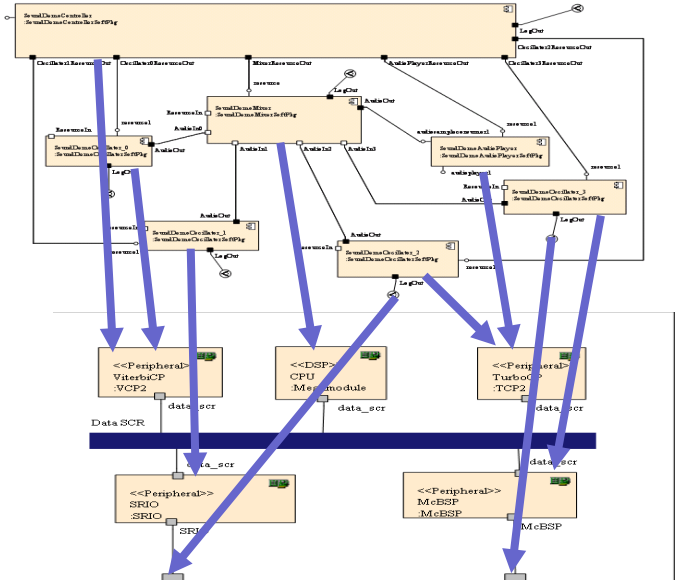
- Common artifacts of activities:
  - Platform
  - Waveforms



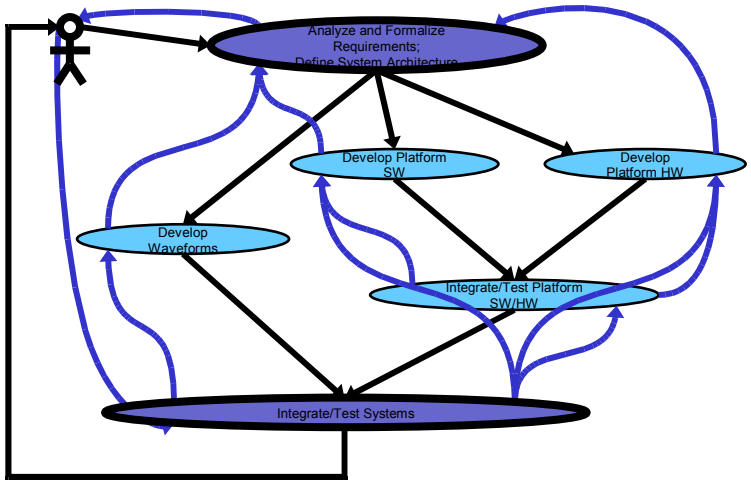


Integration by Shared Tools

- Common artifacts of activities:
  - Platform
  - Waveforms
  - Deployments



- Same tool can be used to model, verify and generate all artifacts in all tasks



Integration Through Tool API

- Simplest type of integration
- Classic example: Configuration Management tools
- Typical API is through command line
- Common in SBC tools today

Integration Through Artifacts

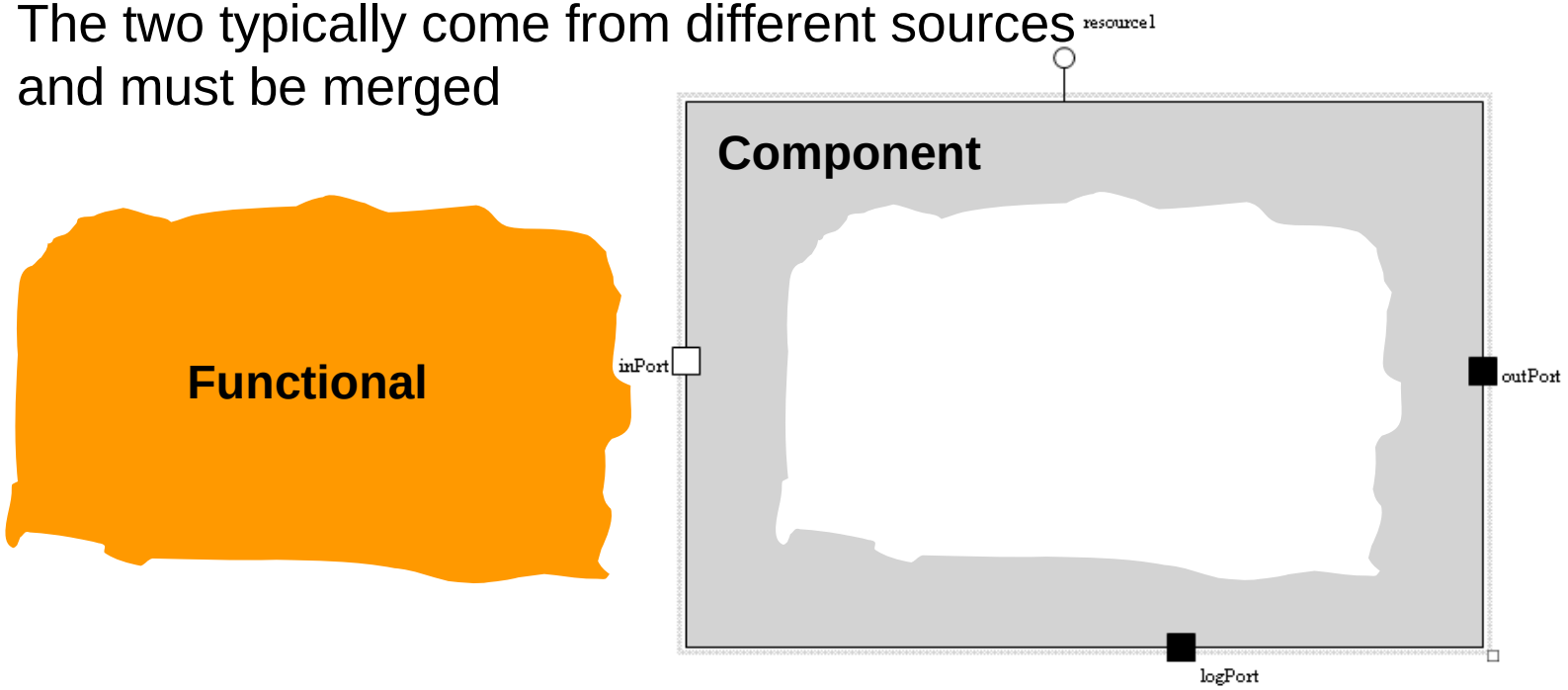
- Example artifacts:
  - Descriptor files
  - Code
- Iteration is the problem
  - Development is iterative and incremental
- Generated artifacts may be modified by subsequent steps
  - Manual changes
  - Generation from multiple sources
- Iterated integration must be lossless
  - Generation must never overwrite

