Architecture for Agile Provisioning of Financial Products and Services

Build products and services and deploy them as content in an appropriately architected system

14 April 2014

IDIOM

Agility

- “Agility is a core competency”

  “Companies must be flexible enough to customise their products [and services] for customers who expect special treatment, but they must also provide that special treatment from a common, standardized supply chain and business infrastructure” **

- Agility and change

  How quickly you can adapt to change = agility

  How quickly you can drive change into the market = agility

  Change arises from leadership – your own or someone else’s!

- Product/Service is the nexus of the Customer/Company relationship

  Company offers benefits under defined conditions

  Customer subscribes to benefits under agreed conditions

- Agility in provisioning Products and Services is now the enterprise “front-line”

** http://www.pwc.com/us/en/operations-management/assets/agility_foundation_for_change.html

Standardised Processes

- "With a strategic combination of standardization and flexibility, companies can more efficiently fulfil their promises to their customers"**
- Standardised processes – enabler, constraint, or both?
  - Standardised product definitions inhibit agility
  - Reusing standardised generic processes aids agility
- Processes <u>can</u> be standardised without compromising product agility
  - To help product agility, standardised processes must be 100% product or service agnostic, requiring zero knowledge of the internal product structure
  - The Product or Service becomes CONTENT in a standardised system
- Benefits of "Product-As-Content"/"Service-As-Content"
  - <u>Strategic Alignment</u>: 100% alignment between business policies and systems
  - <u>Agility</u>: Rapid product and service innovation to meet new market conditions
  - <u>New Options</u>: Economic development of tailored products and services for finer grained, niche markets, or limited issue customised "Partner Products/Services"
  - <u>ROI</u>: Reduce product/service development cost, time, and risk

** http://www.pwc.com/us/en/operations-management/assets/agility_foundation_for_change.html

Financial Products and Services are Agile ‘Content’ in a Standardised System

Build a financial product or service ‘factory’ for agile product development in a standardised framework

Overview

- Empower business owners
  - Enable independent “Product/Service” development by SME “Product Owners”
  - Products and Services are able to be modelled, built, tested, and deployed as discrete content within existing systems
- Change in IT Focus
  - Deliver/maintain highly engineered capabilities for use by products and services
  - Operate and manage products/services as content in suitably architected environment
- Separation of responsibility
  - Business defines + manages products/services within available system capabilities
  - IT provides capabilities to support product and service requirements
- Product or Service structure, details, and behaviour can be managed using the IDIOM tools
  - Details and behavior are controlled by IDIOM Decision Models
  - IDIOM Forms and IDIOM Decision Models manage user interaction
  - IDIOM Workbench is used for product testing and simulation on a large scale

What is a Product or Service

- "Product" and "Service" are broad terms implying a predefined process that is initiated by a specific customer request (the context) and closed by vendor acceptance and pricing
- Requests are recorded as complete instances of the Product or Service and are managed within the Product/Service business policy defined life-cycle
- Includes anything that can be sold, supplied, or allowed based on context specific information including for example:
  - Insurance products (e.g. a policy)
  - Lending products (e.g. a loan)
  - Entitlements and program admissions (e.g. a benefit entitlement)
  - Claims (e.g. claiming and entitlement from an insurance policy)
- Insurance Products are generally used as examples in this document – replace with 'loan' 'benefit entitlement' 'claim' etc. as appropriate to your business. Please read Product as implying Product or Service throughout.

What does a Product look like

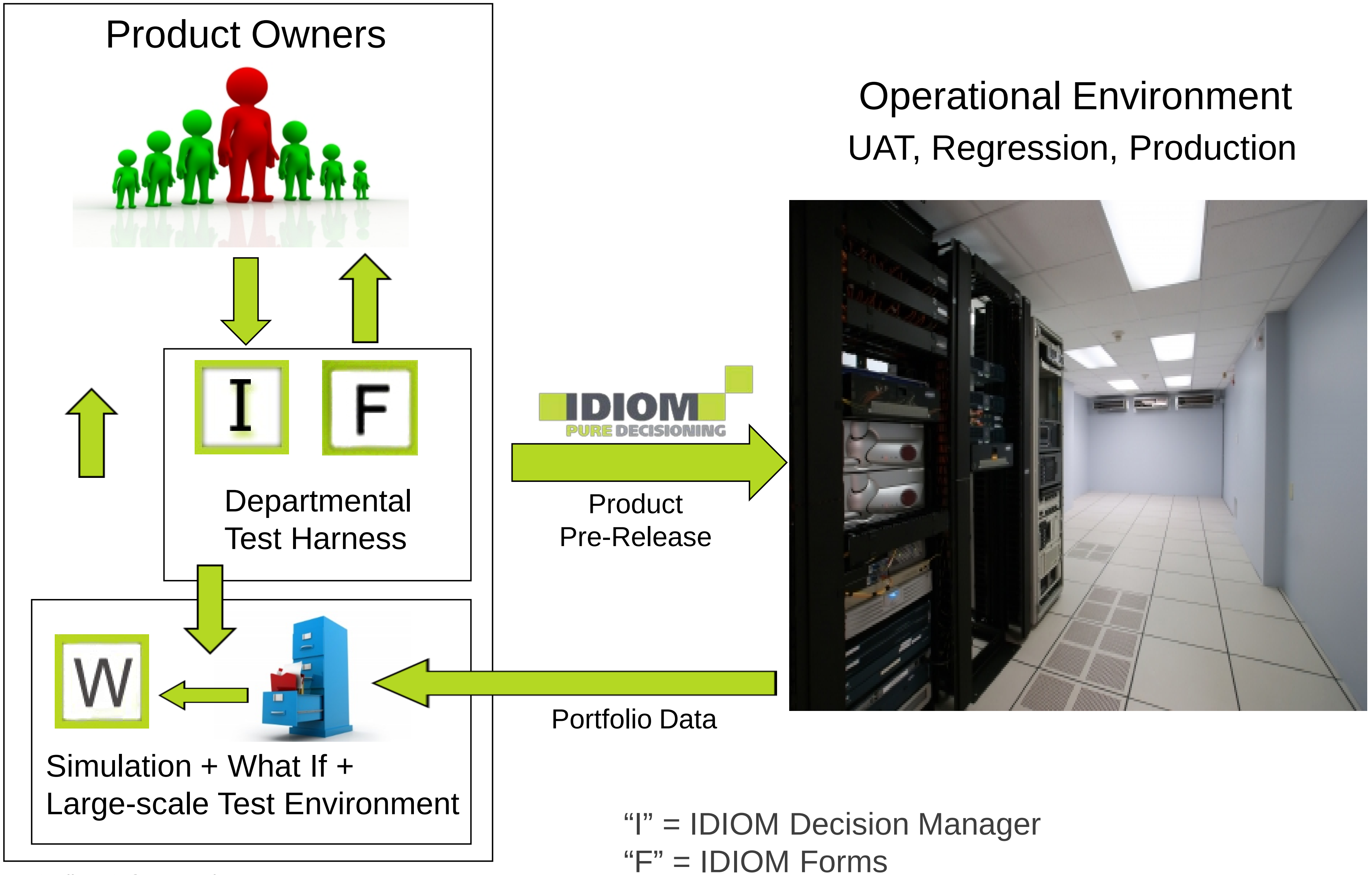
- A set of business policies (<u>rules</u>) that define:
  - What is being offered by way of the product (benefits)
  - Conditions under which it will accept requests for the product (underwriting)
  - Associated costs and terms (rating)
  - Future life-cycle events, and their associated conditions, costs and terms (behaviour)
- <u>Context data</u>: a collection of <u>factors</u> that will be used by the <u>rules</u> to determine the business policy outcomes as above
  - Assume the meta data format is the xml schema (xsd) (factor definitions)
  - Assume the actual data is an xsd compliant xml document (factor values)
  - Often 100s of factor definitions, many 1,000s of factor values per product instance
- Offer and acceptance imply <u>pre and post processes</u> for:
  - Data capture, validation, aggregation, enrichment, and transformation
  - Mappings to/from a range of internal and external systems
- Plus a <u>set of interfaces</u> to allow interaction with the external environment to collect and maintain the factors

