

Our Future – Autonomic Networking

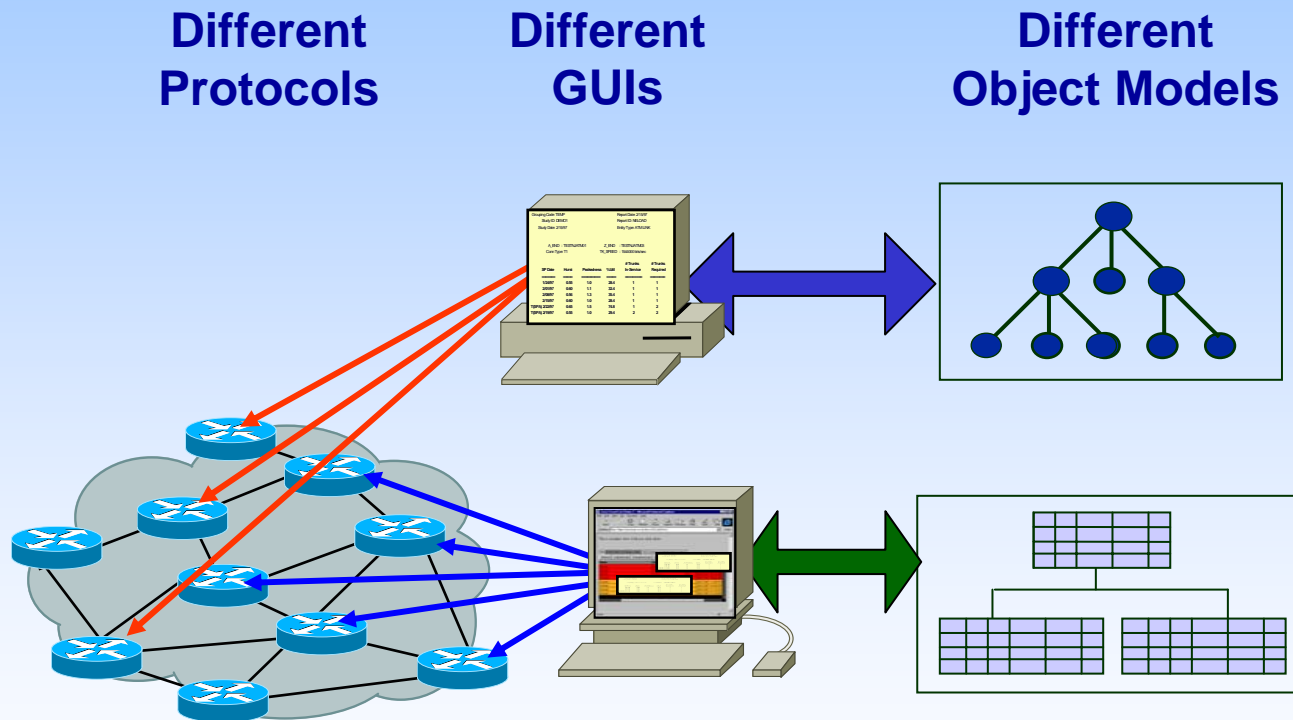
John Strassner
(john.strassner@intelliden.com)

Thank You!

