



Data Upgrade as a First-Class Citizen

Atzmon Hen-Tov
VP R&D - Pontis

Saturn 2015
Baltimore, MD - April 27-30, 2015





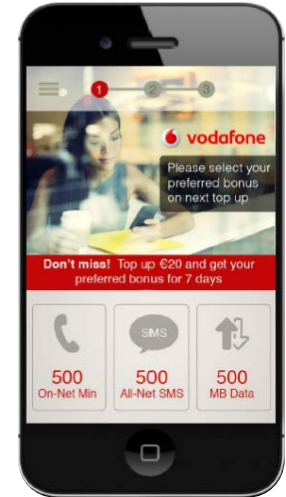
Pontis is a developer of iCLM™

An Online Marketing Automation product for communication service providers

Our customers are Tier I operators in Europe, CIS, South-America, Africa and the Far east

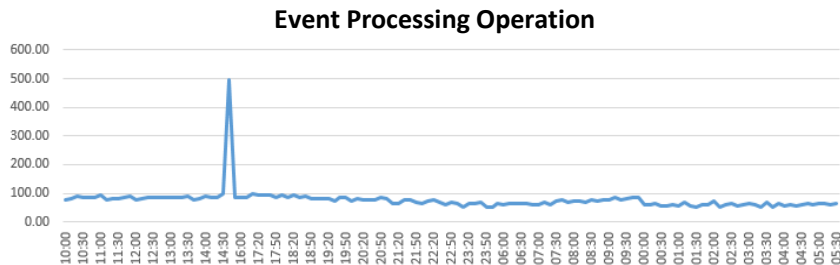


The product engages the customers proposing products and services that best suit the individual customer based on analysis of the customer behavior

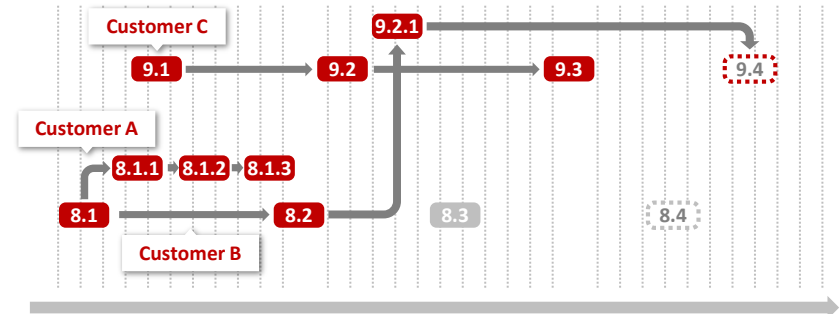


PONTIS

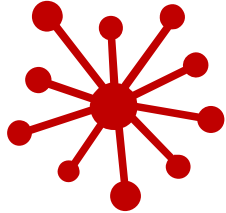
These products conform to strict non functional requirements (Telco grade) supporting billions of transaction per day



The dynamics of the marketing domain requires high agility from the product



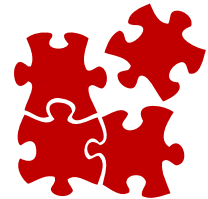
Problem Statement



**Many
production
systems**



**Different
development teams**

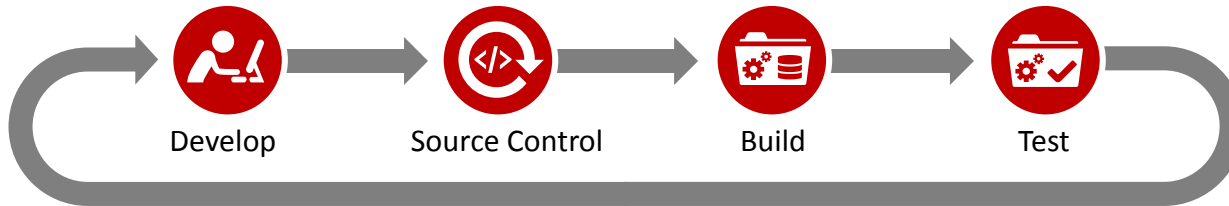


**Multiple
technologies**

We need to upgrade frequently, we cannot risk data loss!

Limitation of Existing Solution

We followed Continuous Integration best practices and recipes



BUT... we found the following deficiencies:

You might **miss changes** not covered by tests



You are counting on **developers awareness**



DBA involvement **hampers agility**



Developers **cannot master all** technologies



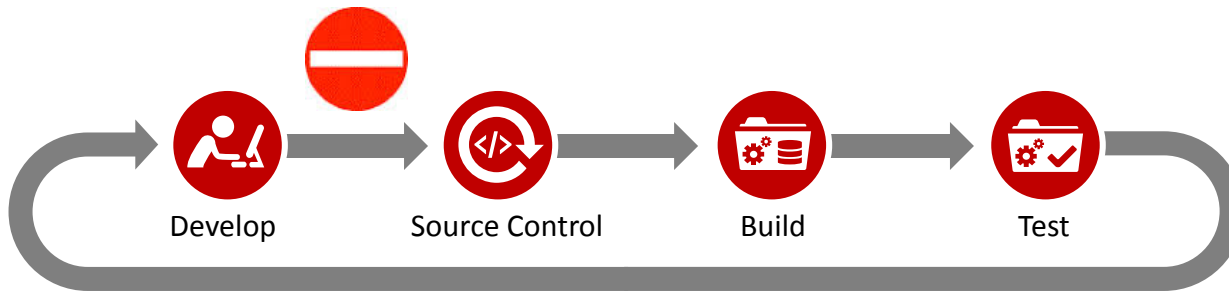
Developers tend to **avoid refactor** after a feature is shipped



In NoSQL, **lack of schema** further intensifies the risk (Polyglot persistence)

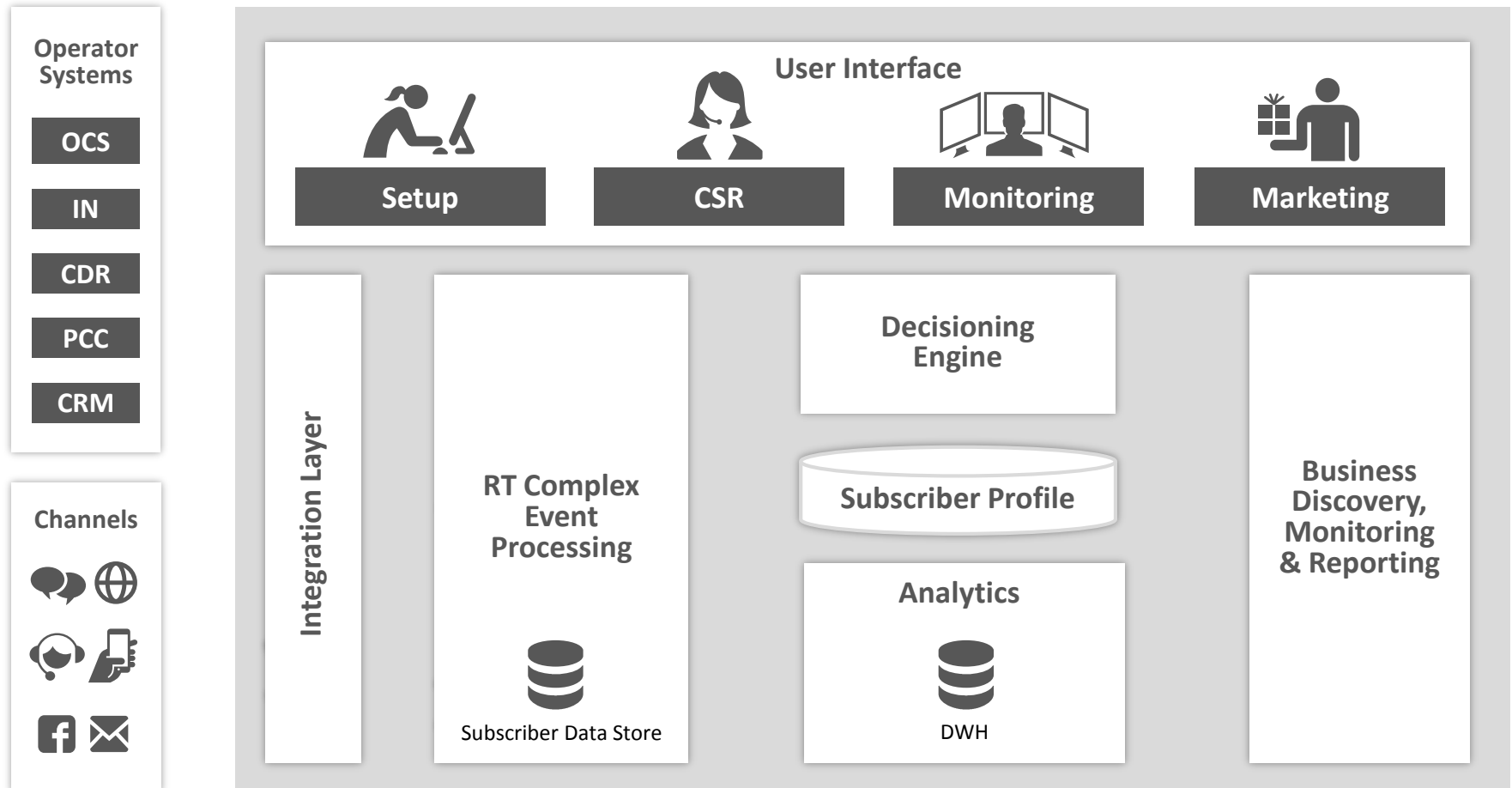


Outline of Our Solution



1. **Validation** - The IDE alerts the developers on changes that they didn't handle
2. **Data Upgrade DSL** - Upgraders are written in a DSL abstracting the underlying technologies and supporting multiple technologies simultaneously

Background: High Level Architecture



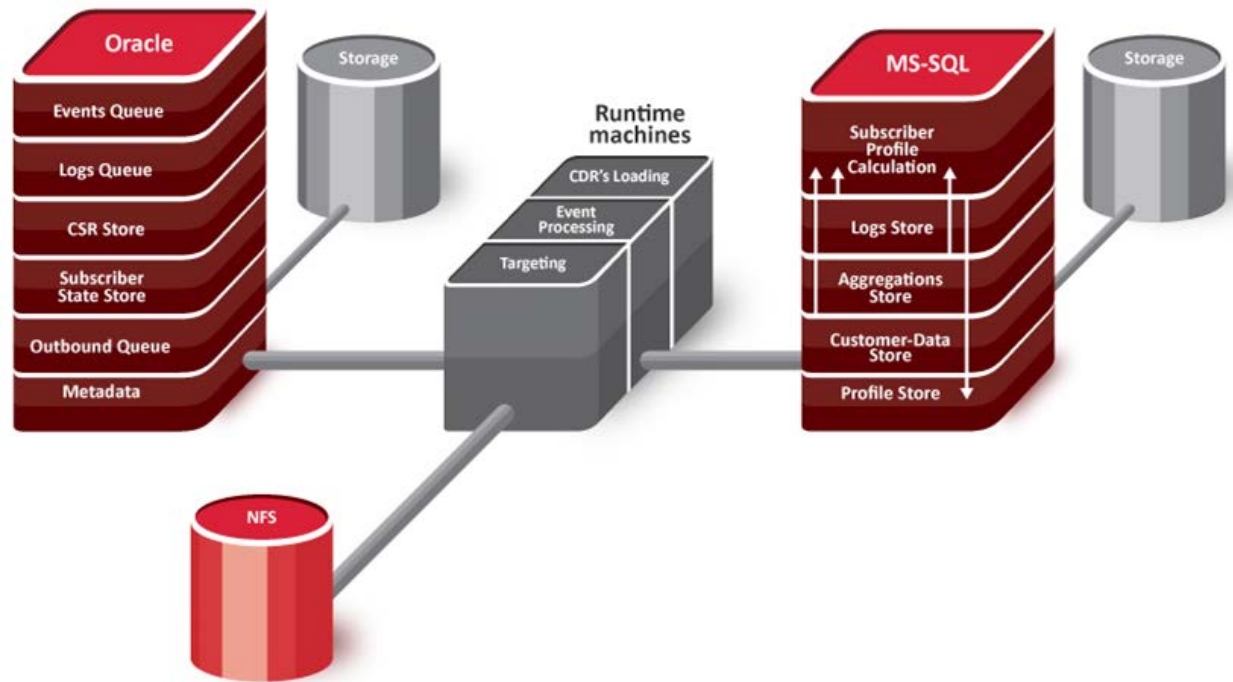
Background: Variability between customers

Multiple, highly customized instances exist for each customer and need to be supported