Product Development Life Cycle - PDLC

A new, propitious “PDLC” for development and deployment of Products under SME control

IDIOM

Product Owners Use IDIOM Tools to Develop, Prove and Implement Product Strategies


Product Owners
I
F
Departmental Test Harness
W
Simulation + What If + Large-scale Test Environment
IDIOM PURE DECISIONING
Product Pre-Release
Portfolio Data
Operational Environment
UAT, Regression, Production

"I" = IDIOM Decision Manager
"F" = IDIOM Forms
"W" = IDIOM Decision Manager Workbench

PDLC – develop new Products

- Product Owner/SMEs develop

  Definition of Factors – by updating the Schema

  All rules – by developing Decision Models

- . . . and test the Product's underlying business policies

  Use the Workbench for full scale product simulation and testing

  Adjust and refine policies to ensure business objectives are met

- . . . before configuring Forms for End User interaction

  Define forms and forms behaviour – by developing IDIOM Forms and UI 'session' Decision Models respectively

- Then test and release to standard deployment process

  When complete, consistent, and correct, transfer to standard IT system test and UAT processes

PDLC – also use Product Configurations

- Build schema, rules, forms to manage product configurations (parameters)
  - Entire sub system for use by Product Owners
  - Product Owners explicitly build rules to respond to these configurations
- Product configuration ‘Forms’ are used to input and manage product parameters
  - Used by Product Owners, effective dated, and stored as ‘ProductConfigXML’
- Execution of Product rules always accompanied by Product Configuration document
  - Rules read ProductConfigXML document to access product parameters at runtime
  - E.g. rates, calculation methods, boundary conditions, etc

Technology Implications

- Product authors and IT infrastructure align around common Schema defined data structures
  - Provides the data ‘sand-pit’ for product development
  - The full scope of data must be instantiated whenever the rules run
- One common schema/set of schemas e.g. for insurance:
  - Common policy details, perils, risks, terms etc
  - Hierarchy of risk elements to cover increasingly specific risk data
  - Each shared schema element to be matched with shared rules for reuse
- Store complete XML document as part of core DB design for data agility and complete auditability
- Use rules to dynamically map agile <u>schema defined data</u> to known standard elements that then map to fixed <u>database columns</u>
  - Targets includes both external (incl 3rd party) and internal systems and databases
  - Extract and process these elements with standard processes – they do not change unless the target system changes

Solution Architecture - Overview

Product as content, and the associated PDLC, both assume the existence of a generic application that is not 'product aware'

Key Assumptions

- The ‘product’ will deploy as a self-contained ‘Product Engine’, which conceptually sits inside a customer specific generic application
- The customer specific generic application:
  - Provides all generic (standardised) application capabilities
  - Requires no knowledge of the internal content of either the Product meta data or context data in order to operate (<u>except</u> where the meta data or context data is targeted directly for consumption by the application)
  - Connects events and their context data with the correct rules for processing
  - Directs the results of rules execution to all related systems to apply as appropriate in order to align with the final state of the context data
- Given agreement to an appropriate set of interfaces, the ‘product engine’ can be inserted into any application (including legacy), or operated as a service from a cloud environment or similar

Features of the Customer Application (1)

- A generic application to provide capabilities for:
  - Persistence
  - Invocation of rules
  - Invocation of forms, human interfaces (if applicable)
  - Internal function invocation
  - Integration with other systems and interfaces
  - Polling service to generate time driven events
- User session management (if applicable)
  - Authentication, authorisation
  - Role based access to generic functions plus ‘Product instance’ search+select
  - ‘Product instance’ search+select results in product specific actions
- External events are recognised by ‘hard-coded’ application response
  - E.g. user selects product specific action; message arrives on queue; date/time reached, etc
  - Application acquires context data and hands-off to the Product Engine for rules +/or forms execution – without knowing anything about the product being processed

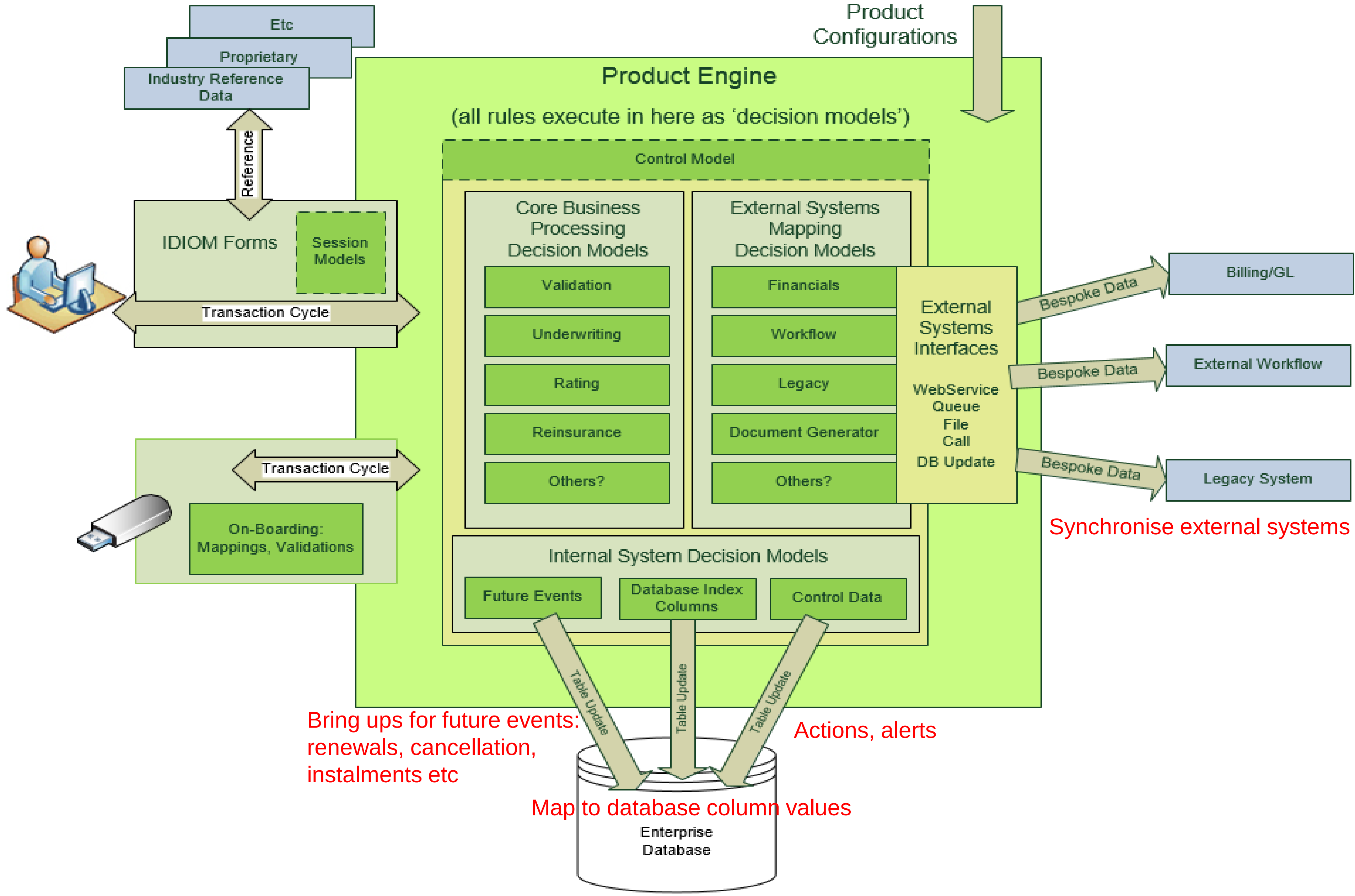
Features of the Customer Application (2)