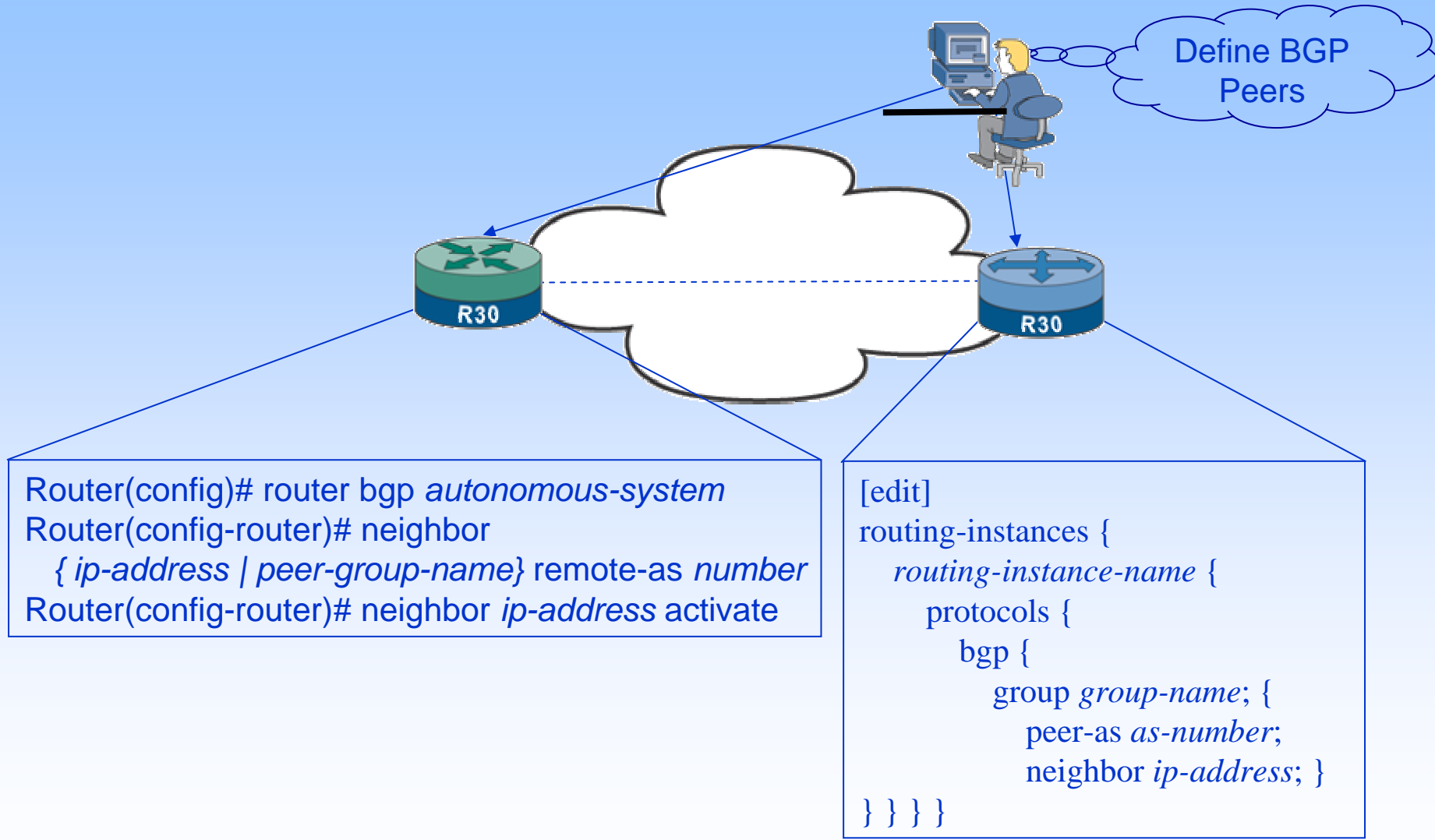
- Without the pioneering work of the AlbatrOSS & OPIUM projects, we would still be struggling!