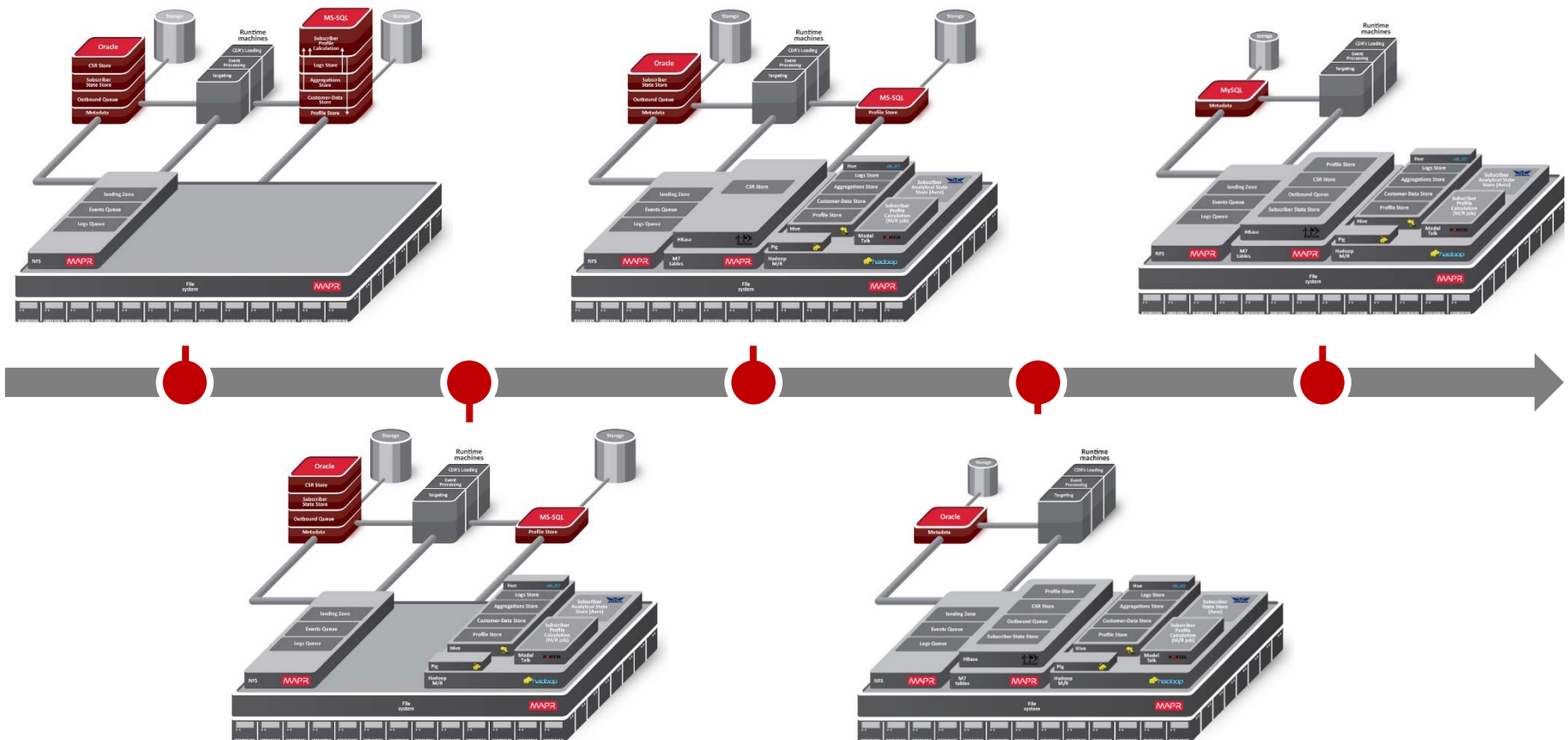
Background: Varying Technologies

Over the last 2 years we evolved our architecture to **Big-Data**



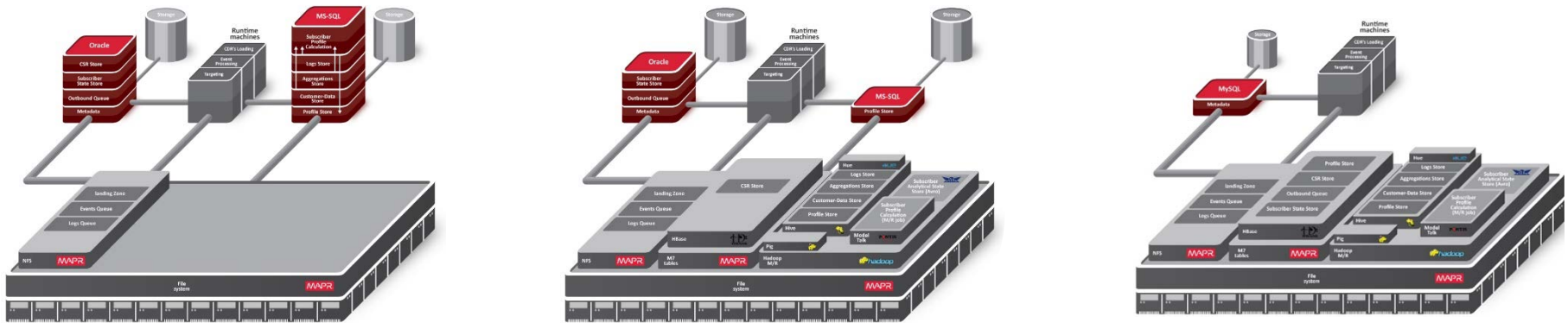
Background: Supporting Multiple Technologies

Now we support both legacy and Big-Data
...and permutations of thereof



Background: Supporting Multiple Technologies

Now we support both legacy and Big-Data
...and permutations of thereof

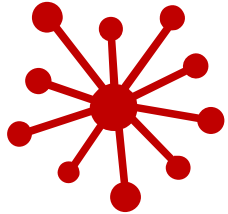


Multiple storage technologies and data formats

Entities are mapped to several store technologies



Problem Statement



**Many
production
systems**

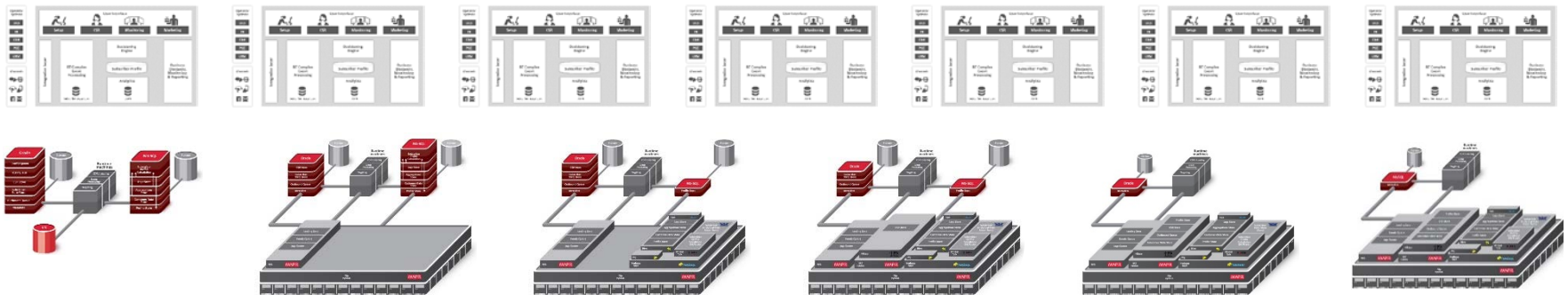


**Different
development teams**



**Multiple
technologies**

We need to upgrade frequently, we cannot risk data loss!

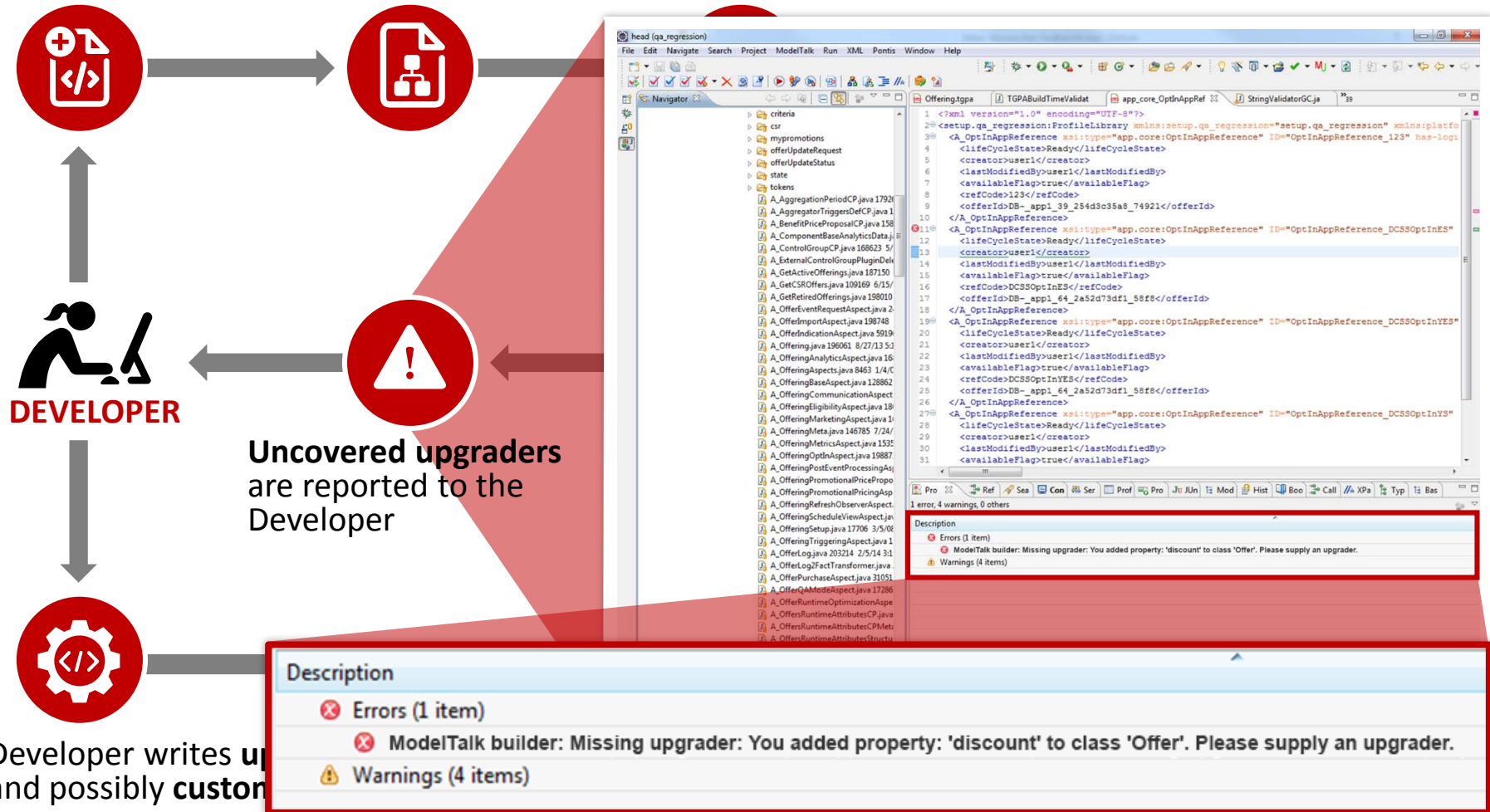


Our Solution: IDE support

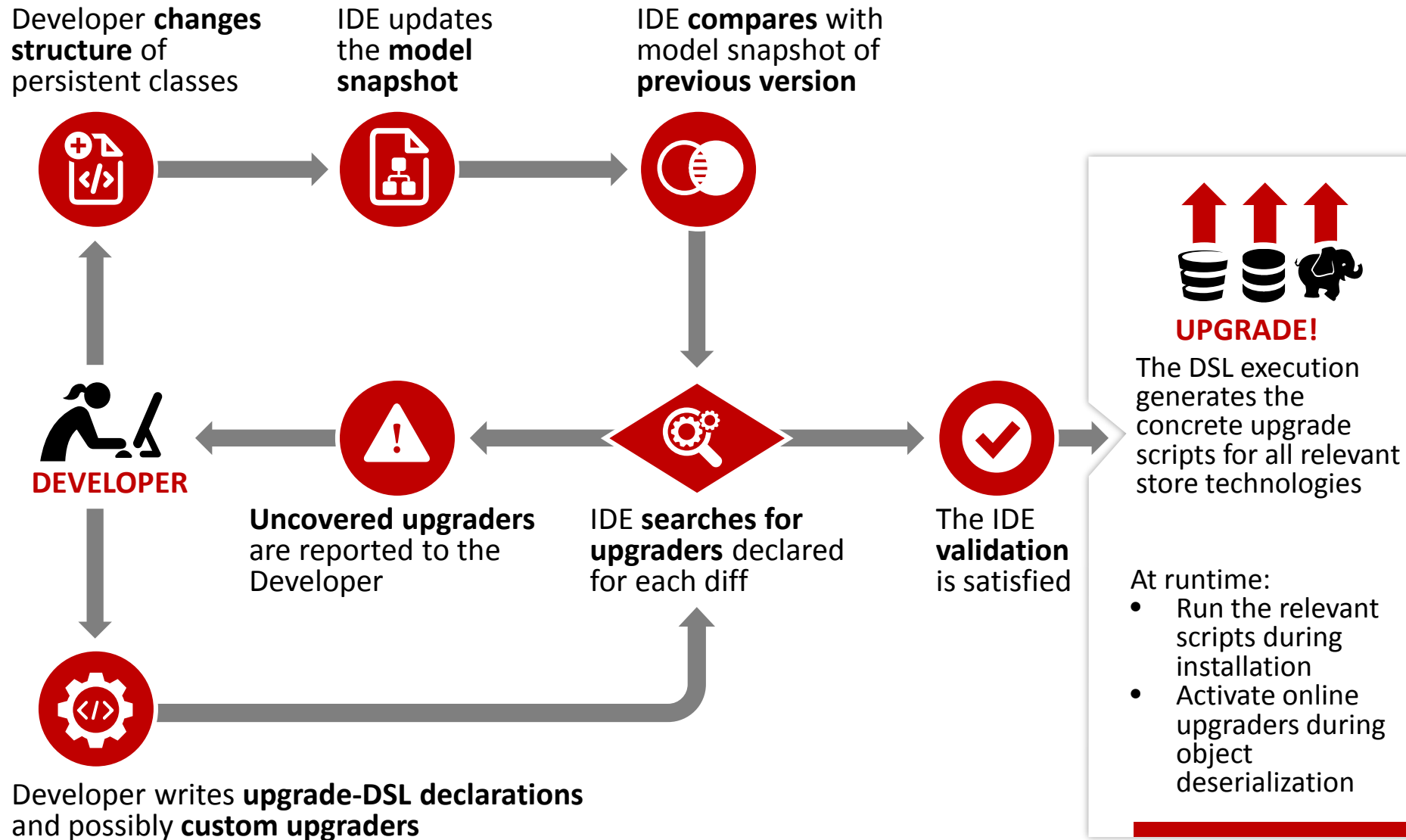
Developer changes
structure of
persistent classes