- The Product Engine manages response in accordance with rules until rules determine the transaction is complete (a new valid state is reached)
- Completion of rules processing includes creation of new context data for:
  - Internal control and workflow (eg automated bring-ups, warnings, action lists)
  - Database column values for insert into standard (fixed) database columns
  - All relevant external system mappings (eg financials, legacy, workflow)
- When the Product Engine returns the completed transaction ‘XML context data’ to the application:
  - Database is updated: the context data is inserted as XML into one XML column; other column values are extracted from the XML and inserted natively
  - Control data (bring-ups, warnings, action lists etc) are extracted and cleared and replaced in the database for this entity
  - All external systems data is extracted and posted via respective integration components
- Result:
  - XML column holds <u>current state of the product instance</u> in 1 XML document
  - All internal and external systems are synchronised with this new state

User Transactions - Summary

- Native user session management for:
  - Authentication, authorisation
  - Role based access to non-product specific functions incl product search/select
  - Product search and select
- User selects non specific product action
  - Application performs the action eg admin, activity lists, print queues, etc
- User selects “New” product instance
  - Hand-off to Product Engine for rules controlled instantiation
- User selects existing product instance
  - Present rules driven list of valid actions
  - Actions may be generic (print, send, etc) or product specific (change, renew, cancel, etc)
  - All product specific actions are handed-off to the Product Engine