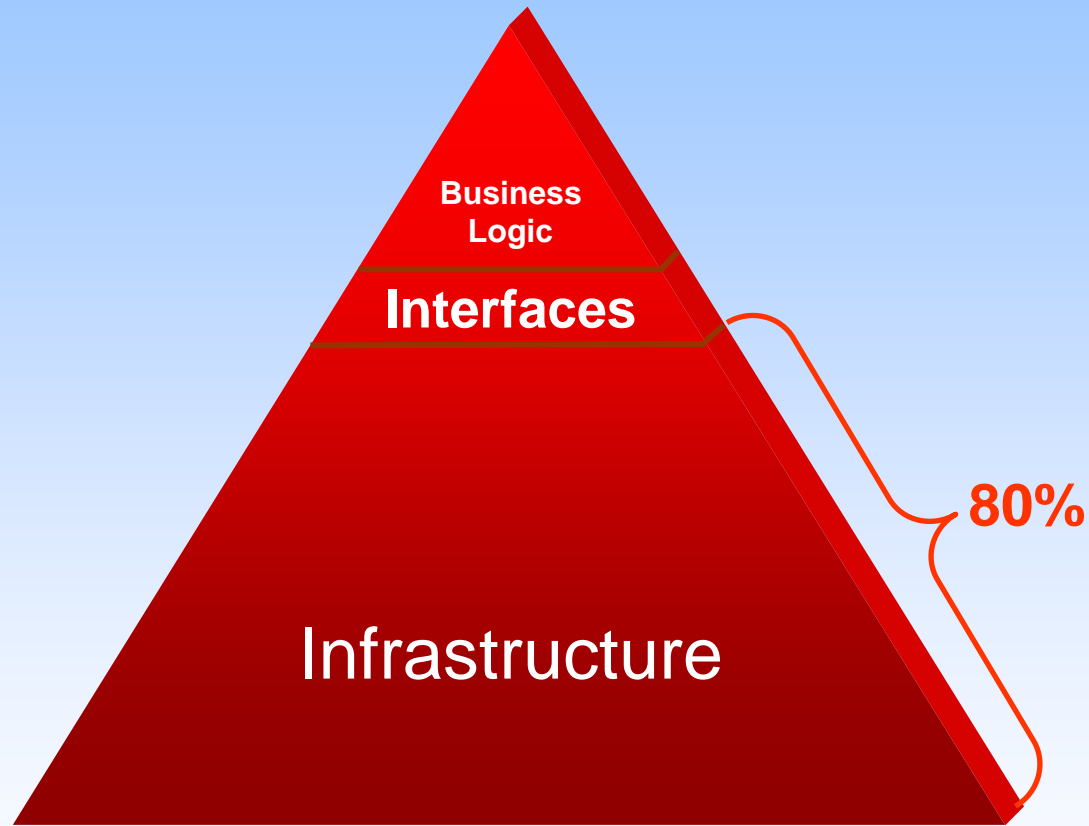
Integration Issues – How Is Data Shared?