IDE updates
the **model
snapshot**

IDE compares with
model snapshot of
previous version

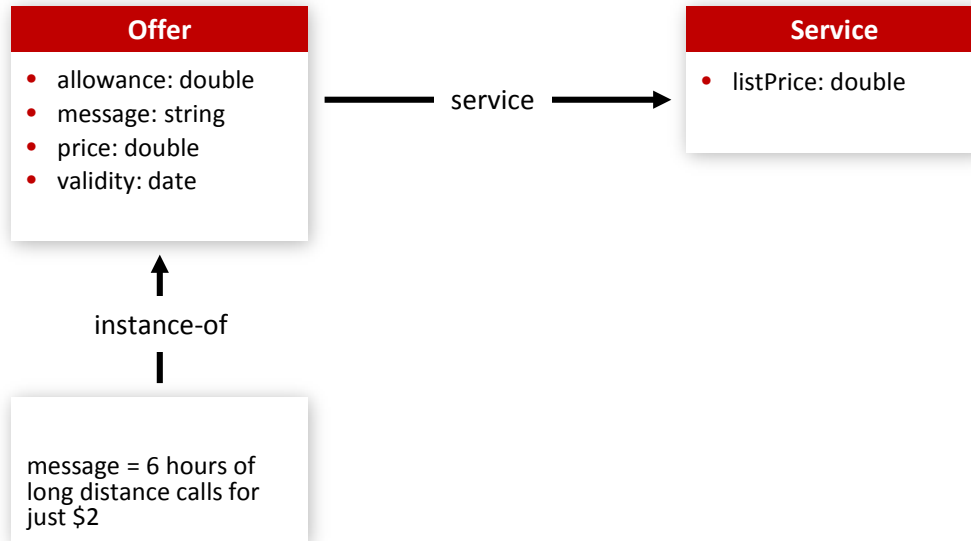


Our Solution: IDE support



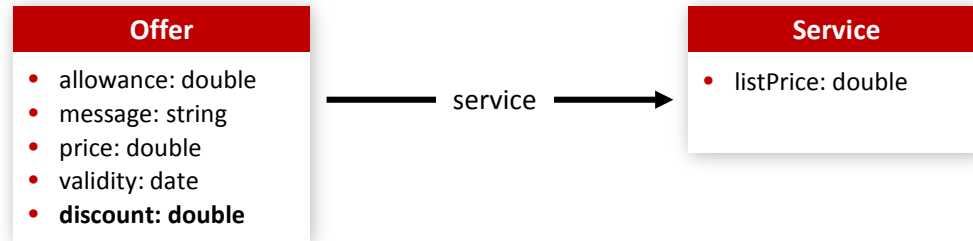


Our Solution: Upgrade DSL



Our Solution: Upgrade DSL

Model evolution – Adding numeric discount property

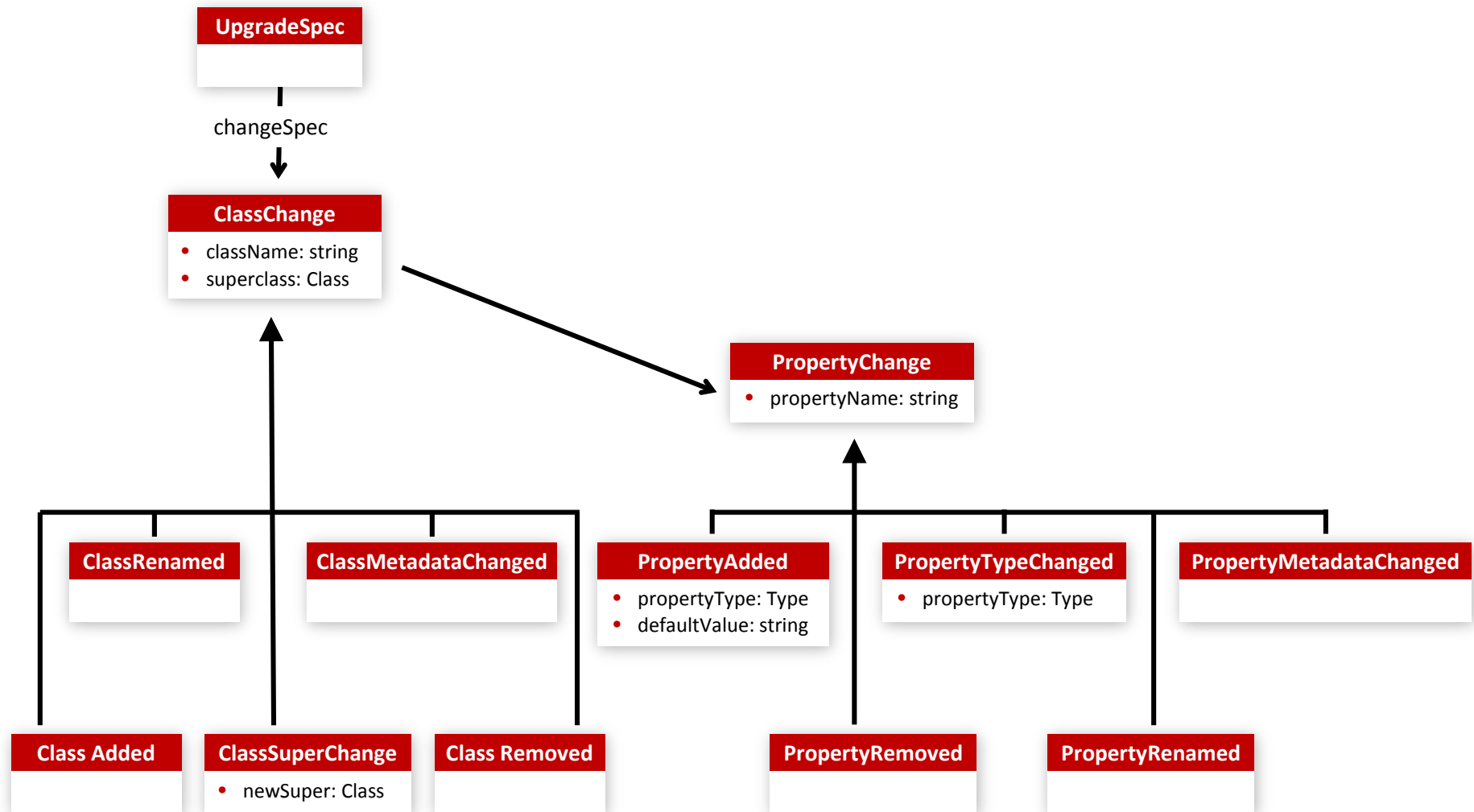


Upgrade

```
UpgradeSpec {
  ChangeSpec class="ClassChange" {
    className="Offer"
    Changes {
      Change class="PropertyAdded" {
        propertyName="discount"
        defaultValue=0
      }
    }
    Upgraders {
      Upgrader class="AutoUpgrader" {}
    }
  }
}
```