‘Product Engine’ Concept Overview



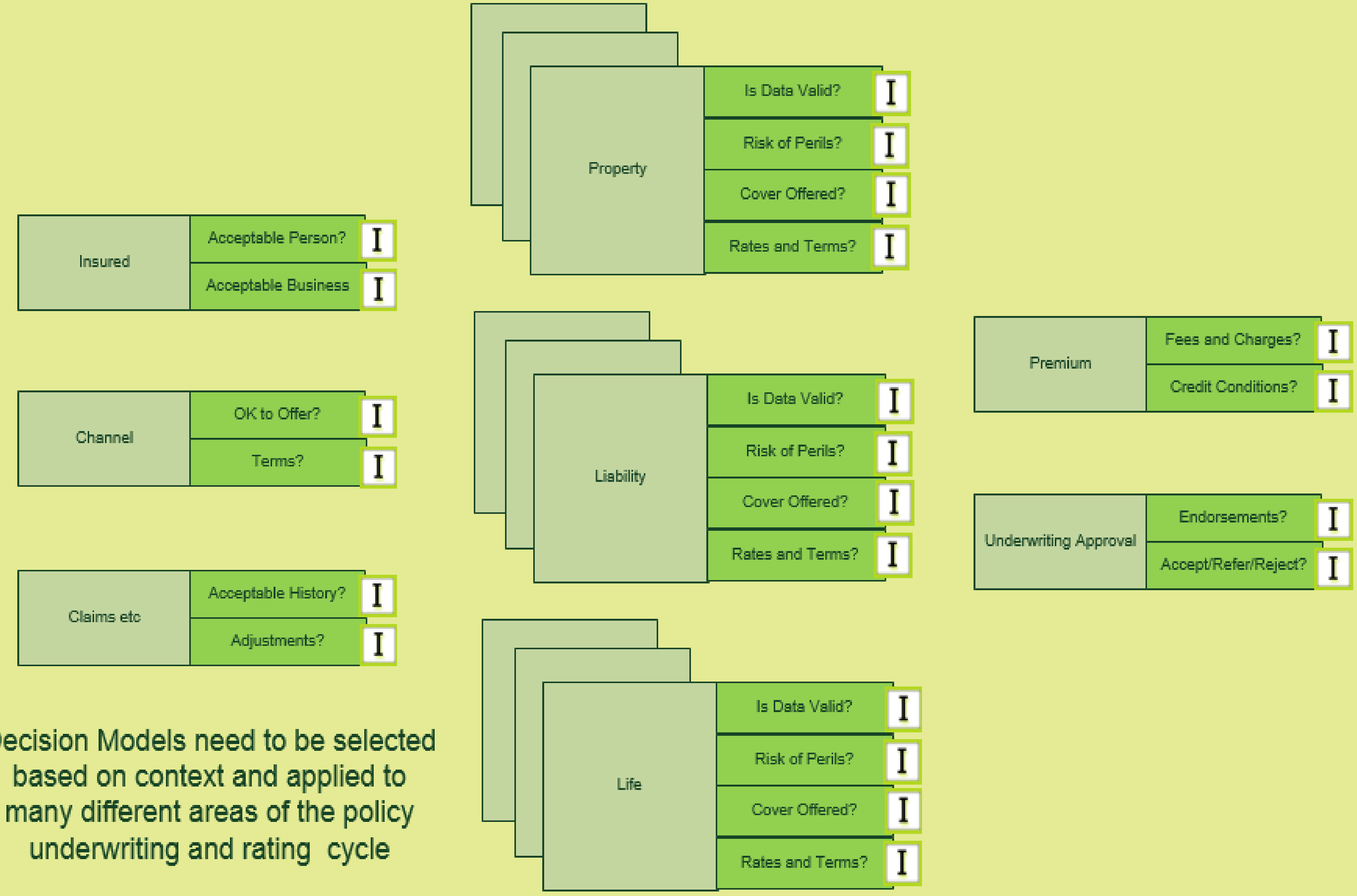
Synchronise external systems

Bring ups for future events: renewals, cancellation, instalments etc

Actions, alerts

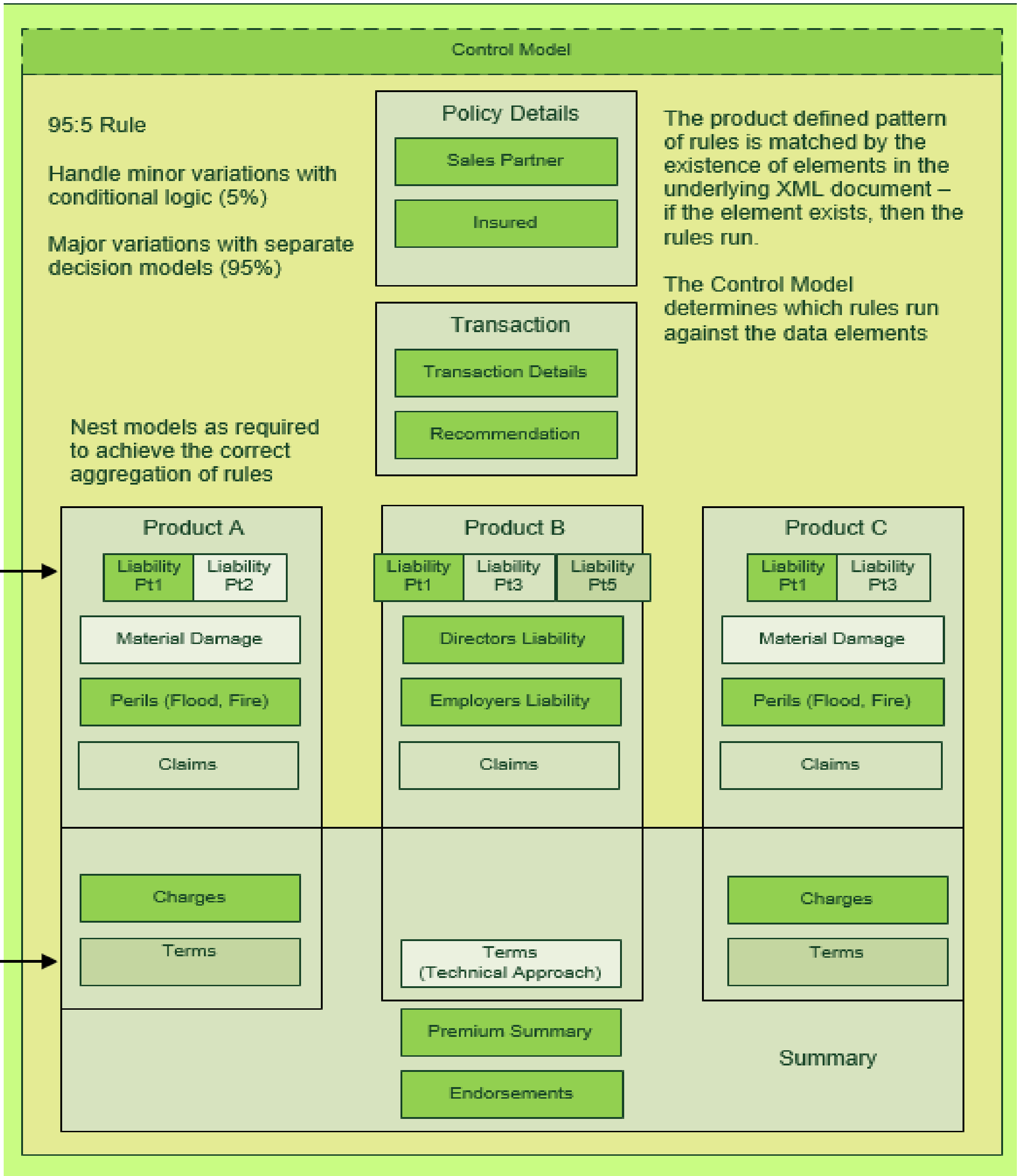
Map to database column values

Rules Apply throughout the Transaction Cycle



Decision Models need to be selected based on context and applied to many different areas of the policy underwriting and rating cycle

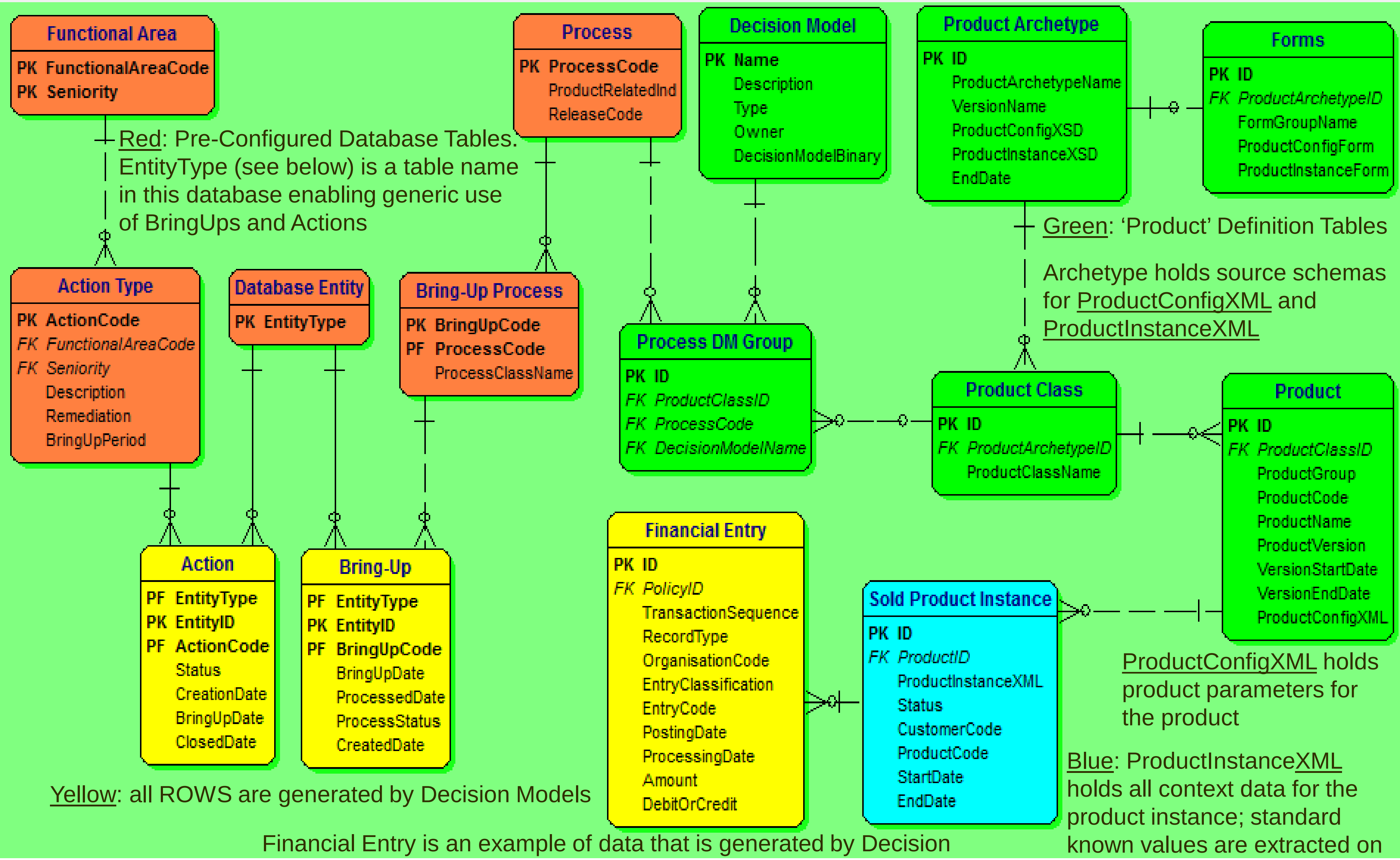
Sample Pattern for Core Business Models


Control Model
95:5 Rule
Handle minor variations with conditional logic (5%)
Major variations with separate decision models (95%)
Policy Details
Sales Partner
Insured
The product defined pattern of rules is matched by the existence of elements in the underlying XML document – if the element exists, then the rules run.
The Control Model determines which rules run against the data elements
Transaction
Transaction Details
Recommendation
Nest models as required to achieve the correct aggregation of rules
Product A
Liability Pt1
Liability Pt2
Material Damage
Perils (Flood, Fire)
Claims
Charges
Terms
Product B
Liability Pt1
Liability Pt3
Liability Pt5
Directors Liability
Employers Liability
Claims
Terms (Technical Approach)
Product C
Liability Pt1
Liability Pt3
Material Damage
Perils (Flood, Fire)
Claims
Charges
Terms
Premium Summary
Endorsements
Summary

Control Model can dynamically aggregate rules components to service some risk types

Same function, different approach

Proforma 'Product Factory' Database Design



Red: Pre-Configured Database Tables. EntityType (see below) is a table name in this database enabling generic use of BringUps and Actions

Green: 'Product' Definition Tables

Archetype holds source schemas for ProductConfigXML and ProductInstanceXML

ProductConfigXML holds product parameters for the product

Yellow: all ROWS are generated by Decision Models

Blue: ProductInstanceXML holds all context data for the product instance; standard known values are extracted on 'Save' to populate the other column values