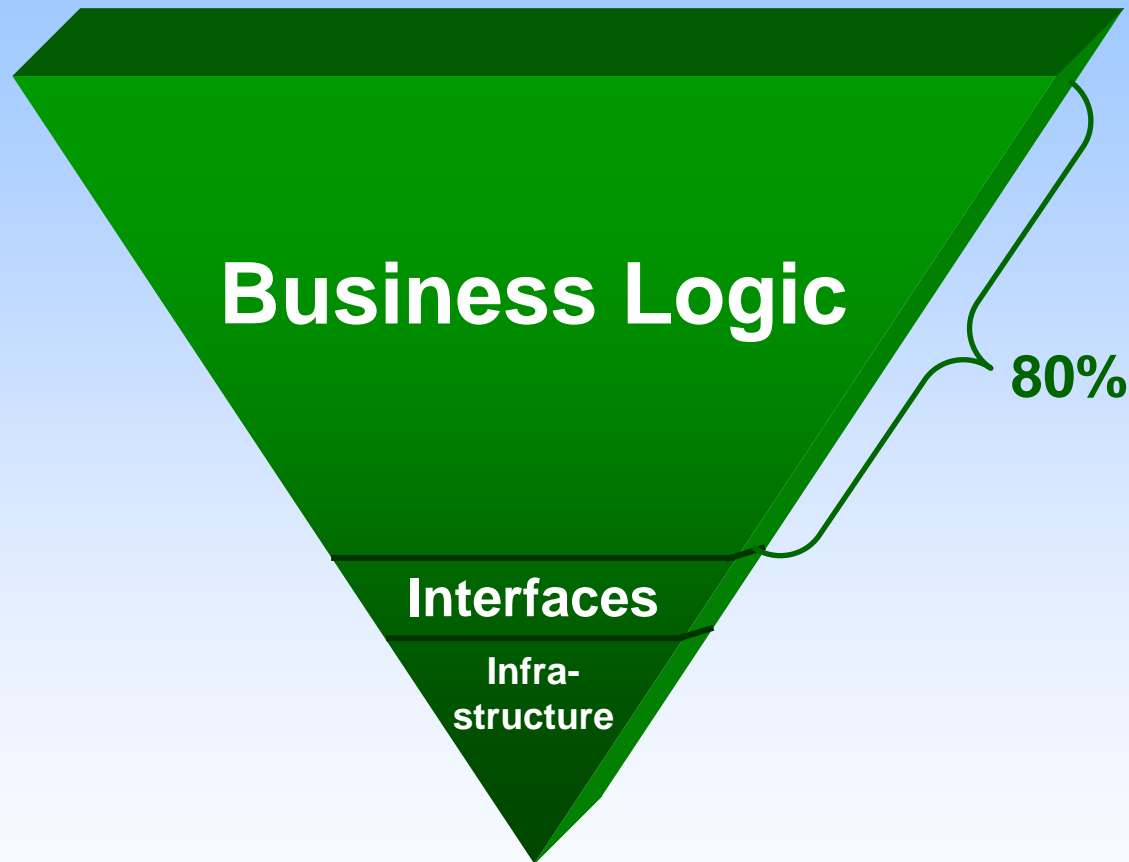
DEN-ng is the Network Lingua Franca



Changing the Focus From...



Changing the Focus To...



We Must Conquer Complexity

- The complexity of system design and management keeps increasing
 - Stovepipe systems offer best-of-breed functionality, but present an integration nightmare
- The complexity of business is also increasing
 - People are demanding a pervasive presence
 - Many types of businesses LOSE MONEY if they can't react fast enough to changing demands
 - The varieties of threats, problems, and non-optimized behavior keeps increasing
 - ***We need self-management***

We Must Continue to *Innovate*

- Web Services are a Good Thing...
 - A standard set of extensible protocols for discovery, definition and message passing
- ...And NOT Such a Good Thing...
 - Web Services is a Band-Aid
 - Encourages packaging of pre-defined functionality
 - Non-functionals and advanced negotiation models are not supported
- *Autonomic Computing requires services to be built dynamically from simpler building blocks*

What IS Autonomic Computing?

- Autonomic nervous system controls involuntary functions
 - Frees our brain from worrying about low-level (but *vital*) functions
- Autonomic computing similarly controls vital resources and services with little or no human intervention
 - Really all about self-government, building new computing paradigms that are inspired from biological behavior

Just Add Water and Stir

- The integration of self-configuration, -protection, -healing, and -optimization is critical
 - If that happens, these separate concepts will merge into more powerful concepts
 - For example, self-maintenance is the holistic combination of all four of these principles
 - Using anti-virus software as an example, the system will pro-actively try and upgrade its functionality
 - Adjustment of workload in response to changing conditions (e.g., component failures) and environment (e.g., new users running new apps)
- Self-management of a system is more than the sum of the self-management of its individual components