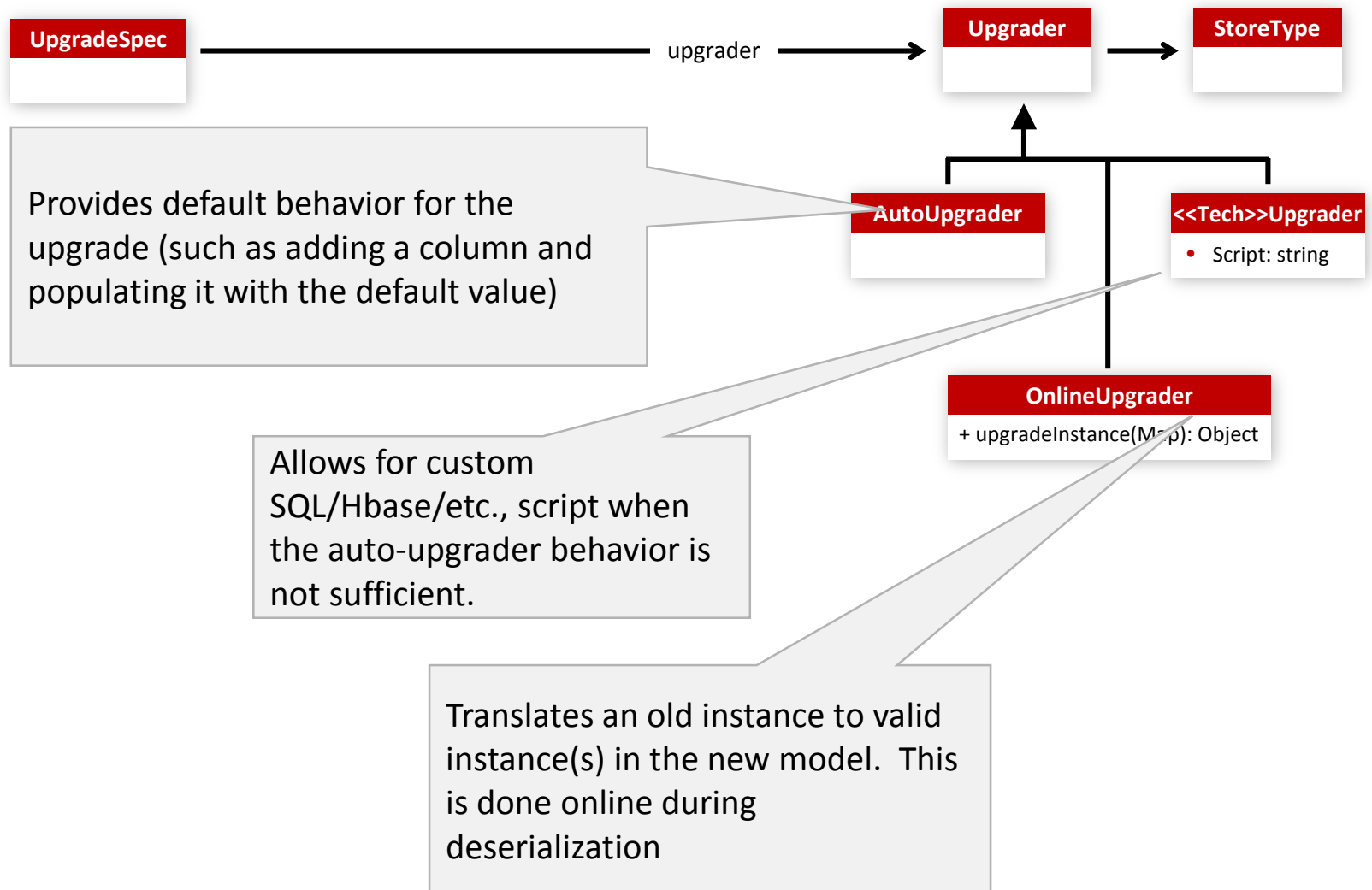
Our Solution: Upgrade DSL

Upgrade Language Meta Model



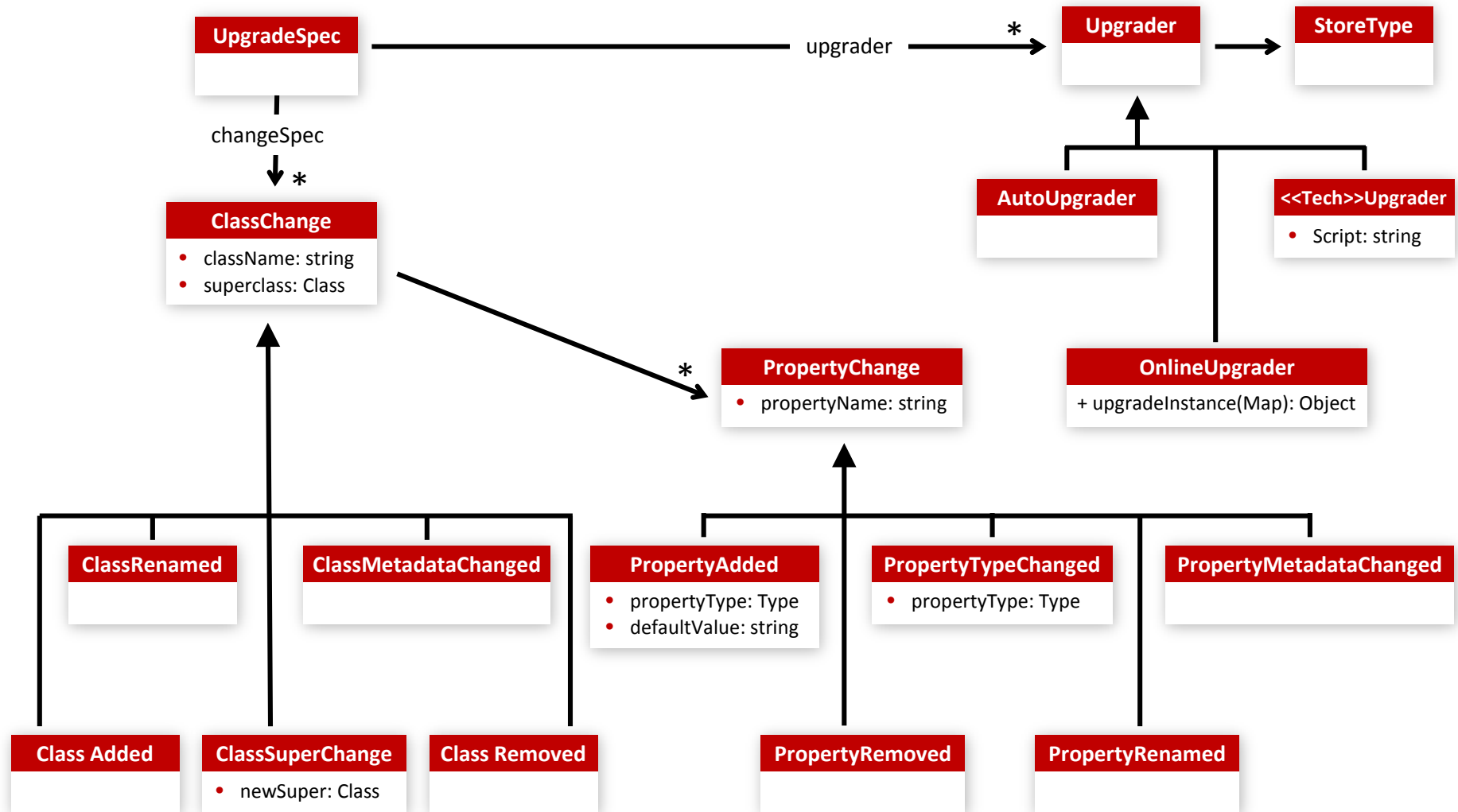
Our Solution: Upgrade DSL

Upgrade Language Meta Model



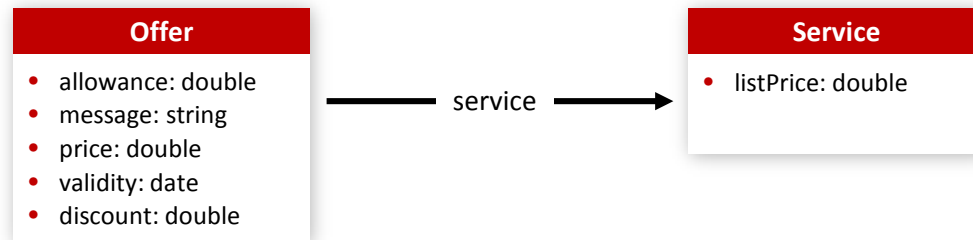
Our Solution: Upgrade DSL

Upgrade Language Meta Model



Our Solution: Upgrade DSL

Model evolution – Custom upgrader for the discount property



Upgrade

```
public class MyOfferUpgrader extends OnlineUpgrader {

    Object upgradeInstance(Map<String, Object> oldInstance) {
        Offer result = new Offer();

        // Copy all unchanged fields from the map to the new instance.
        super.copyAllFields(oldInstance, result);

        // Calculate the discount.
        result.setDiscount(result.getService().getListPrice() - result.getPrice());

        return result;
    }
}
```

```
age" {

    propertyAdded" {
        discount"

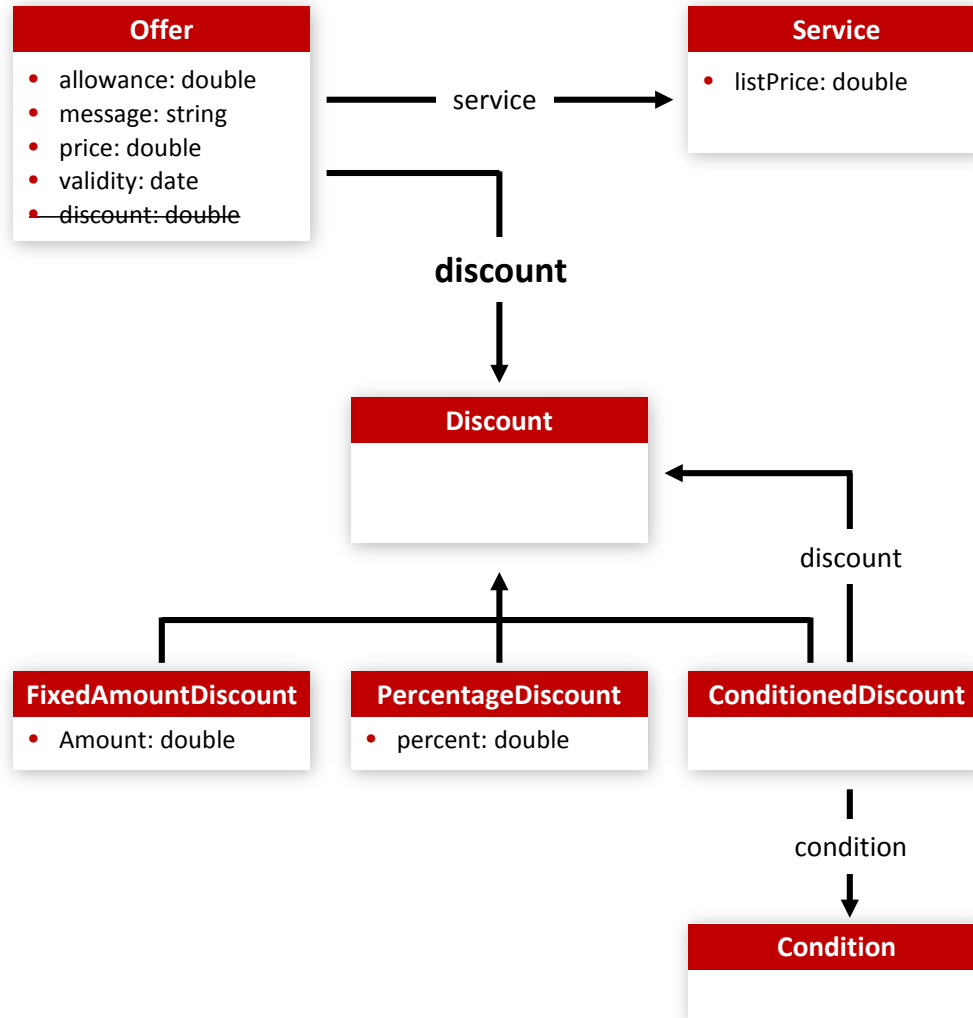
    }

    toUpgrader" {}

    instance with calculated discount:
    ce
    offerUpgrader" {
        "com.pontis.upgraders"
```

Our Solution: Upgrade DSL

Model Evolution – Discounters added to the model. “discount” is changed to be of type Discount



Upgrade

```
UpgradeSpec {
  ChangeSpec class="ClassAdded" {
    className="Discount"
  }
  ChangeSpec class="ClassAdded" {
    className="FixedAmountDiscount"
  }
  ChangeSpec class="ClassChanged" {
    className="Offer"
    Changes {
      Change class="PropertyTypeChanged" {
        propertyName="discount"
        newType="Discount"
      }
    }
  }
  Upgraders {
    Upgrader class="AutoUpgrader" {}

    // transform the numeric discount into
    // an instance of FixedAmountDiscount
    Upgrader class="MyOfferDiscounterUpgrader" {
      java_package = "com.pontis.upgraders"
    }
  }
}
```

Related work

Database Continuous Integration

- [Sadalage 2007] **Recipes for Continuous Database Integration**
- [Fowler & Sadalage 2012] **NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence**
- [Humble, Farley] **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**

Evolution patterns

- Fowler, Martin; Sadalage, Pramod J. **Evolutionary Database Design** <http://martinfowler.com/articles/evodb.html>
- [Hen-tov, Lorenz, Nikolaev, Schachter, Wirfs-Brock, Yoder] **Dynamic Model Evolution**, In Proceedings of the 17th Conference on Pattern Languages of Programs. SPLASH/OOPSLA 2010.
- [Hen-tov, Nikolaev, Schachter, Wirfs-Brock, Yoder] **Adaptive object-model evolution patterns**, SugarLoafPLOP 2010

Domain Specific Language Workbench

- The Ink language workbench: <https://code.google.com/a/eclipselabs.org/p/ink/>
- [Hen-tov, Lorenz, Pinhasi, Schachter] **ModelTalk: When Everything Is a Domain-Specific Language**. IEEE Software, vol. 26, no. 4, pp. 39-46, July/Aug. 2009, doi:10.1109/MS.2009.972009

Conclusion

- **Problem**
 - No support for Data upgrade in development environment
 - Current practices rely on developer awareness
- **Solution – Technique for handling data upgrades using tool support and DSLs**
 - IDE alerts developers at design time on model changes
 - Simplified handling of data upgrade with DSLs

Pontis has implemented the solution and gathered experience from hundreds of SW upgrades in numerous Telco production sites

- **Our (subjective) assessment:**
 - Enhanced agility
 - Less data upgrade issues leak to productions
 - Developers are not avoiding refactor

Thank You!

Atzmon Hen-tov

Atzmon@ieee.org

How do we tackle both variability and agility

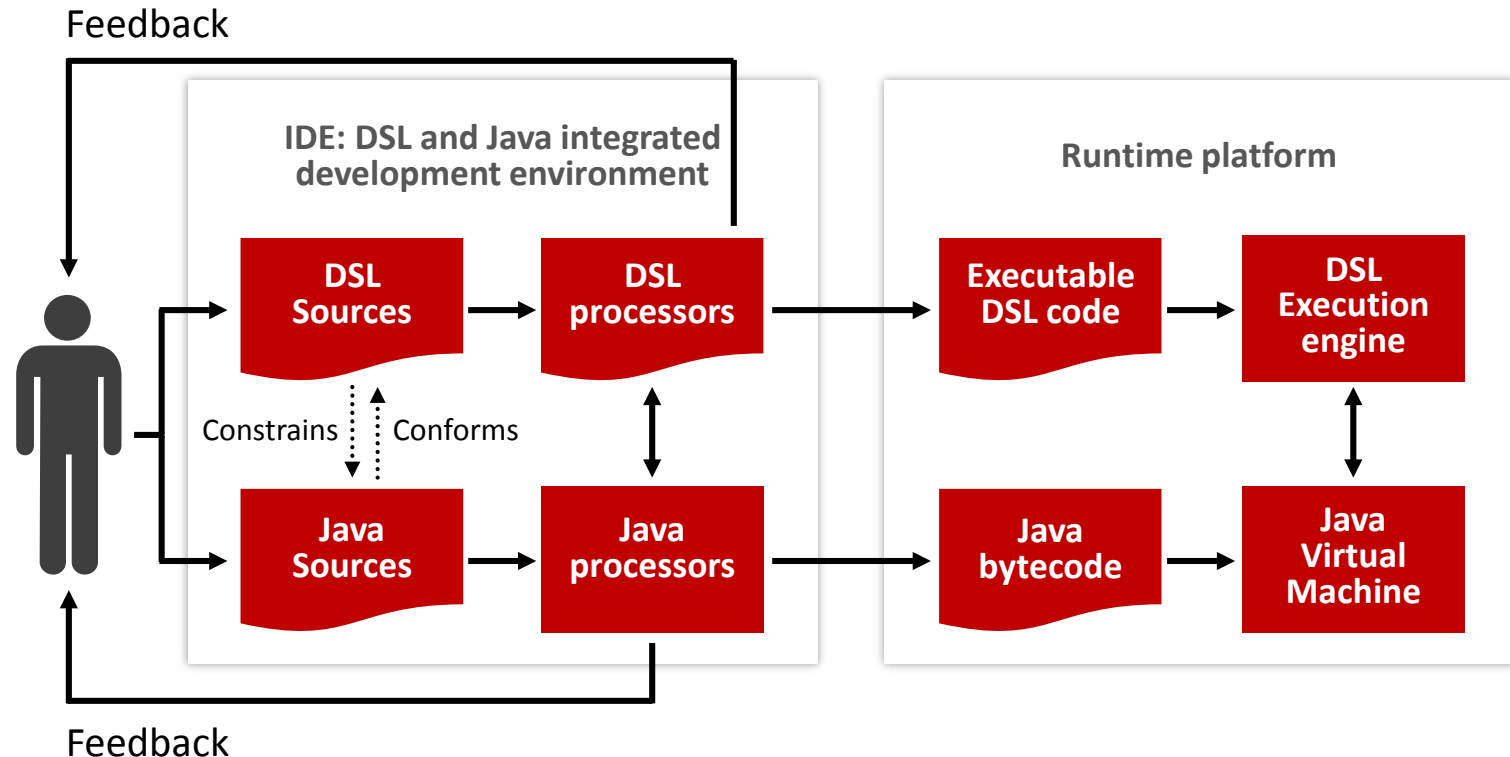
To address the variability challenges our development is based on:

Layered pipeline

Executable Model

AOM

ModelTalk: An interpretive domain specific language workbench



DSL: Domain-specific language

How do we tackle both variability and agility

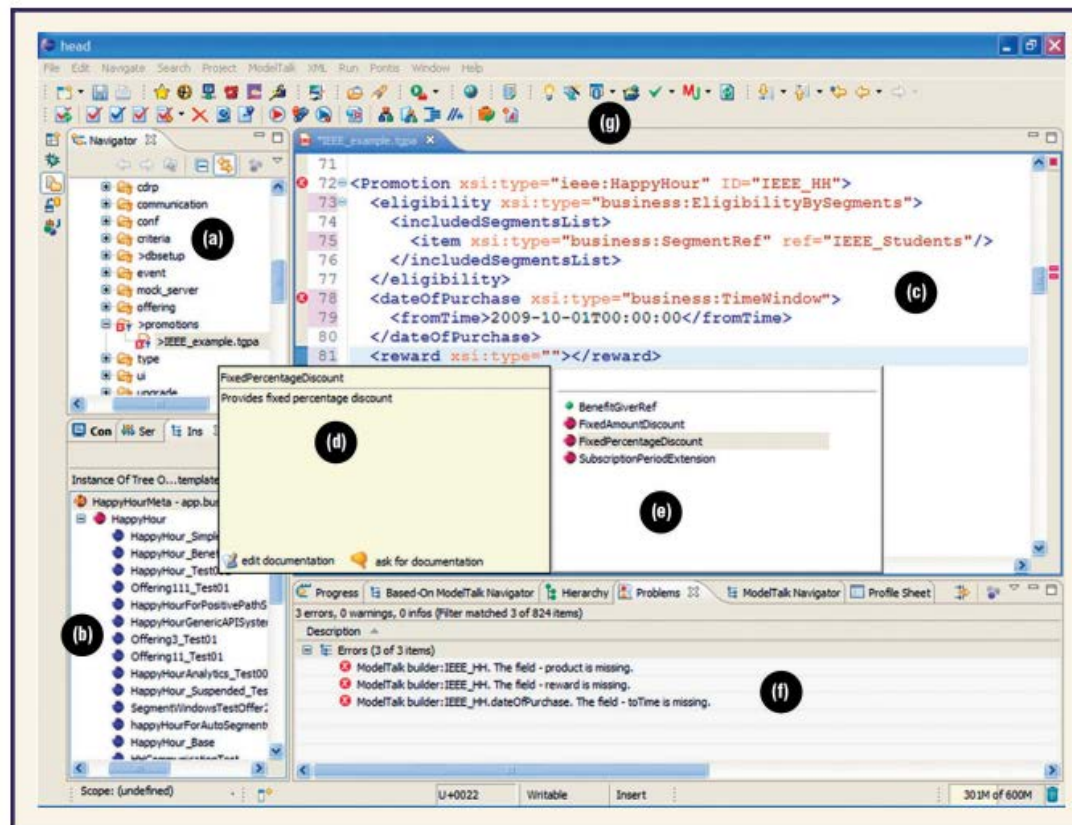
To address the variability challenges our development is based on:

Layered pipeline

Executable Model

AOM

Modeltalk: An interpretive domain specific language workbench



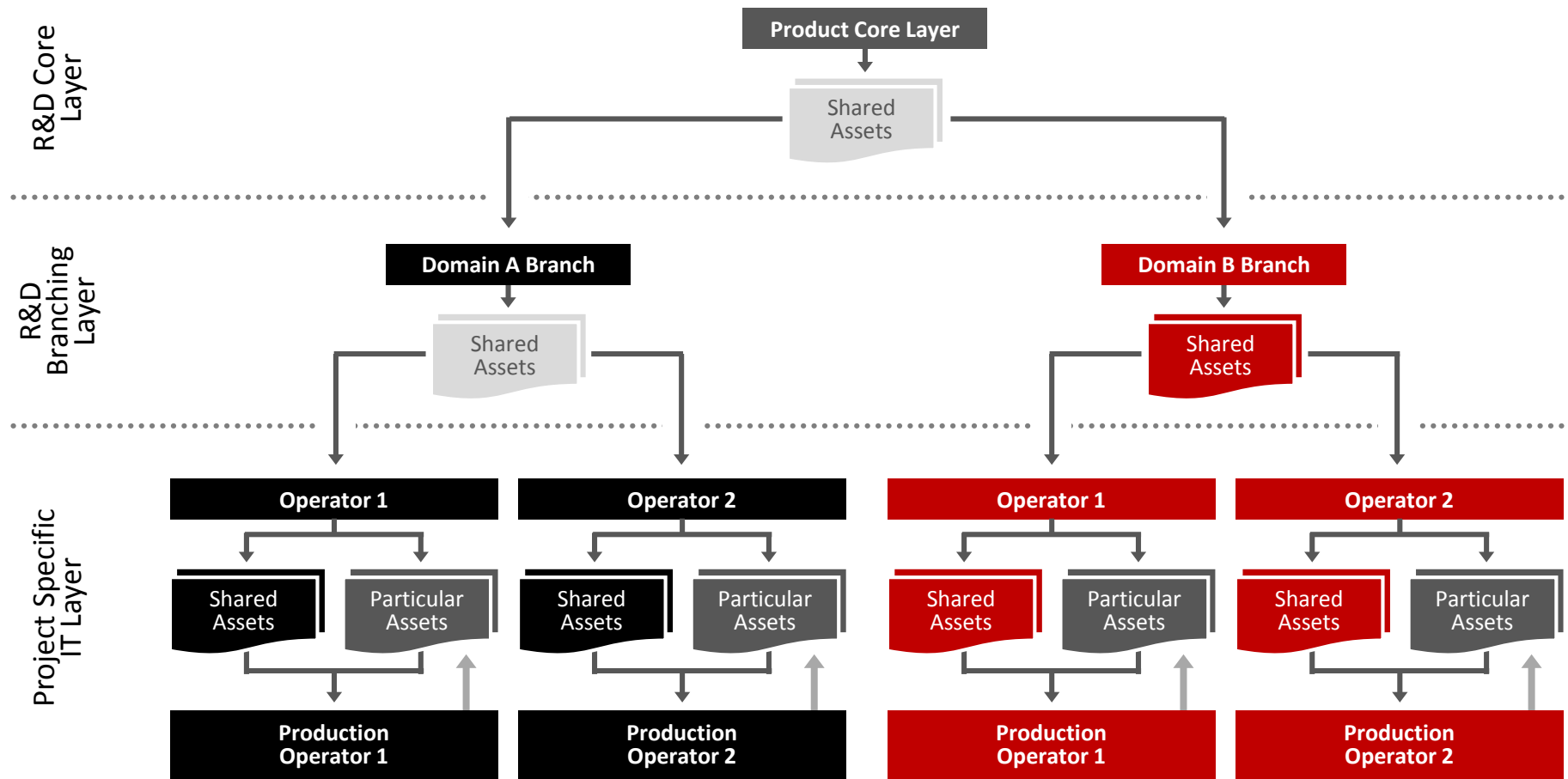
How do we tackle both variability and agility

To address the variability challenges our development is based on:

Layered pipeline

Executable Model

AOM



How do we tackle both variability and agility

To address the variability challenges our development is based on:

Layered pipeline

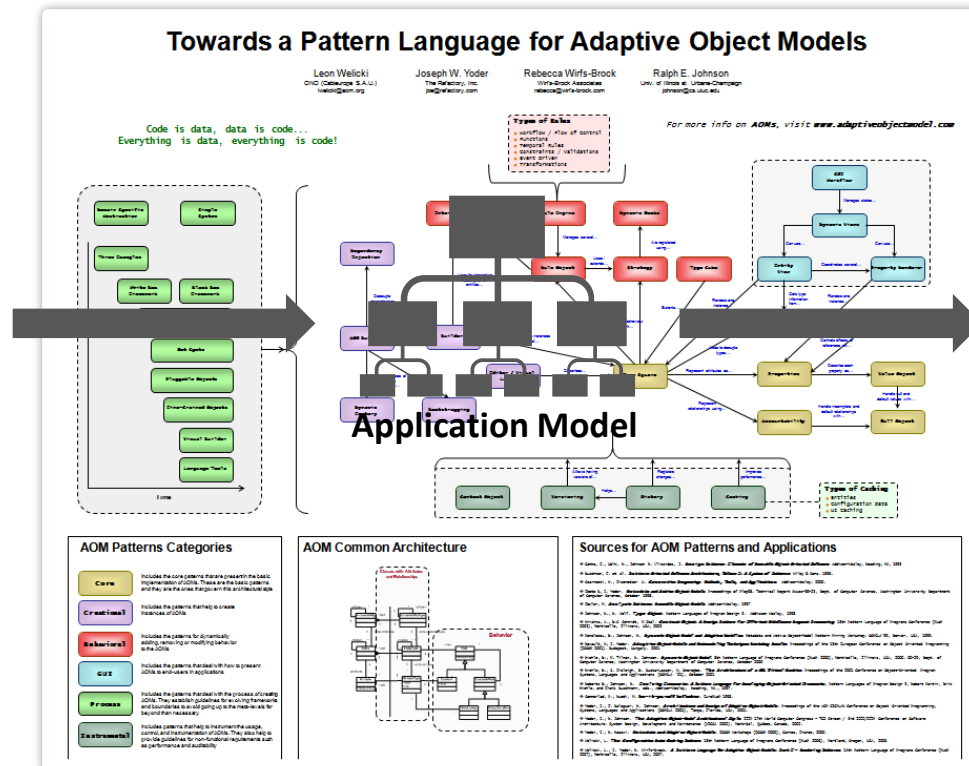
Executable Model

AOM

A strong metadata layer empowering users to evolve the system (see more in <http://adaptiveobjectmodel.com/>)



Power User



Business User

High-Performance Dynamic Healthcare Analytics

Jordan Menzin

**Software Architect and Product Lead
Boston Health Economics, Inc. (BHE)**

SATURN Conference

April 28, 2015

Company

- Healthcare analytics company that provides insights into the value of medical technologies
- Small, cross-functional team

Platform

- Analytics platform that allows analysts to complete projects 90% faster than with traditional approaches
- Users can implement a broad spectrum of data analytics projects
- Currently supports 12 types of datasets

Users

- Research analysts and managers
- Life Sciences
 - Pharma
 - Biotech
 - Medical devices
- Health Plans
- Medical Groups

The Data

- Data classes are electronic health records, administrative claims, hospital billing/clinical data, survey data
- Within each class there are many different specific data sources provided by commercial companies or government agencies
- Relational schema or denormalized schema (sometimes mixtures of both with schema varying year-by-year)
- Datasets have lots of variation in terms of files (tables) provided/fields available

The Data, cont'd.

- “Large” datasets
 - 150 million patients
 - 100 billion rows
 - 6TB
- Statistically De-Identified

Data Model

- Demographics
- Enrollment
- Encounter
- Medication
- Diagnosis
- Procedure
- Lab
- Observation
- PhysicianNote

Example Projects

Therapeutic Area	Data Source	Analysis Topic	Results
Atrial Fibrillation	EMR	Matched cohort analysis of resource use and outcomes	1 meeting abstract 1 manuscript in progress
Diabetes	EMR	Treatment patterns and outcomes	Study report
Epilepsy	Claims	Adverse events, adherence, and adjunctive therapy	4 meeting abstracts
Multiple Sclerosis	Claims	Prevalence and treatment rates/ outcomes	3 meeting abstracts
Orthopedic Surgery	Hospital	Matched case control analysis of costs	3 meeting abstracts
Stroke	Claims	Risk factors and costs	2 meeting abstracts 1 manuscript 1 manuscript in progress
Venous Thromboembolism	EMR	Treatment patterns and Outcomes	1 meeting abstract 1 manuscript

Example Projects, cont'd.

CLINICAL FOCUS: CARDIOMETABOLIC HEALTH, AND PULMONARY AND
VASCULAR MANAGEMENT

Treatment Patterns and Outcomes Among Hospitalized Patients With Venous Thromboembolism in the United States: An Analysis of Electronic Health Records Data

Joseph Menzin, PhD¹
Ronald Preblich, PharmD,
MPH²
Mark Friedman, MD¹
Jordan Menzin¹
Molly Frean¹
Winghan Jacqueline Kwong,
PhD, PharmD²

¹Boston Health Economics, Inc.,
Waltham, MA; ²Daiichi Sankyo, Inc.,
Parsippany, NJ

Correspondence: Joseph Menzin, PhD,
Boston Health Economics, Inc.,
20 Fox Road, Waltham, MA 02451.
Tel: 781-290-0808
Fax: 781-290-0029
E-mail: jmenzin@bhei.com

DOI: 10.3810/hp.2014.10.1143

Abstract

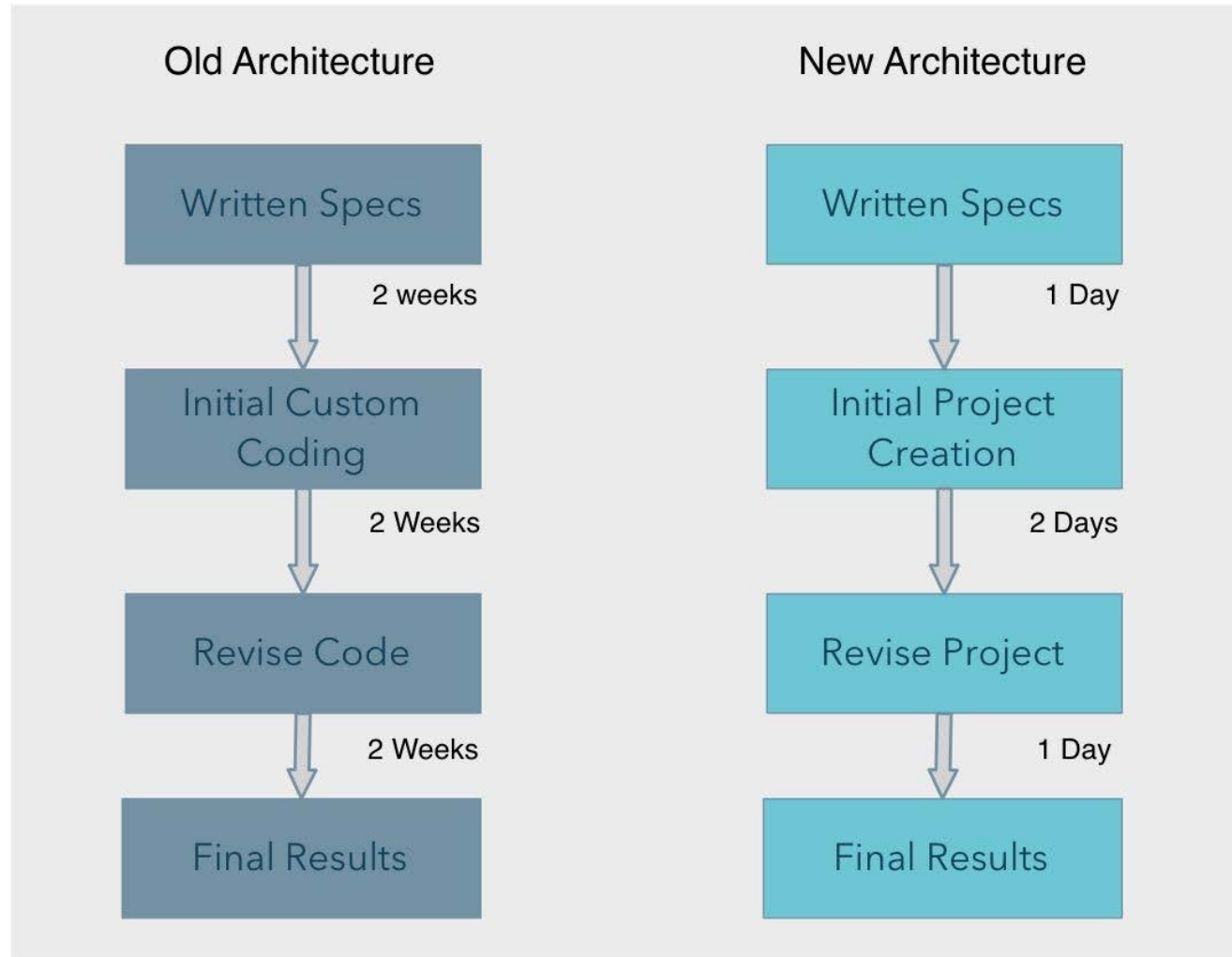
Background: With the advent of new treatment options for venous thromboembolism (VTE), it is valuable to gain insights into current clinical practices. **Objective:** Assess treatment patterns and recurrence among patients hospitalized for VTE. **Methods:** This retrospective study evaluated patients hospitalized with an incident VTE diagnosis (index) from 2008 to 2012 in a de-identified electronic health record database. Patients were further required to receive anticoagulant treatment and/or a VTE-related procedure for study inclusion. Patients were excluded if they: (1) did not have a medical encounter in the 6 months before index (baseline); (2) had a prior VTE diagnosis or used an anticoagulant during the baseline period; or (3) had a diagnosis of atrial fibrillation/flutter, cardiomyopathy, or a coagulation disorder during baseline or the year after index (follow-up). Hospitalization for recurrent VTE and bleeding were evaluated. **Results:** A total of 2060 patients were identified (mean age, 60.9 years; 53.0% women), with a mean length of stay of 8.1 days. Of the VTE types, acute DVT was the most common (41.9%), followed by PE (33.3%), and DVT + PE (24.7%). Almost all patients (96.9%) received anticoagulants, of which 94.3% received heparin and 76.5% received warfarin. Although 77.4% of warfarin users were prescribed it at discharge, only (40.2%) had a warfarin prescription within 30 days of discharge. Overall 30 day, 90 day and 1-year VTE recurrence rates were 2.0%, 4.2%, and 7.5%, respectively, and the major bleeding rate was 6.8%. **Conclusion:** In a real-world population of hospitalized VTE patients, heparin treatment in combination with warfarin was common. However, continuation of warfarin post-discharge was challenging. Initiatives to improve continuation of therapy may be important to reduce VTE recurrence.