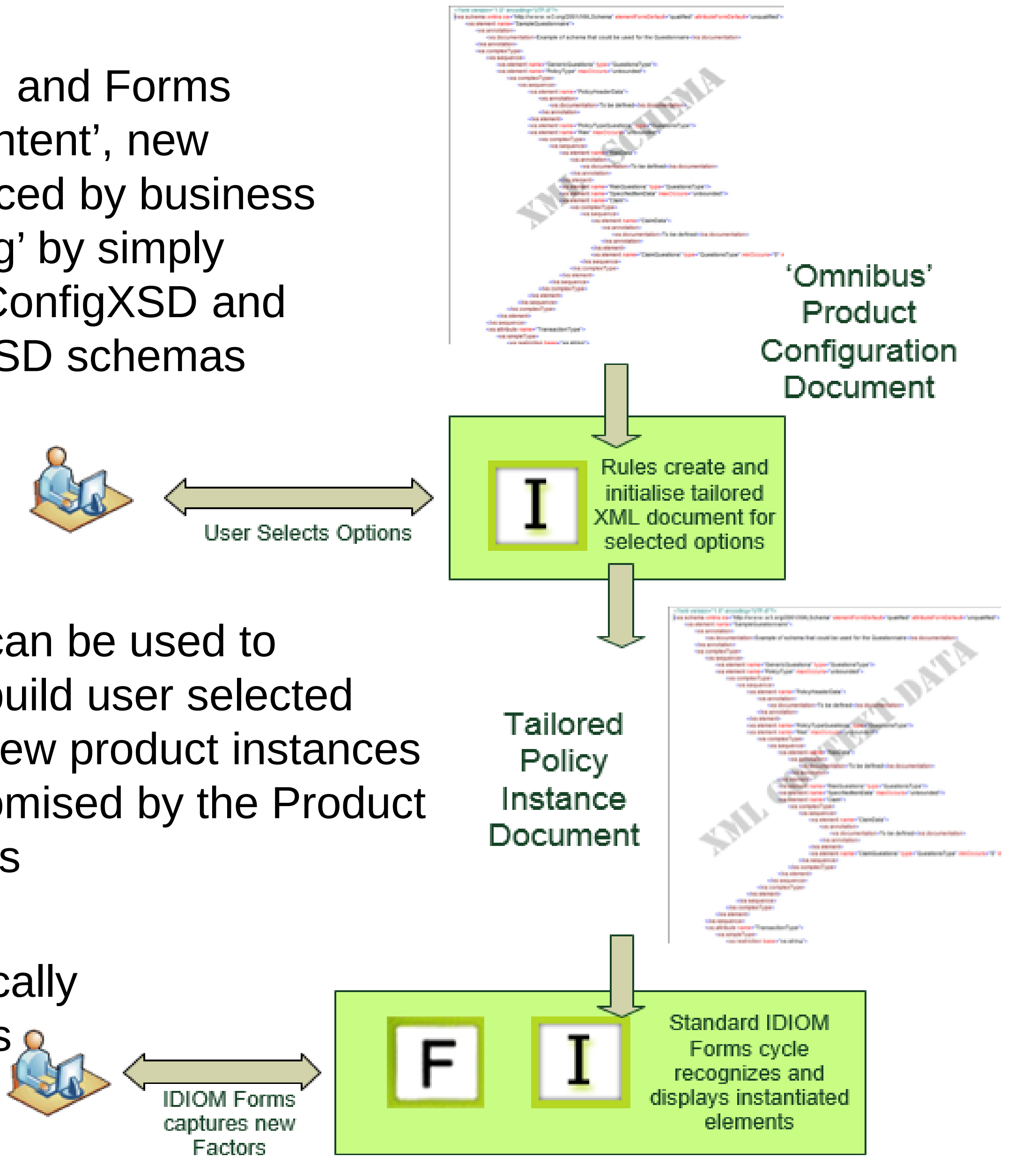
Financial Entry is an example of data that is generated by Decision Models specifically for a known external system (columns are bespoke to that system)

Dynamically Add New Context Data (Factors)

With Schemas, Rules, and Forms aligned as system 'content', new factors can be introduced by business owners 'without coding' by simply updating the ProductConfigXSD and the ProductInstanceXSD schemas

This feature can be used to dynamically build user selected options into new product instances that are customised by the Product Configurations

IDIOM Forms automatically recognizes and displays the new factors


XML SCHEMA
'Omnibus' Product Configuration Document
User Selects Options
I
Rules create and initialise tailored XML document for selected options
Tailored Policy Instance Document
XML CONTEXT DATA
IDIOM Forms captures new Factors
F
I
Standard IDIOM Forms cycle recognizes and displays instantiated elements

IDIOM Tools - Introduction

IDIOM Decision Manager

IDIOM Forms

IDIOM Decision Manager Workbench (DMW)

IDIOM Decision Manager

- IDIOM Decision Manager is a tool for graphically modeling and deploying business decisions - without programming!
- A tool for the policy maker, not the programmer
- IDIOM Decision Manager automates complex policy based decision-making at the enterprise level, deployable as industrial strength stand-alone components
- In day-to-day practice it is usually used by IDIOM trained analysts or SMEs working interactively with Product Owners.

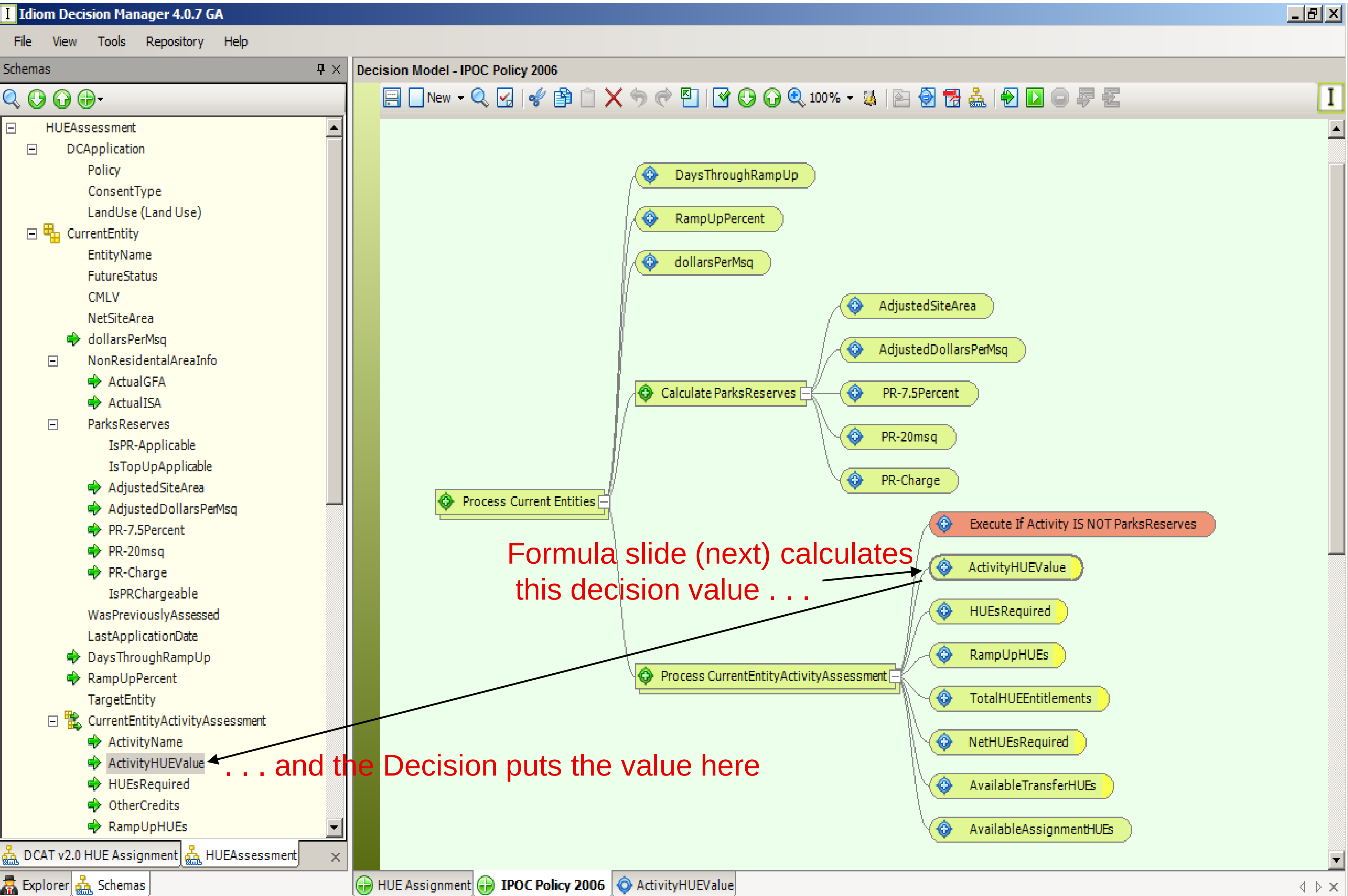
  Together they model the business/policy domain in terms of both data and decisions (see Decision Model slide:26) before moving on to define the underlying ‘Formula’ logic that binds them together (slide:27)
- Deployment as software components is fully automated and ‘without fingerprints’