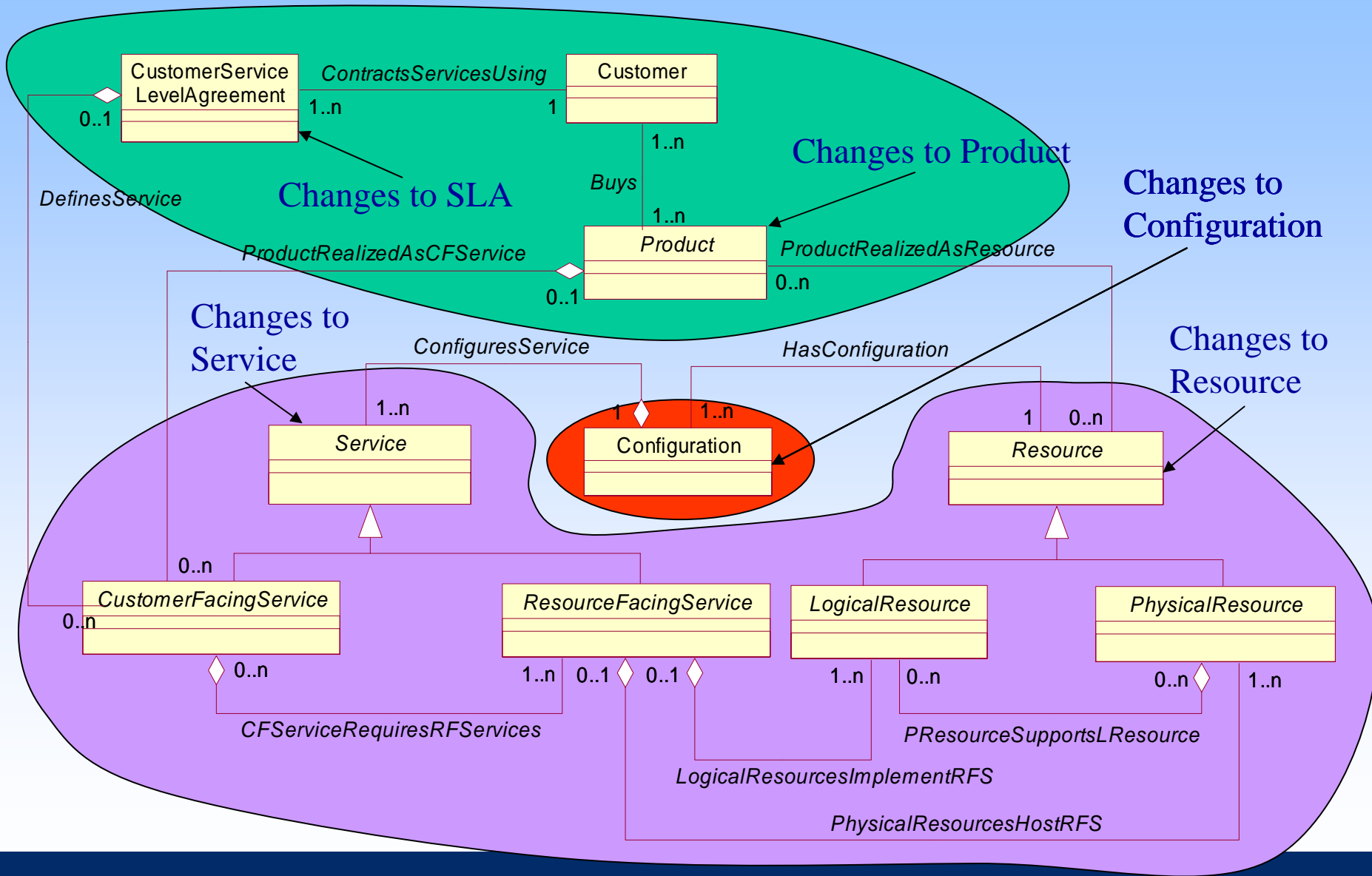
Specific Focus on Autonomic Networks

*“How can an environment be
autonomic when the network
is still manually configured?”*

Unique Design Points of DEN-ng

- DEN-ng views the world in terms of
 - **Capabilities** – normalized functionality
 - **Constraints** – restrictions on what you can use
 - **Context** – describes the environment in which different objects operate
 - **Profiles** – describes starting requirements and initialisation of an entity
- DEN-ng is built using **patterns** and **roles**
- DEN-ng uses a **finite state machine**
 - Other models represent the current state of an object
 - DEN-ng builds different models of different states, and binds them together using a finite state machine