Integration Through Model Transformation

- Define model in one tool
- Transform to model for use in another tool
- Example: component-based model and functional behavior model

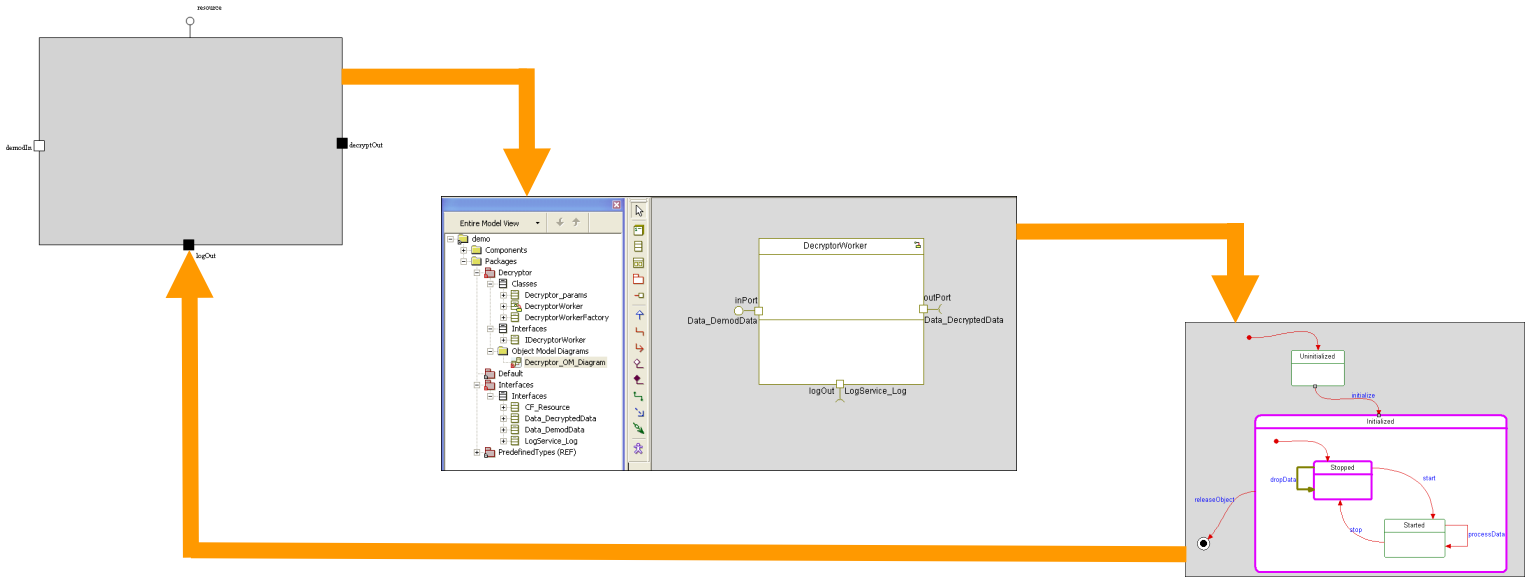
Component Architecture

- Component-based behavior: how the component interacts
- Functional behavior: what the component does
- The two typically come from different sources and must be merged



Integration Through Model Transformation

- Create component model
- Transform component model to functional model skeleton
- Add behavior (state, sequence diagrams)
- Generate, compile and run
- *Iterate*



Future Work: Integration Through Shared Models

- If two tools use the exact same model, sharing is simple
- No obvious SDR examples today
- One day this will come…

Summary

- SBC developers identify a lack of task integrations as a major problem
- Integrations between tasks exist
  - Through shared tools across different tasks
  - Through tool APIs
  - Through artifacts
  - Through model transformation
- *Get out of the Office more often*

zeligsoft

Thank You

www.zeligsoft.com +1 819 684 9639

hogg@zeligsoft.com