Keywords: treatment; thromboembolism; anticoagulation; adverse outcomes

Introduction

Venous thromboembolism (VTE), including deep vein thrombosis (DVT) and pulmonary embolism (PE), has an estimated annual prevalence of > 400 cases per 100 000 patients in the United States, a figure that is expected to increase because of the documented increased risk of VTE with increasing age as well as the aging of the US population.¹⁻⁴ Venous thromboembolism poses a significant clinical and economic burden to those who are affected. The 1-year mortality rate after a VTE episode is estimated to be 17% to 22%, and, after a diagnosis of VTE, patients are prone to other complications such as recurrent VTE, major bleeding, heparin-induced thrombocytopenia,

Workflows



Demo

- Presented live at SATURN 2015

Data Model

- Adaptable Core
- Extensible
- Adapt or Extend?

Specification Language

- Users need a way to specify types of analyses they want to perform
- Can be defined using the web interface
- Can be submitted programmatically through a REST API
- “Internal” DSL

Code Generation

- Templates/Model Interpretation
- Complex, optimized code generated based on user specs

Web App

- No coding for end user
- Common UI for many data sources
- Concepts from domain model apply fairly directly

Key Flex Points

- Supporting Additional Data Classes
- Adding Patient Population Filters (Engineering)
- Adding Measures (Engineering)
- Inline Custom Measures (User/Engineering)
- Integration with R, SAS, other tools (User)

Lessons from Building Domain-Specific Analytics Platform

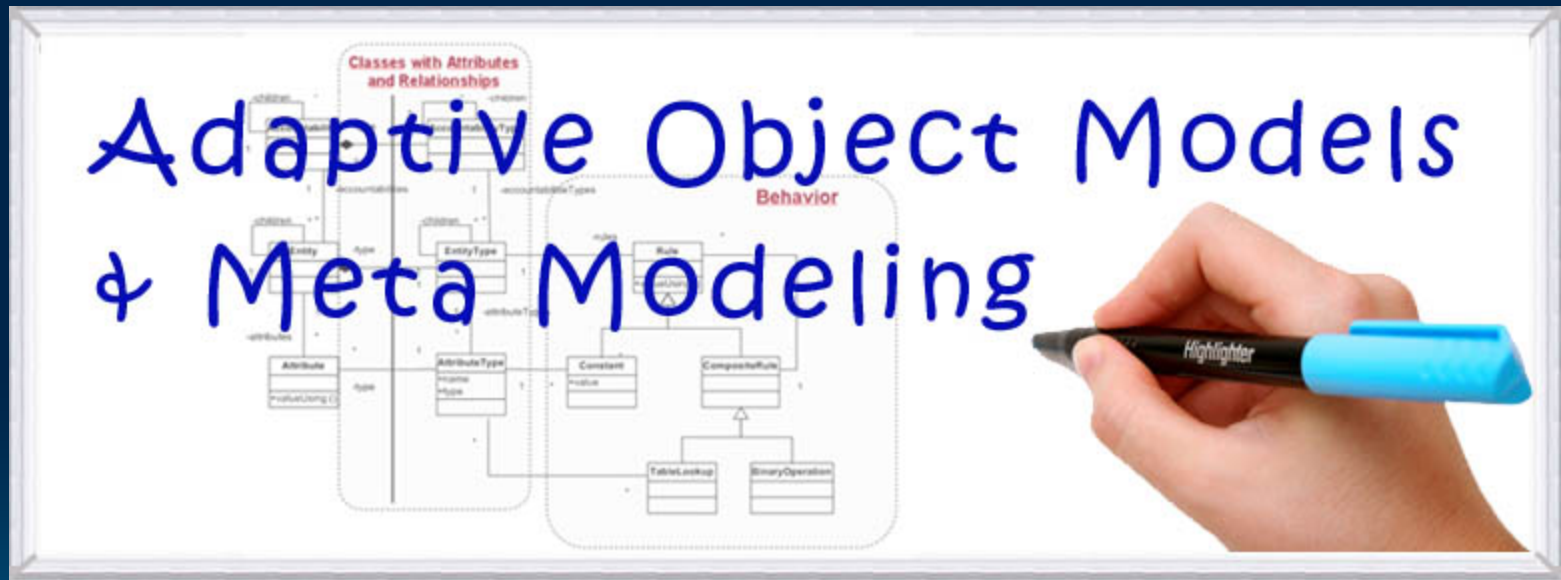
- Execution and programming time savings
- Higher quality
- Difficult to create appropriate abstractions to support variability
- Flexibility versus ease of use

Lessons, cont'd.

- 80/20 Rule
- Flex points
- Core Model
 - Adaptable
 - Extensible

Sustainable Architecture

“Supporting Data Variability”

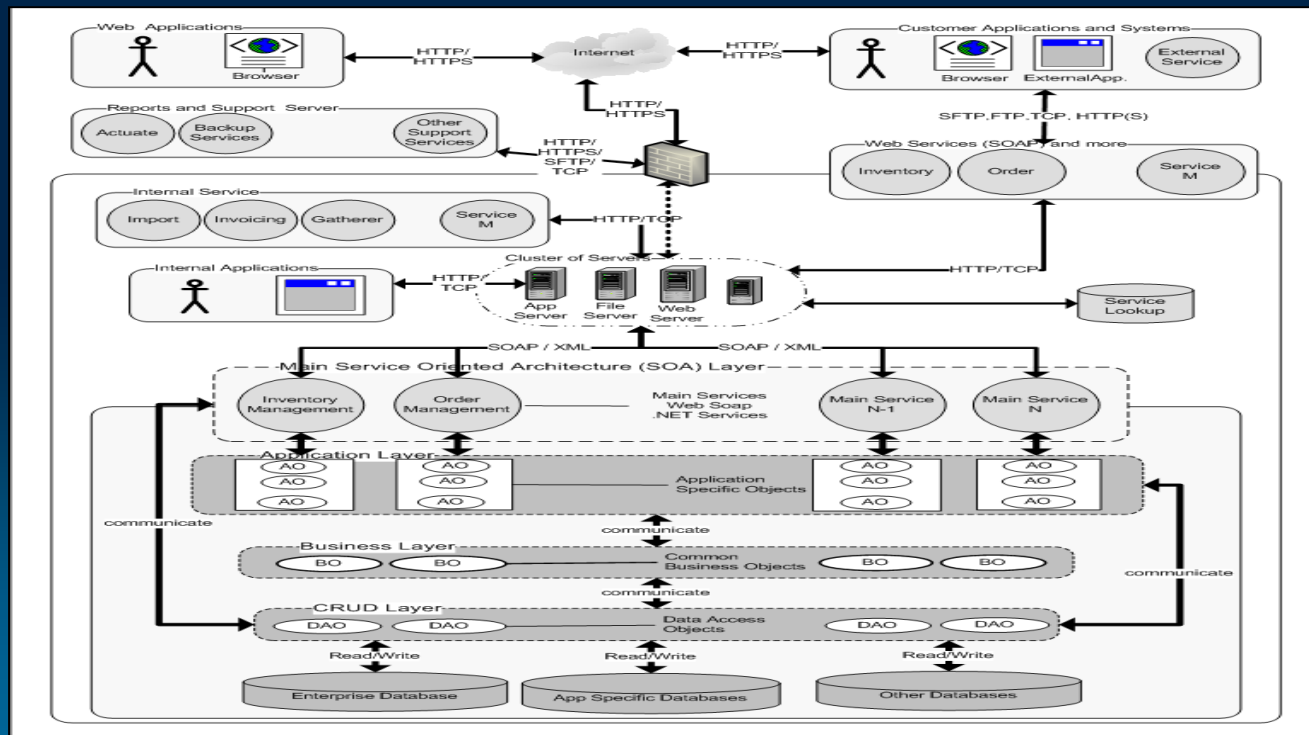


SATURN – April 28th, 2015
Joseph W. Yoder -- www.refactory.com

Sustaining Architecture

When dealing with data variability...

“What can be done to help sustain our architecture in the long run?”



Sustaining Your Architecture

Motivation: Need to Quickly Adapt to Change

- Business Rules and Domain Elements are changing quickly:
 - New calculations for insurance policies and new types of policies offered
 - Online store catalog with new products and services and rules applying to them
 - New cell phone product and services...
- Need quick ways to **develop** and **adapt** to these changing requirements.

Core Ideas

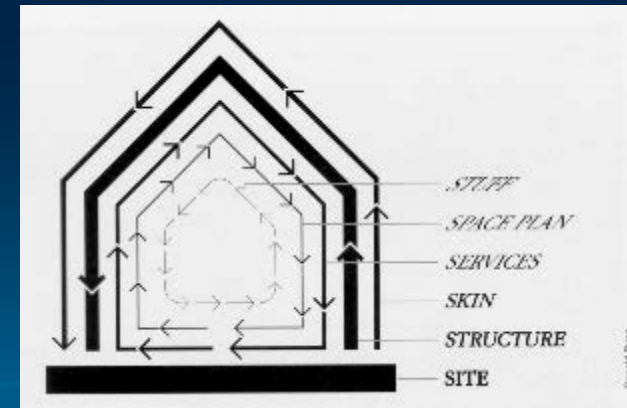
- Separate what changes from what doesn't change
- Protect core components
- Find the variability points and provide ways to support this variability in data
 - Description of changes (metadata)
 - Hook Points for variability
 - Dynamic Process Rules with Data





Stuart Brand's Shearing Layers

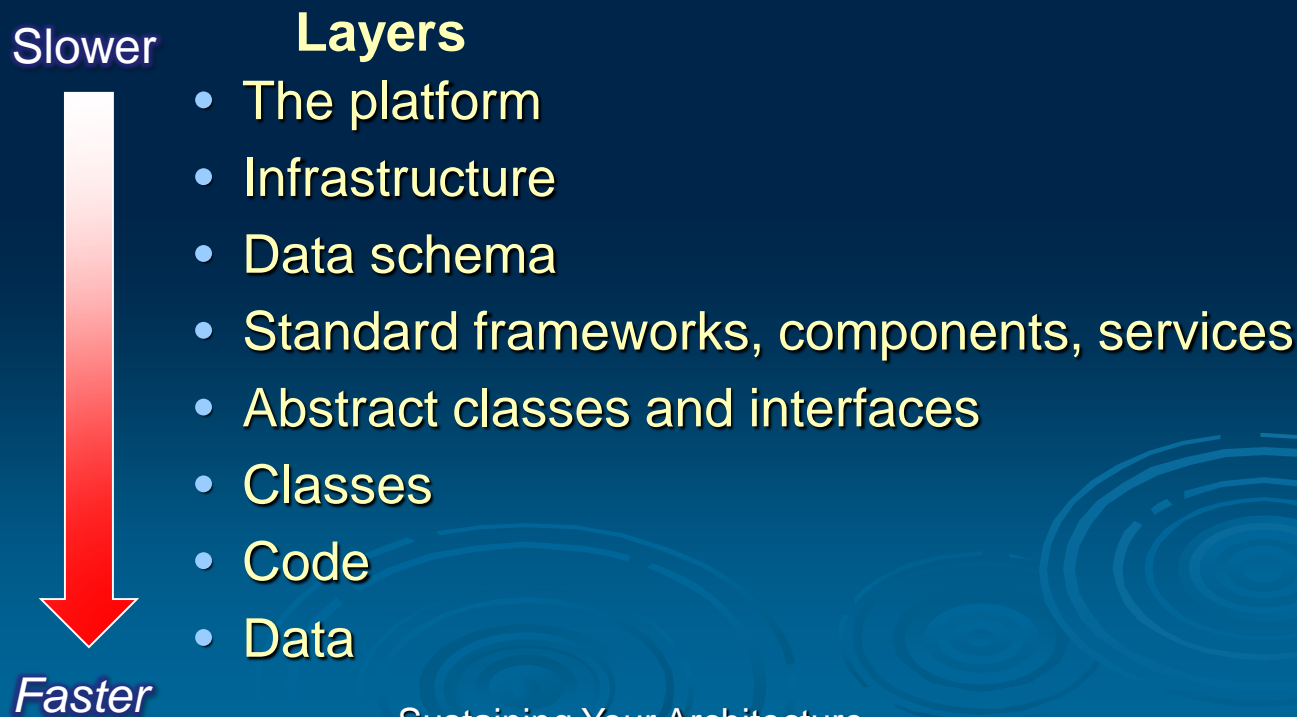
- Buildings are a set of components that evolve in different timescales.
- Layers: site, structure, skin, services, space plan, stuff. Each layer has its own value, and speed of change (pace).
- Buildings adapt because faster layers (services) are not obstructed by slower ones (structure).



—Stuart Brand, *How Buildings Learn*

Yoder and Foote's Software Shearing Layers

“Factor your system so that artifacts that change at similar rates are together.”—Foote & Yoder, Ball of Mud, PLoPD4.



Sweep It Under the Rug



Cover it up to keep other areas clean
(Façade and other Wrapper Patterns)



Put a rug at the Front Door

Protect Important Components!