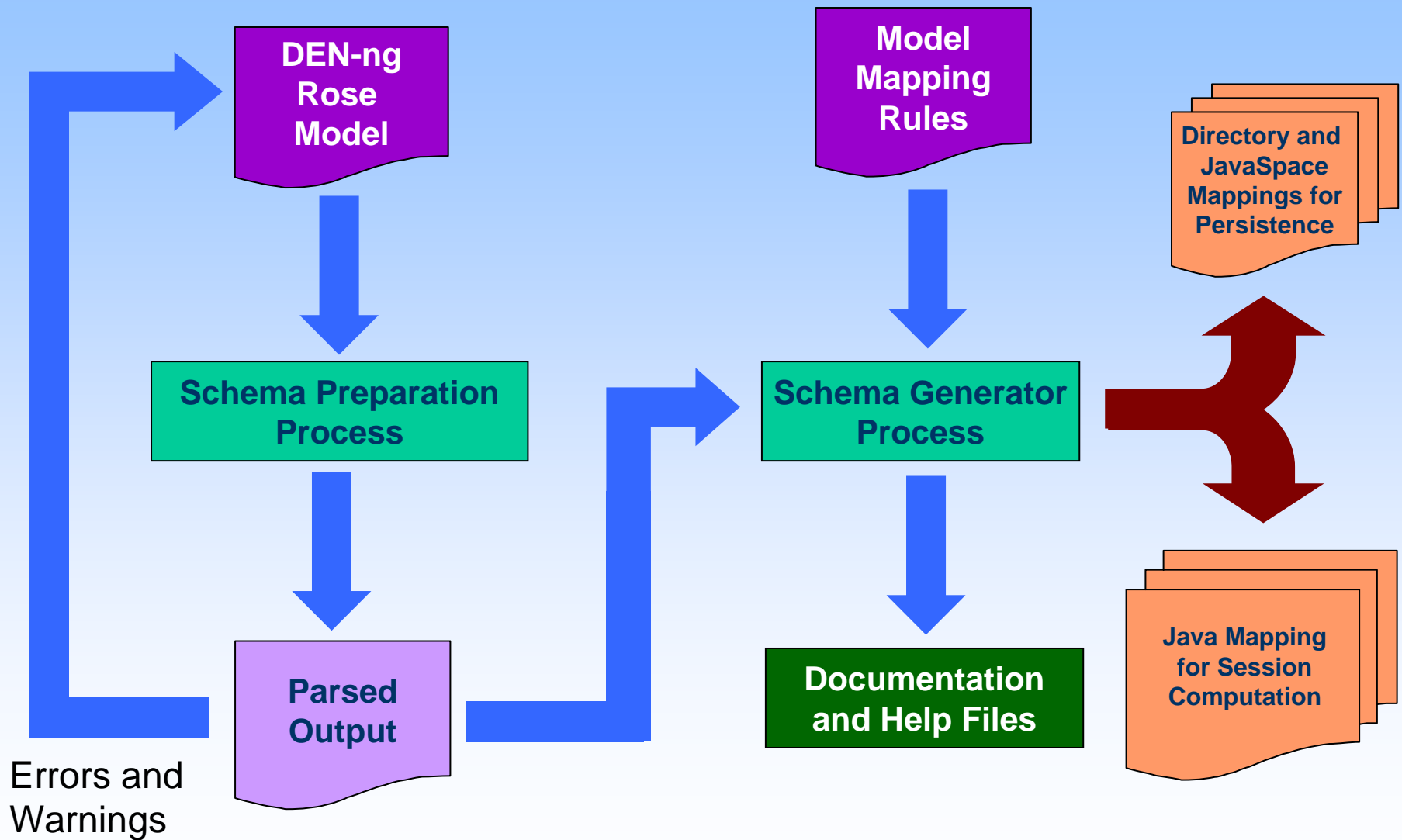
Business Driven Network Services



So Why Not Use MDA As Is?

- MDA has a LOT of complexity
 - Overkill for our needs
 - Lack of patterns, hard to implement
- COTS code generators were unable to handle the complexity of our needs
 - Existing code generators are full of bugs
 - Do not use patterns, no QA facilities
- Wanted to decouple the logic (in Java) from underlying persistence mechanisms