IDIOM Decision Manager (Example)

(See next Slide)

- This example is a real model drawn from a City Council implementation of policy that calculates financial contributions to be paid by property developers
- The policy is decomposed using a ‘mind mapping’ approach until we reach the atomic units that we call decisions (rounded boxes)
- This ‘decision model’ is demonstrably aligned and integrated with the adjacent data model (left hand panel) - validating and strengthening both
- The atomic ‘decisions’ provide an easy entry point for specification of the underlying rule details via the Formulas

IDIOM Decision Manager (Decision Model)



Idiom Decision Manager 4.0.7 GA

File View Tools Repository Help

Schemas

- HUEAssessment
  - DCApplication
    - Policy
    - ConsentType
    - LandUse (Land Use)
  - CurrentEntity
    - EntityName
    - FutureStatus
    - CMLV
    - NetSiteArea
    - dollarsPerMsq
    - NonResidentalAreaInfo
      - ActualGFA
      - ActualISA
    - ParksReserves
      - IsPR-Applicable
      - IsTopUpApplicable
      - AdjustedSiteArea
      - AdjustedDollarsPerMsq
      - PR-7.5Percent
      - PR-20msq
      - PR-Charge
      - IsPRChargeable
    - WasPreviouslyAssessed
    - LastApplicationDate
    - DaysThroughRampUp
    - RampUpPercent
    - TargetEntity
    - CurrentEntityActivityAssessment
      - ActivityName
      - ActivityHUEValue
      - HUEsRequired
      - OtherCredits
      - RampUpHUEs

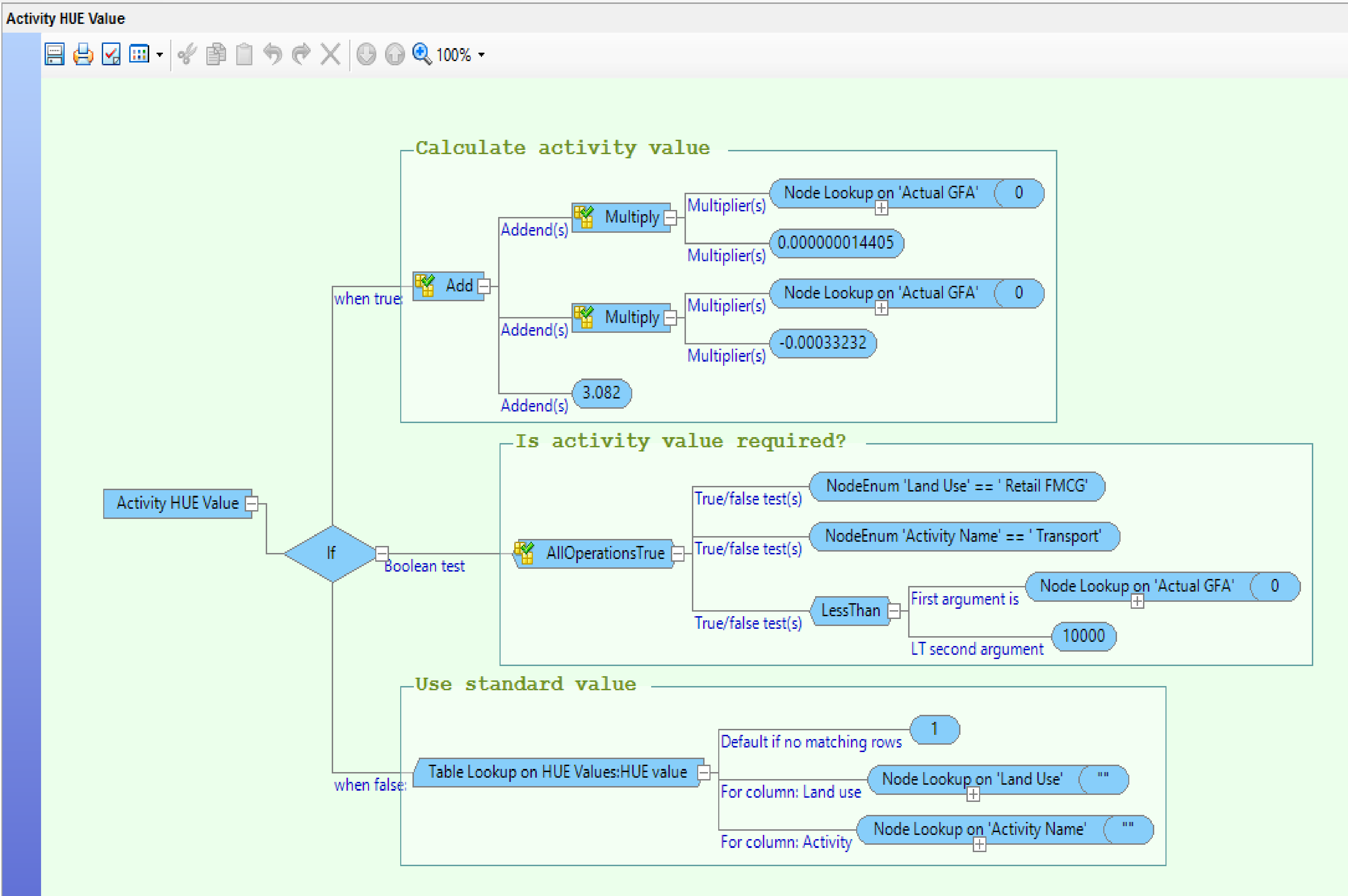
Decision Model - IPOC Policy 2006

- Process Current Entities
  - DaysThroughRampUp
  - RampUpPercent
  - dollarsPerMsq
  - Calculate ParksReserves
    - AdjustedSiteArea
    - AdjustedDollarsPerMsq
    - PR-7.5Percent
    - PR-20msq
    - PR-Charge
  - Process CurrentEntityActivityAssessment
    - Execute If Activity IS NOT ParksReserves
    - ActivityHUEValue
    - HUEsRequired
    - RampUpHUEs
    - TotalHUEEntitlements
    - NetHUEsRequired
    - AvailableTransferHUEs
    - AvailableAssignmentHUEs

Formula slide (next) calculates this decision value . . .