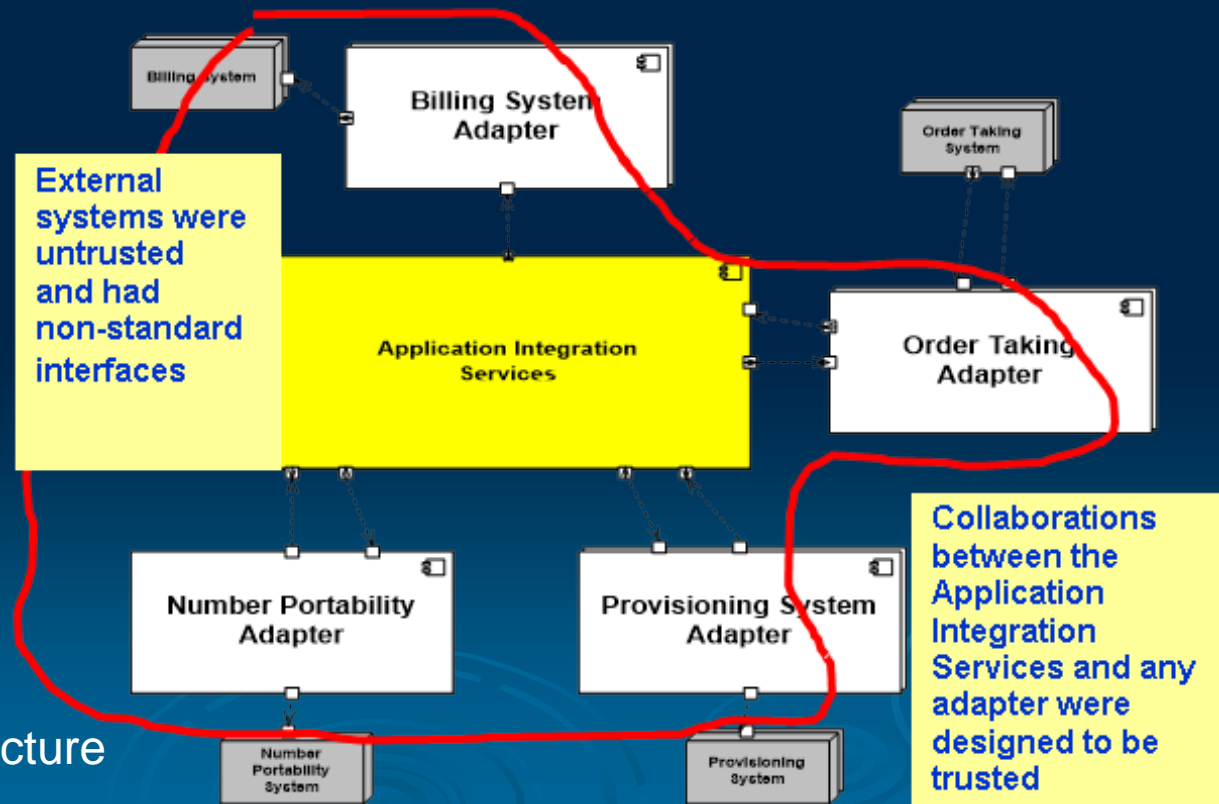
Keep other parts of the system clean.

Sometimes Glue code (Mediators) helps
keep others parts of the system cleaner.
(Anti-Corruption Layer -- Eric Evans)



Wipe your Feet at the Front Door

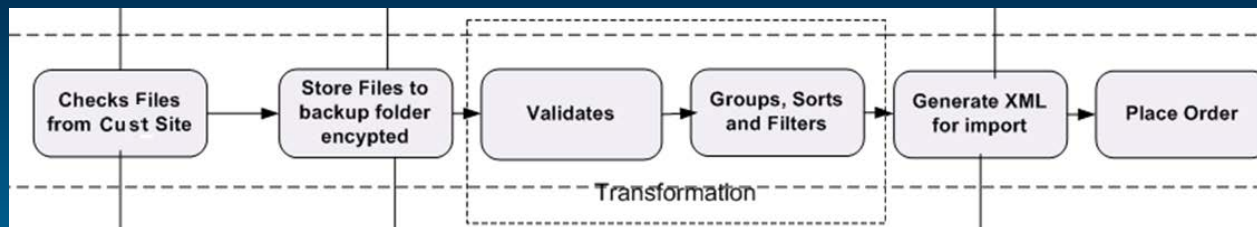
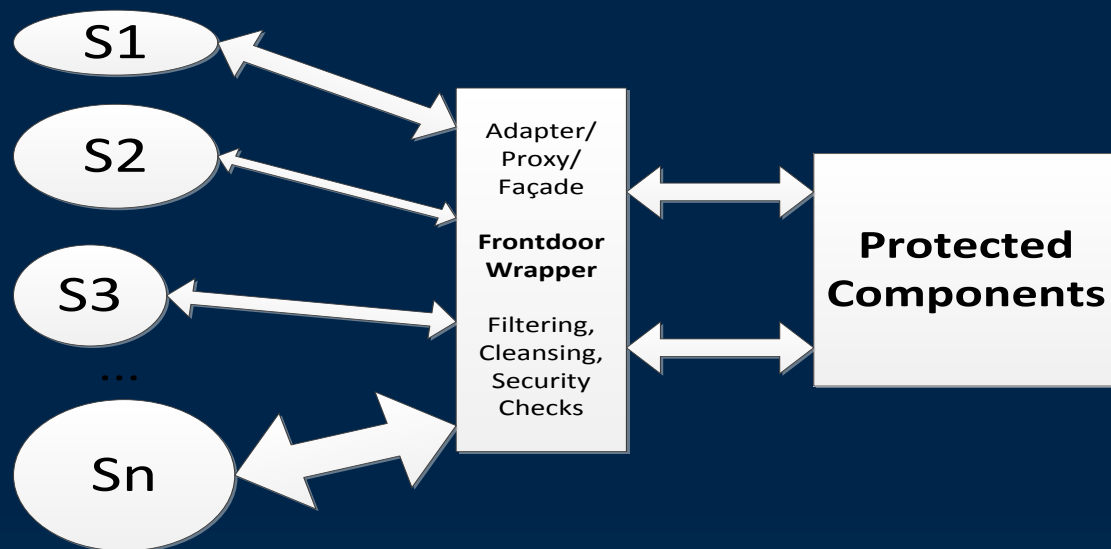
*ALIAS: ENCAPSULATE AND IGNORE
KEEPING THE INTERNALS CLEAN*



Patterns for
Sustaining Architecture
PLoP 2012 Paper

Sustaining Your Architecture

Wipe your Feet at the Front Door



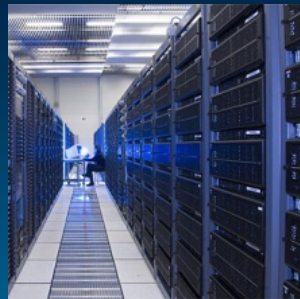
**Filtering and Cleansing Sequence to keep
Place Order Interface Clean**

Sustaining Your Architecture

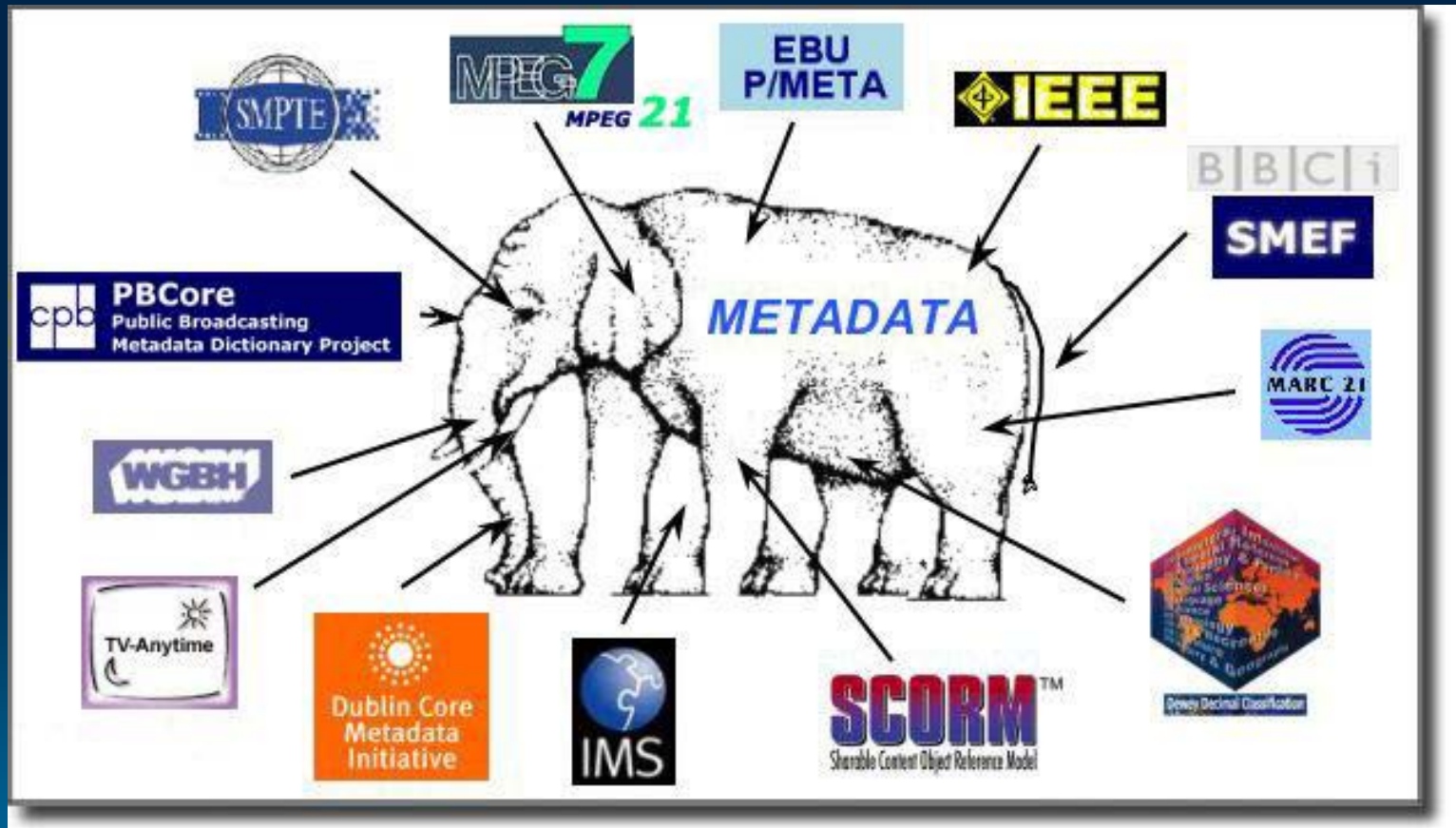
In my youth...two bad words

M and R words

Metadata and Reflection



Metadata



The Power of Metadata

Code is data, data is code. Everything is data. And data can drive behavior

"Anything you can do, I can do Meta"

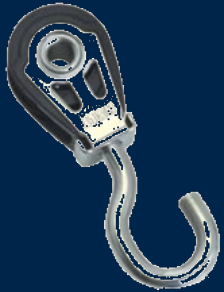
Meta data simply describes other data.

"If something is going to vary in a predictable way, store the description of the variation in a database so that it is easy to change"—Ralph Johnson

"Meta is Beta"

Dynamic Hook Points

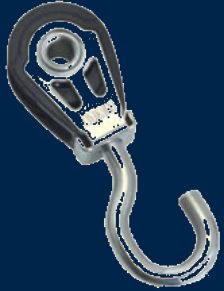
Eli Acherkan, Atzmon Hen-Tov, David H. Lorenz,
Lior Schachter, Rebecca Wirfs-Brock, Joseph W. Yoder
Asian PLoP 2012



When building dynamic systems, it is often the case that new behavior is needed which is not supported by the core architecture. One way to vary the behavior quickly is to provide well-defined variation points, called hook-points, in predefined places in the systems where new behavior can be dynamically looked up and invoke at runtime when desired.

Dynamic Hook Points

Eli Acherkan, Atzmon Hen-Tov, David H. Lorenz,
Lior Schachter, Rebecca Wirfs-Brock, Joseph W. Yoder
Asian PLoP 2012



Paving over the Wagon Trail



Patterns for Sustaining Architecture
PLoP 2012 Paper

ALIAS: MAKE REPETITIVE TASKS EASIER
STREAMLINING REPETITIVE CODE TASKS

Create simple examples, templates, & scripts

Develop a tool that generates code

Identify and use existing tools or frameworks

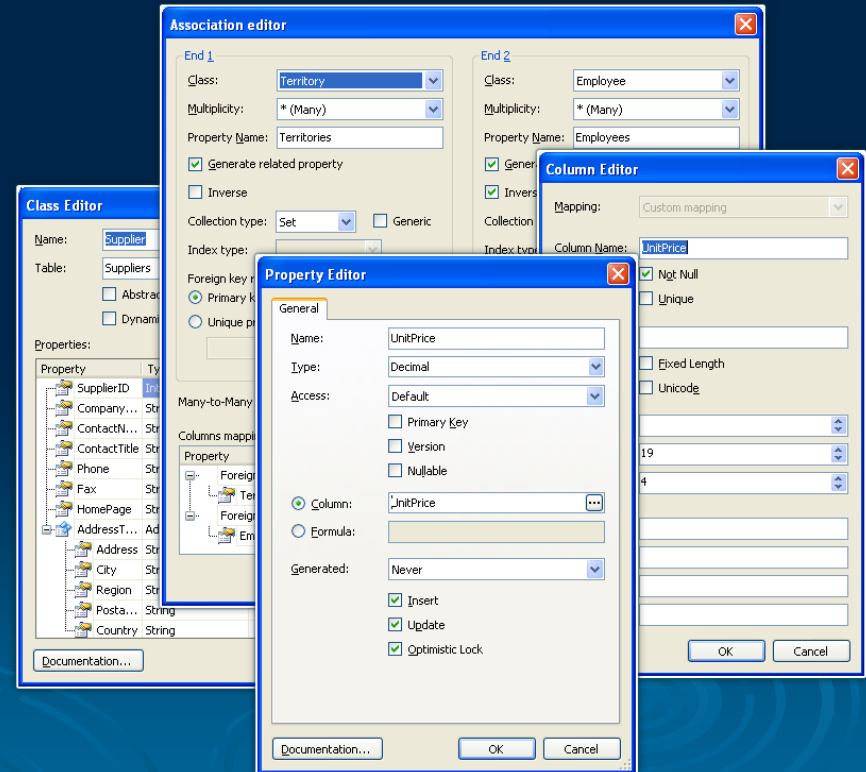
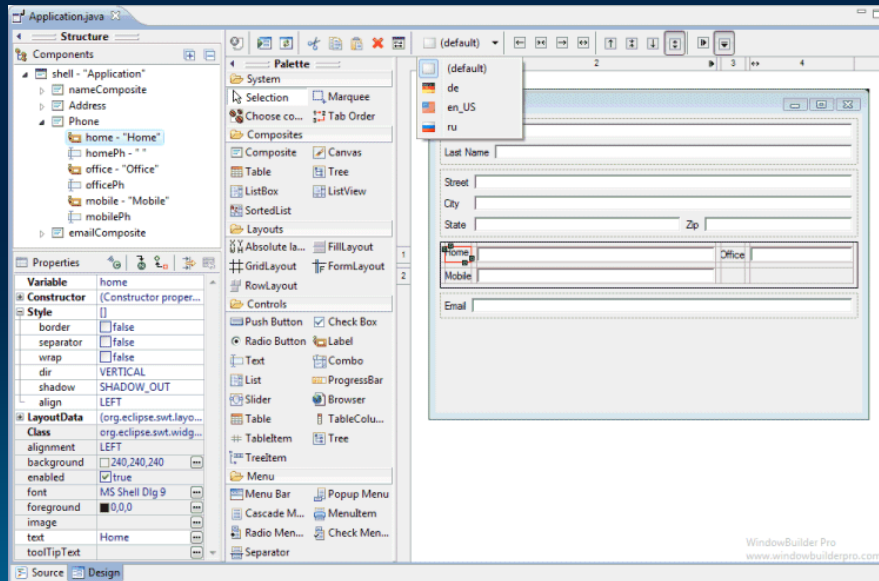
Develop a framework &/or runtime environment

