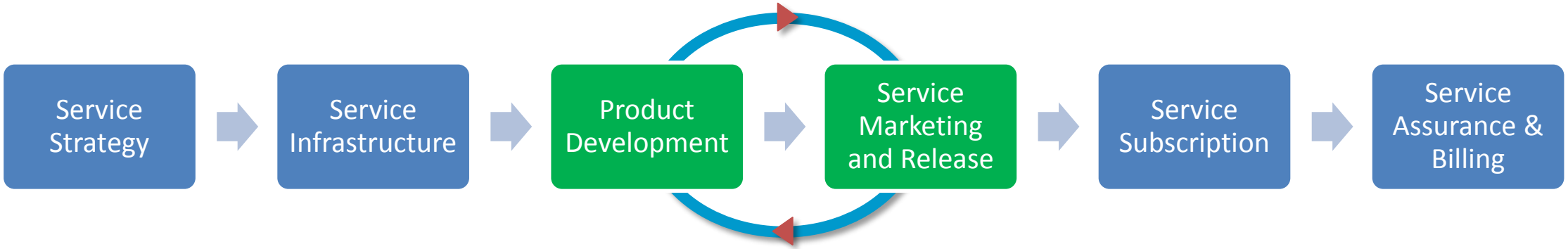
SaaS Engineering

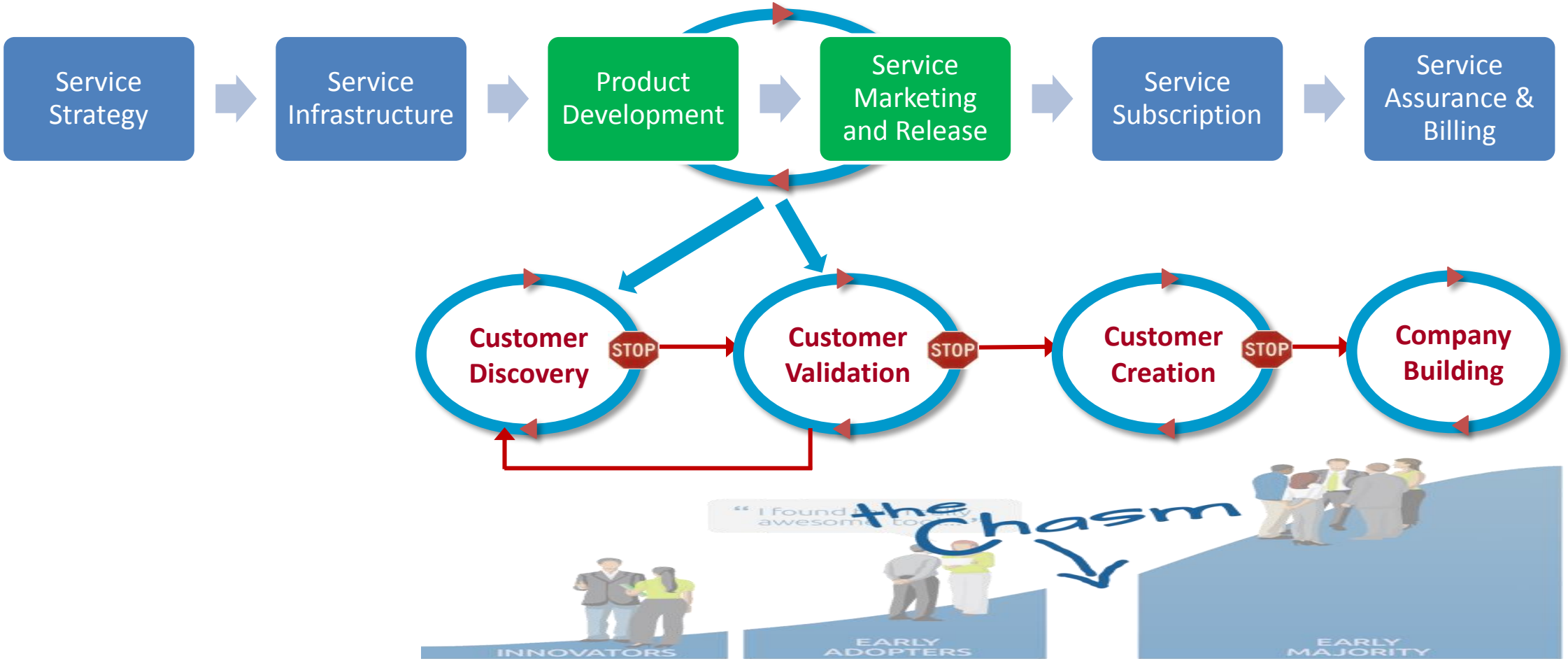
Professor June Sung Park http://flavors.me/june_sung_park
KAIST