. . . and the Decision puts the value here

IDIOM Decision Manager (Formula Palette)



IDIOM Decision Manager (Formula Palette)

- The underlying rules details are easily captured using a ‘Lego’ like drag-and-drop development approach
  - ‘More fun than playing golf’ according to the CEO of one of our largest customers
  - There is no scripting or coding required to build these formulas
- The rules can be tested immediately within the IDIOM Decision Manager palettes
- When finished, IDIOM Decision Manager generates computer source code (C# or Java) with a single button click
  - Callable by any application at run-time using any of a wide variety of simple interfaces and wrappers (in-line, dll, web service, queue service, many more)
  - Can also be published directly into the IDIOM Decision Manager Workbench
- At the same time it generates the model into business readable documentation (PDF)

IDIOM Decision Manager – Key Points

- IDIOM's decision models do for policy decisions what data models do for data – a powerful abstraction that makes the underlying complexity visible and manageable
- The models allow internal data transformations and business rules to be intermingled within a single transaction

  Business rules acting alone are severely limited in their ability to fully implement business policy – invariably, in-line data transformations are necessary to match the terminology* used in the policy statements

- Decision models that incorporate both data and rules behaviour enable a further critical capability that is unique to IDIOM Decision Manager – the models can be fully tested using real-world test cases directly in the builder palettes

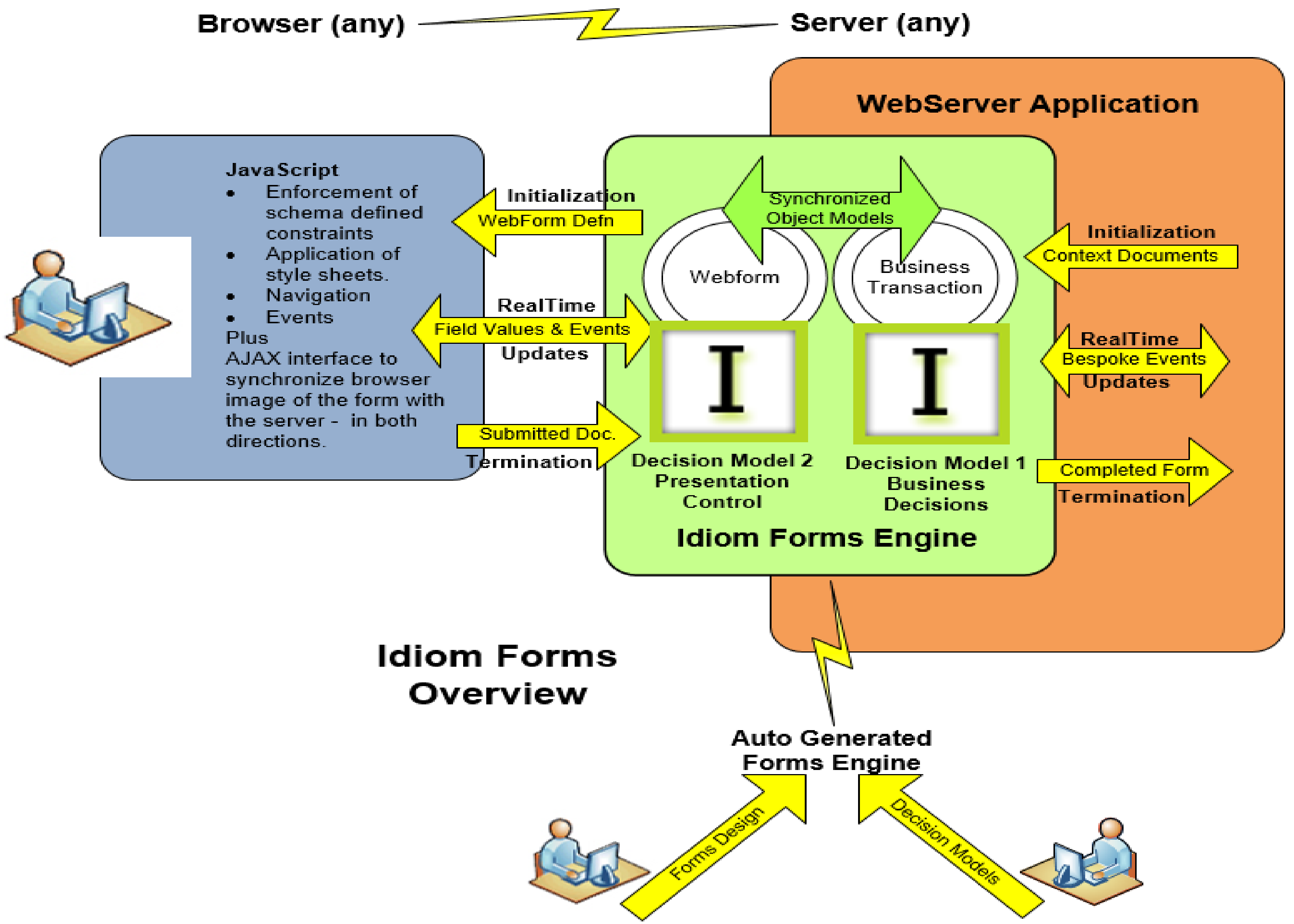
  No external technology or application support is required to empirically prove the correctness, completeness, and consistency of the models

- The decision models are converted into a form of 'logical English' and/or XML for complete transparency

  In addition to the C# or Java program source code that fully automates the models!

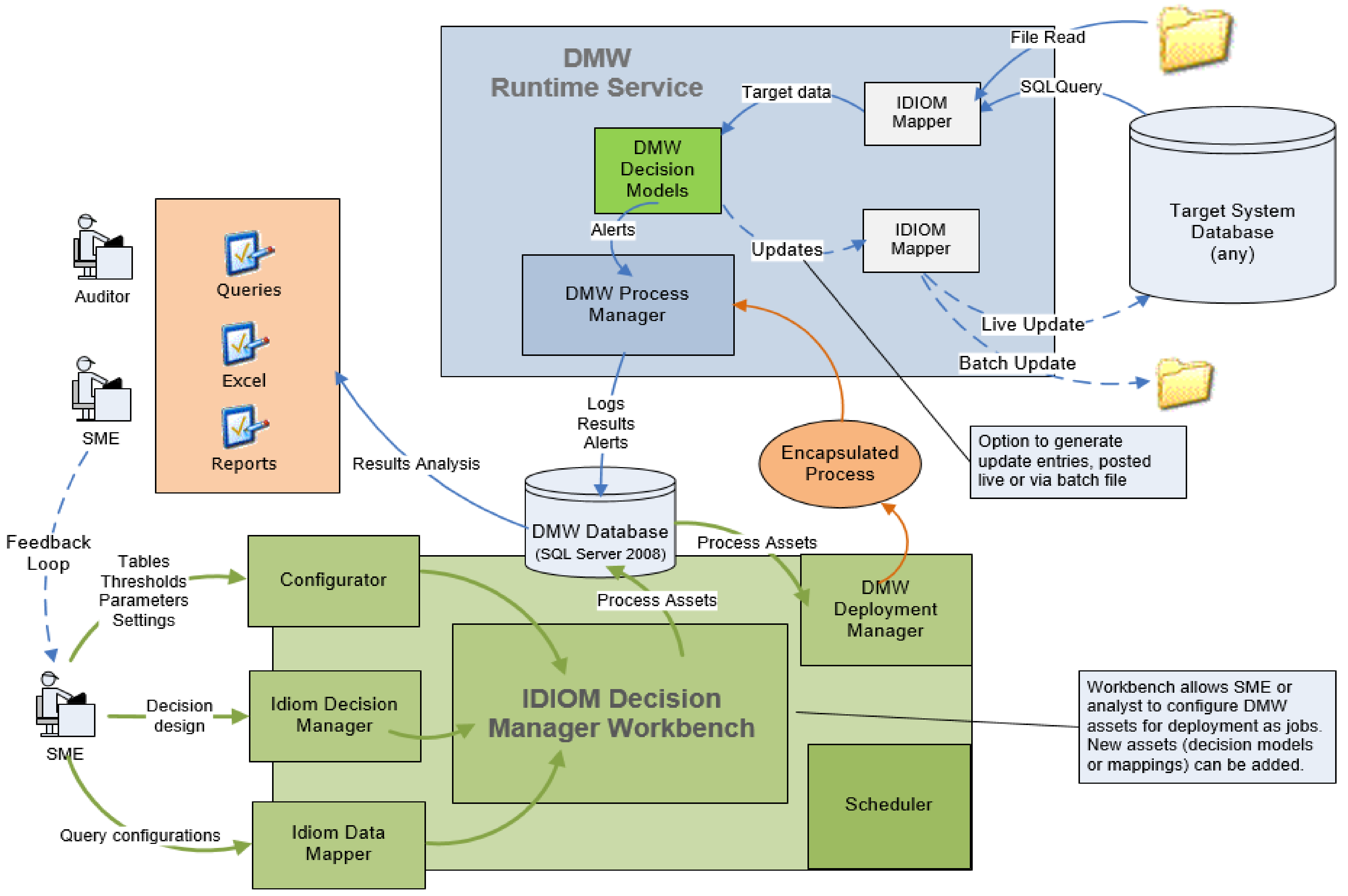
* Personalised terminology is an 'idiom'; hence the name of our company and product