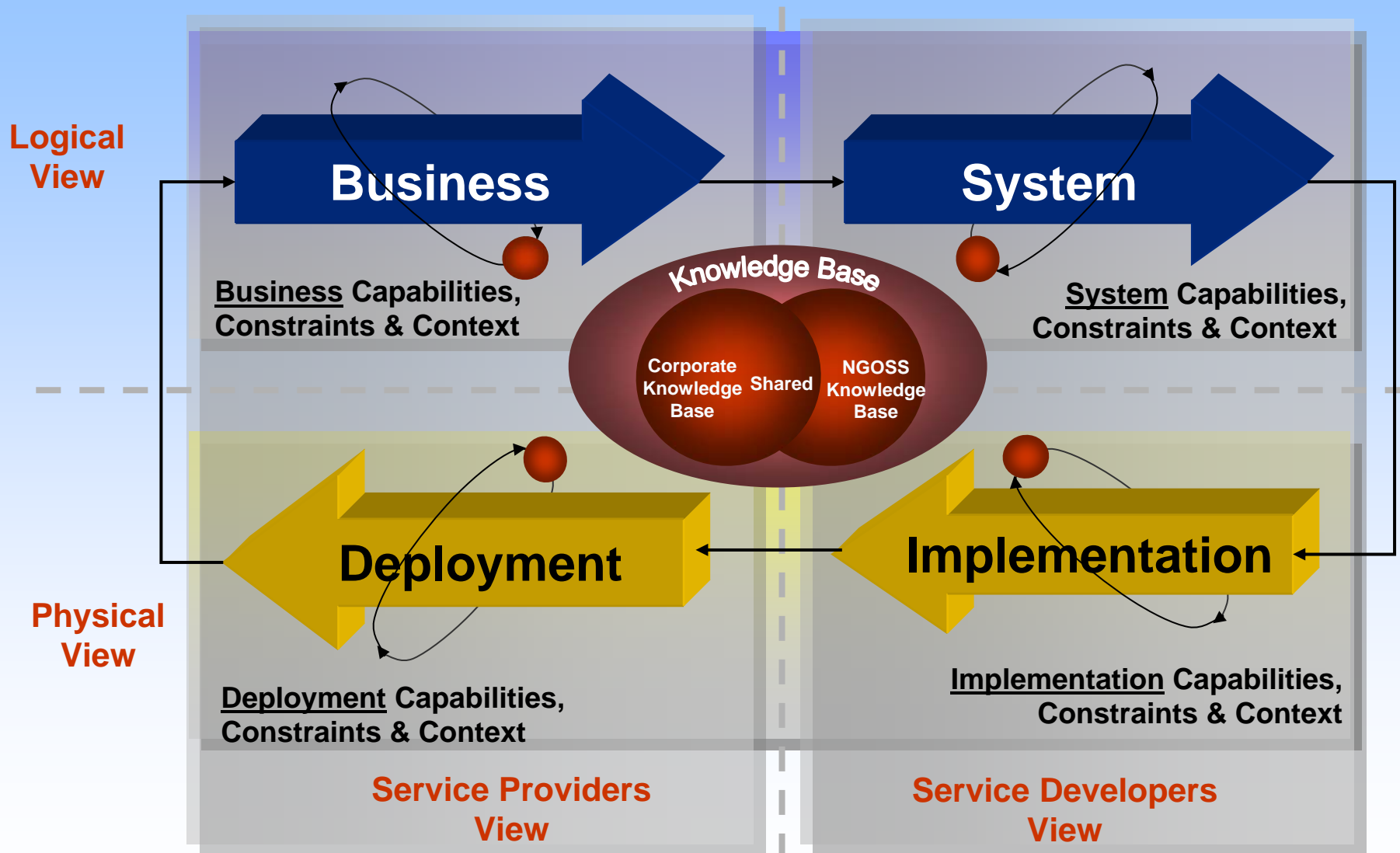
DEN-ng Model Driven Code Generation



Code Generation On Steroids

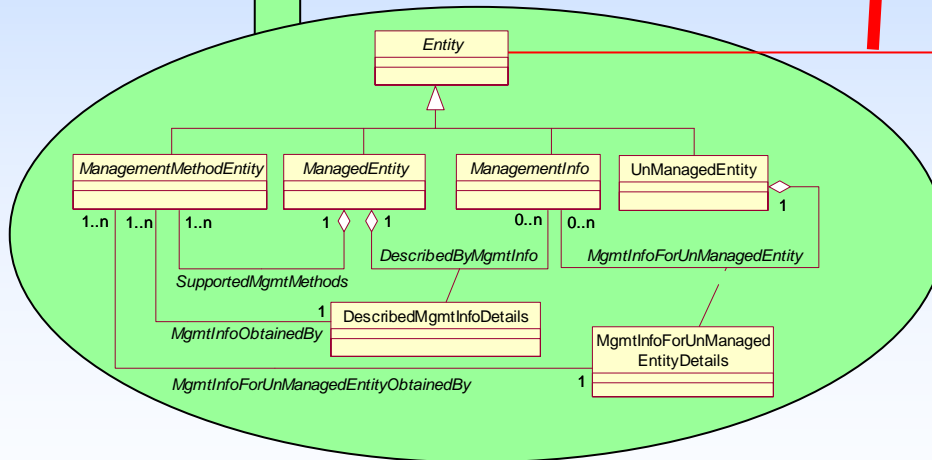
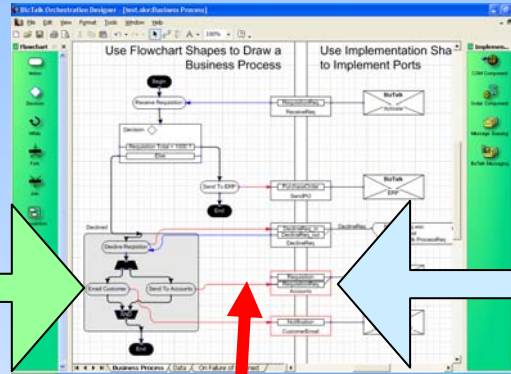
- Conversion of UML-isms to Java
- Class and attribute name translation driven by configurable property files
- Extraction of enumerations from UML source files
- Auto-insertion of import statements to handle class dependencies
- Association maintenance
- Produce a *programmer's reference* to
 - Facilitate developer's usage and understanding of the model
 - Translate complex relationships, especially recursion, into English