Develop a domain-specific language

Paving over the Wagon Trail



Patterns for Sustaining Architecture
PloP 2012 Paper



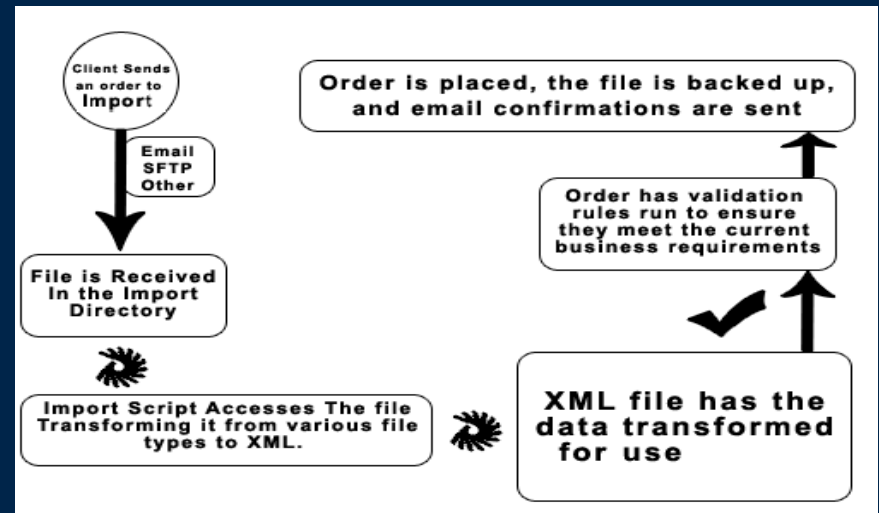
Sustaining Your Architecture

IMPORT EXAMPLE:

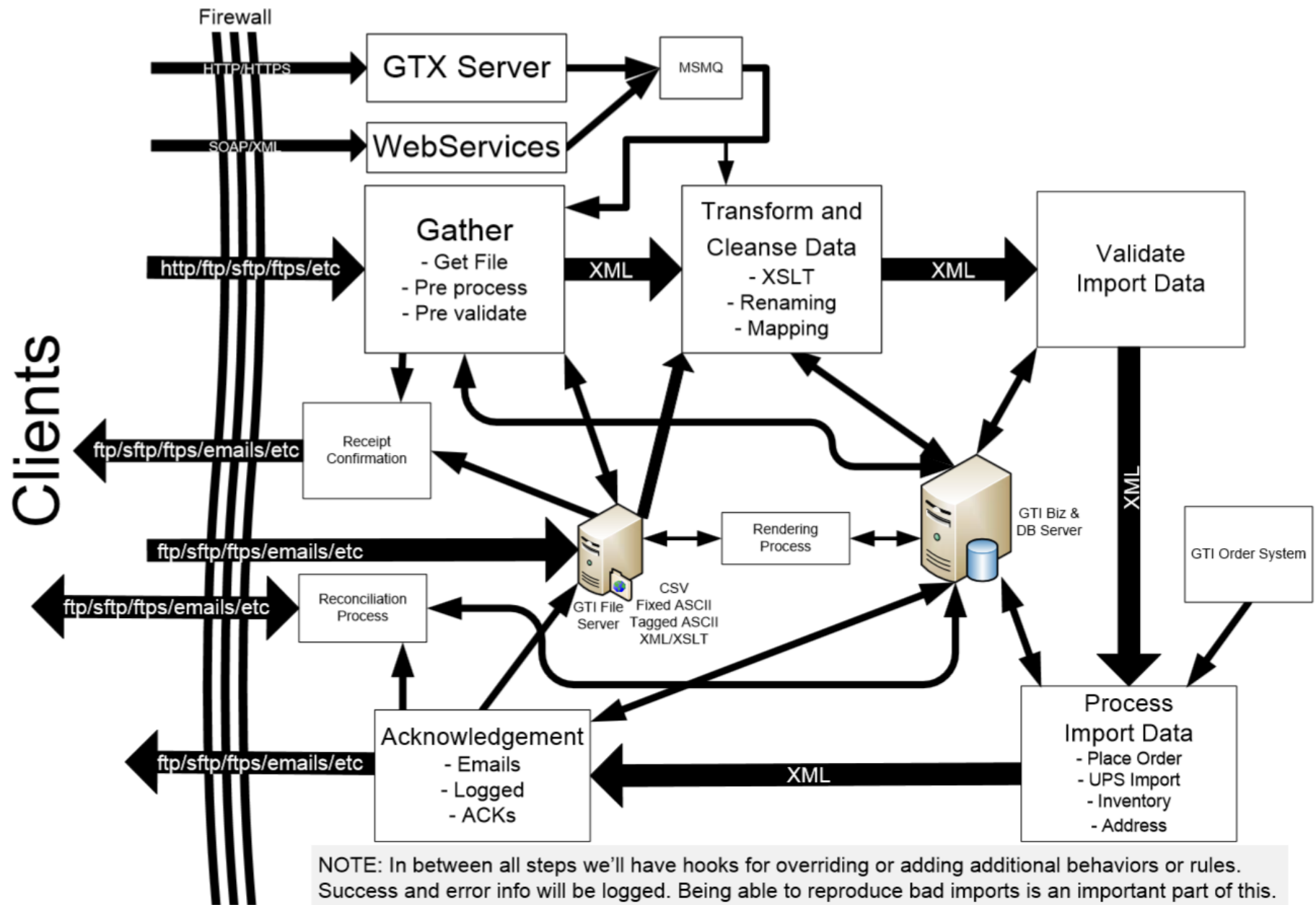
**HOW TO DEAL WITH
CLIENT VARIABILITY?**

Import Orders

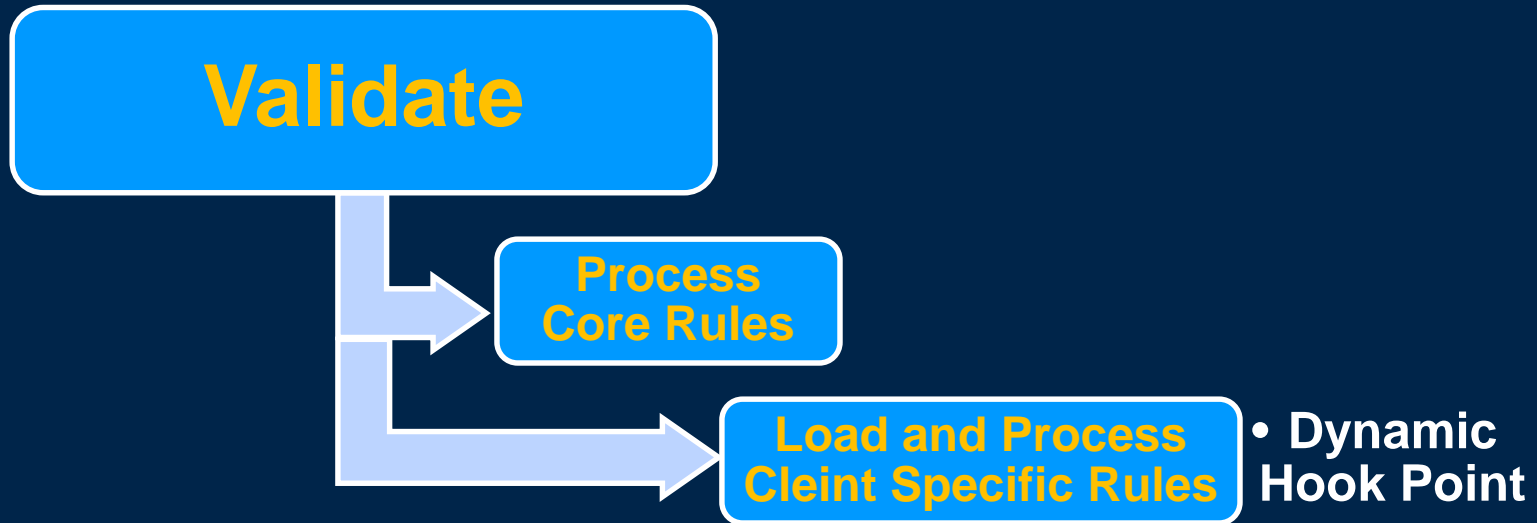
- Provided a standard format for imports
- Lots of duplication
- If Clients needed to vary from that format, we had to write or update a complete, self-contained custom import program
- Imports that need special processing, or had client specific rules, required to write a complete, self-contained import program



Gather/Import/Reconciliation Process Flow



Import Validation



```
[ValidationRule("Joes Validation Rule")]  
  
public class JoesValidationRule : ValidationRule {  
    public JoesValidationRule () : base  
    public override void Validate(ImportContext context)  
    ...}
```

Architectural Practice: Support Data Variability

- Separate what changes from stable part of system and provide support for adaptability/flexibility
- Integrate new learning into your architecture
 - Refactoring
 - Redesign
 - Rework
 - Code clean up



Sustaining an Architecture



- Minimize architectural debt: Support the ability to change/adapt what needs to change
- Make what is too difficult, time consuming, or tedious easier
- Decide at the most responsible moment, not the last possible moment
- Learn and evolve

Keep the system “livable” for its users and developers

Sustaining Your Architecture

Don't Pave the Cow Path



Patterns for Sustaining Architecture
PLoP 2012 Paper

*DON'T AUTOMATE JUST BECAUSE YOU CAN
BE CAREFUL HOW FAR YOU GO!!!*

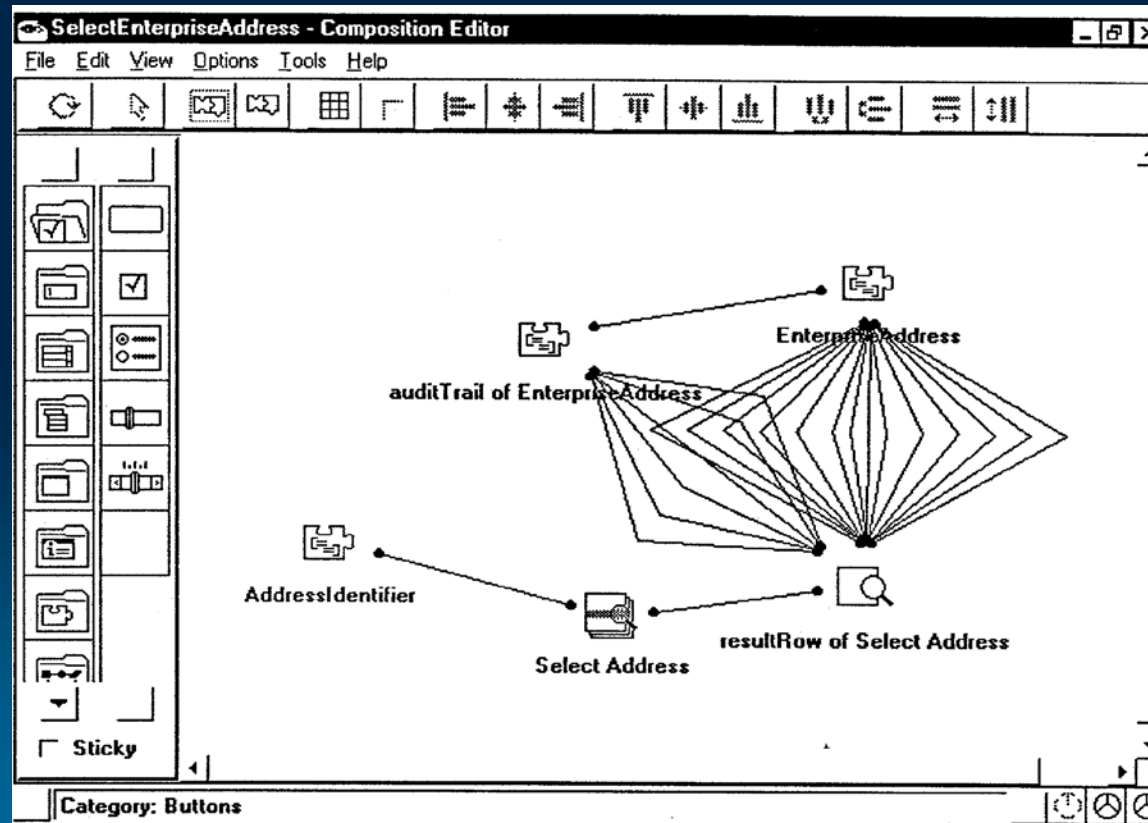
*ONLY ADD WHAT VARIABILITY YOU NEED WHEN
YOU NEED IT AT THE MOST RESPONSIBLE TIME*



Don't Pave the Cow Path



Patterns for Sustaining Architecture
PLoP 2012 Paper



Sustaining Your Architecture

Design Values

Supporting Variability

- Respect your system's shearing layers
 - Understand the rates of what changes
- Determine & support who should be able to make changes, when, and at what cost
- Make what is too difficult, time consuming, or tedious easier
 - Create tools, leverage design patterns, use data to drive behavior...
- Don't overdesign!!! (Only design what you need when you need it)



Resources

➤ Adaptive Object Models

- www.adaptiveobjectmodel.com
- www.metaplop.org
- Contact us for training or onsite consulting

➤ Agile Software

- Agile Alliance: www.agilealliance.org
- The Agile Manifesto
- 12 Principles of Agile Development

➤ Refactoring www.refactory.com



Thanks!!!



joe@refactory.com
Twitter: @metayoda