IDIOM Forms



Browser (any)
Server (any)
WebServer Application
JavaScript
• Enforcement of schema defined constraints
• Application of style sheets.
• Navigation
• Events
Plus
AJAX interface to synchronize browser image of the form with the server - in both directions.
Initialization
WebForm Defn
RealTime
Field Values & Events
Updates
Submitted Doc.
Termination
Synchronized Object Models
Webform
Business Transaction
Decision Model 2 Presentation Control
Decision Model 1 Business Decisions
Idiom Forms Engine
Initialization
Context Documents
RealTime
Bespoke Events
Updates
Completed Form
Termination
Idiom Forms Overview
Auto Generated Forms Engine
Forms Design
Decision Models

IDIOM Decision Manager Workbench



Use IDIOM Decision Manager Workbench For:

- Routine testing for intended/unintended changes during development
  - Unintended consequences can be a major cost
  - Can run regression for every change, with expected outcomes masked
  - Run daily during development, plus comprehensive release testing
- Verify business policy changes
  - Execute new business policies (e.g. underwriting and rating, or claims) across existing portfolio
  - Use further decision models to assess outcomes and verify that the changes are beneficial and as planned
- Routine verifications and investigations
  - Develop models for portfolio investigation and reporting
- Full file pass to generate updates for new, low cost batch processes
- Production of masked test data from production sources
- Migration of data between unlike databases or versions thereof

Worked Example

Example Scenario: A Group Insurance Scheme Distributed via a Superannuation Fund to its Members

Scenario Outline

- Insurer provides basic insurance cover for all fund members
  - Say, basic death cover
  - Provided by auto processing existing member data in standard batch process
- Insurer offers an opt-in opportunity for members for more complex products
  - Insurer offers up-sell based on existing member data
  - Say, death/disability/trauma for all family members
  - Member opts-in and provides additional details for immediate cover, or follow-up referral if needed
- Opt-in for all Products using a single Form accessed via a ‘Member Portal’
  - Portal may be Fund, Administrator, or 3rd Party
  - Form is defined by the Insurer
  - Form includes Insurer’s validation, underwriting, rating rules tailored for the specific scheme/product instance

Assume Insurer has 'Standard' Capabilities (1)

All of these capabilities are reusable across all schemes (2 slides)

- A single XML Schema defines policy data for all scheme products
  - Includes standard insured, policy, risk, cover and financial elements
  - Includes standard workflow elements – bring-ups, warnings, actions
  - Includes standard elements for updating Insurer's back-end systems
  - Includes standard elements for updating Member Administration systems
- Standard mappings to/from industry systems (SQL <> XML)
  - In-bound member details are mapped to standard insurance elements
  - Insurance and workflow elements are mapped out to Insurer's internal systems
  - Member updates are mapped back to Member Admin system (e.g. Acurity, Sonata, Blue Door et al) IDIOM Mapper can be used for all mappings
- Standard business policy defined rules are built over the Schema
  - Insurance rules provide standard validation, underwriting, rating
  - Support rules generate financials, workflow, internal/external system elements
  - Session rules dynamically morph the Form for required behaviour

Assume Insurer has 'Standard' Capabilities (2)

- One umbrella Scheme management database for all Schemes

  The single XML Document describing insurance policy data is stored extant

  Workflow elements are extracted and put in tables for Insurer workflow

  Scheme Fund details, deployment details, batch transfer details et al are standard table data in this database for operational management of schemes

- Product Configuration

  Product configuration document (per product) contains parameters for validations, underwriting, rating (e.g. rates, allowable ages, calculation methods)

  All decision models understand and use the product configuration document

  Most <u>versions</u> of products, and many <u>new products</u>, can be created by simply cloning and updating an instance of the Product Configuration document

- Two IDIOM Forms

  One Form is to maintain the Product Configurations (Insurer use only)

  One Form is to access all of the Policy details (Insurer, Fund, Members???)

  The Policy Form morphs dynamically for individual scheme and user combinations under the control of decision models

Process: Negotiate + Configure a New Scheme

- Can the new Scheme be simply configured via a Product Configuration
  - Yes: Clone, update, and set up Scheme operational data. Finish! (hours)
- NO: does it fit within the data available in the existing schema?
  - No: extend schema with new standard and/or scheme specific elements
- Continue: adjust rules for scheme
  - Minor variations simply added to existing decision models (hours)
  - Build specific decision models for more substantial variations (hours to days)
  - Adjust the 'control model' to include any new decision models (minutes)
- Update the existing Form
  - Only elements used by <u>this</u> scheme will appear in the Form – automatically
  - Provide new style-sheet if new styling required to align with Member Portal
  - If entirely new look+feel needed, clone and adjust Form or rebuild (days)
- Set up Scheme operational data. Finish!

Test and Deploy

- Insurer testing using IDIOM Decision Manager Workbench
  - Fund uses Workbench to copy and mask member test data for Insurer
  - Insurer uses Workbench for full insurance life cycle testing of Batch processes
  - Insurer uses Workbench for full insurance life cycle testing of Forms processes
- Test deploy the Form to Insurer test harness to test visuals + behaviour
- Deploy batch process with correct mappings and decision models
  - Fund Administrator to run batch processes stand-alone, or embed in daily/monthly process, or run via IDIOM Workbench
  - All rules now specific to the scheme but most simply reused
  - Extract and transfer generated elements back to Insurer (backend) and Member Admin database by preferred method
- Deploy Form and link to external websites
  - Deploy Form in a frame within existing Portal and redirect back to insurer system
  - Or, generate and embed within the Member Portal (not connected to Insurer)
- System test and go live!

Summary

This architecture works – today there are <u>millions of on-risk insurance policies</u> being managed in the architecture as presented

Business Benefits

- 100% alignment of strategic business policies and computer systems
- Business fully controls business policy and product/service development and deployment
- Much more efficient rules development; maximum rules reuse once developed
- Local ‘simulation workbench’ for testing products/services and their updates to assist product and business policy development
- Product changes verified against existing portfolio before release – no surprises
- And of course, maximum product agility in a standardised framework as promised

Thank You

Mark Norton

mark.norton@idiomsoftware.com

+64 21 434669