 - Define what the legal containers of a given element are
 - Serves as a QA check of the model

NGOSS Lifecycle Methodology

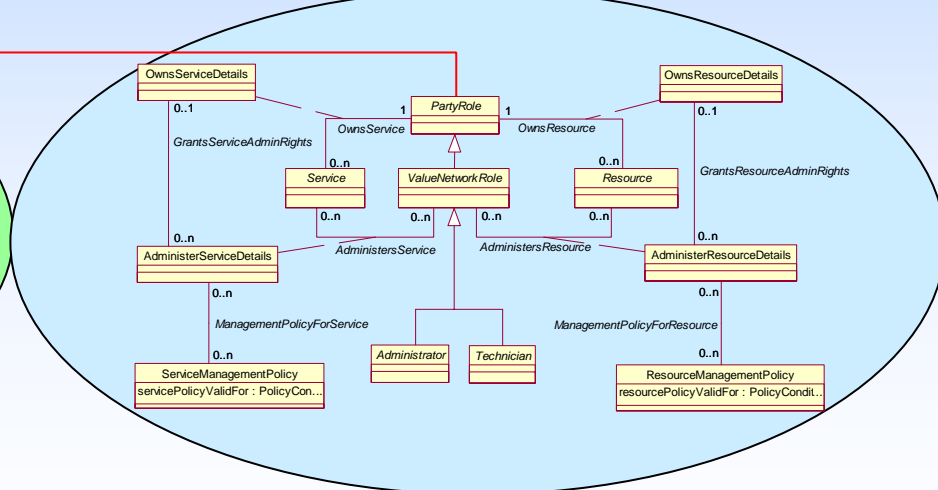


MDA a la DEN-ng

A Finite State Machine That Models Behavior



Model From Domain A



Model From Domain B

Results of an Early Prototype

- Trap an Event (e.g., Syslog Card Insert)
- Determine Type of Card ({V, T, M, OS})
- Determine Card Characteristics
- Associate Behavior With Card Ports
- Configure Ports According to Roles
- Validate and Notify OSS/BSS

- *Service is Pre-Provisioned and Ready for Billing, Hands-Free*