Table of Contents

Value Chain of SaaS Business



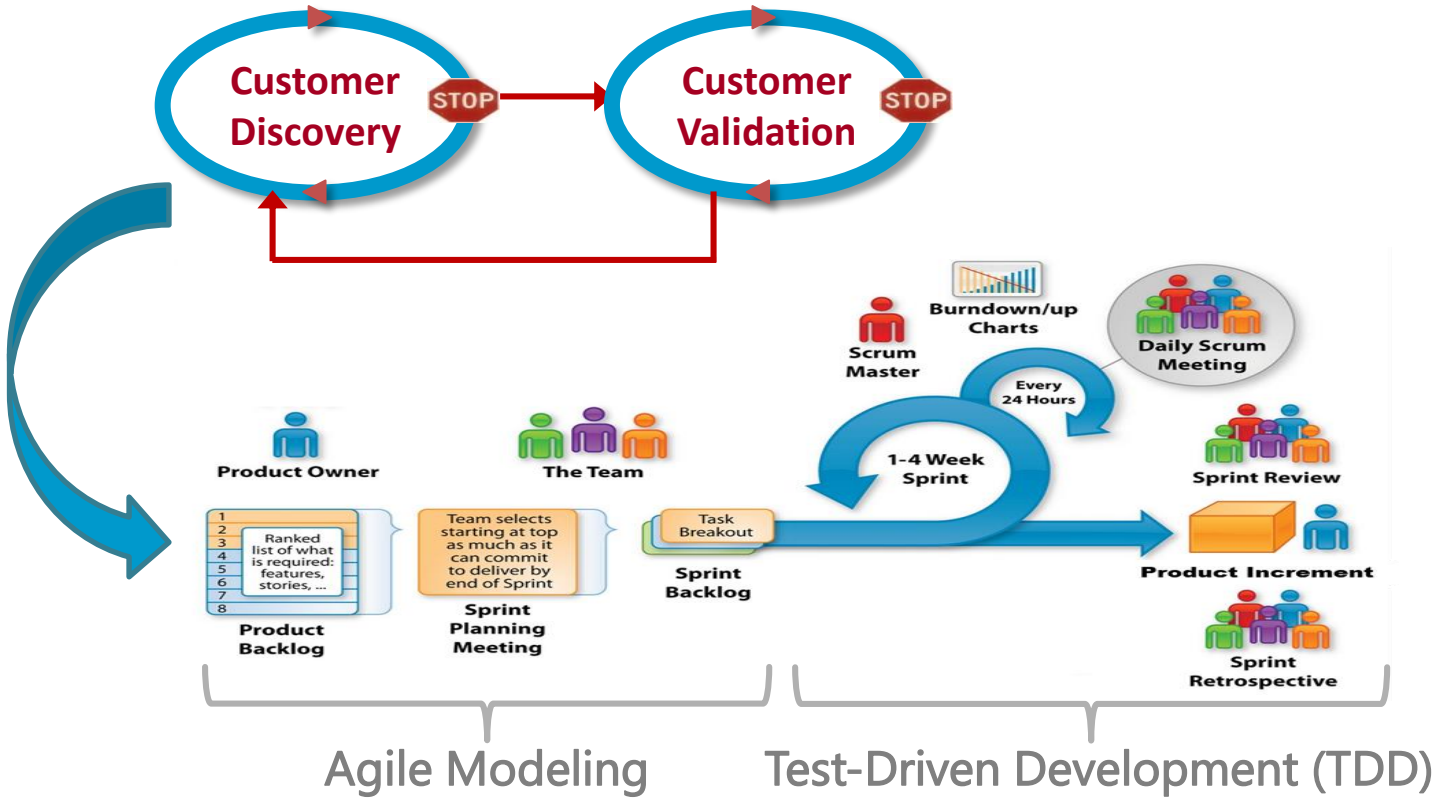
SaaS Customer Development Process



Steven Blank, The Four Steps to the Epiphany-Successful Strategies for Products that Win, 2006.

SaaS Software Development Process

- Customer Development Process requires Agile Software Development Process.
- Agile Software Development requires Agile Requirement Modeling.



Don Wells, Agile Software Development: A Gentle Introduction (http://www.agile-process.org/)

Jeff Sutherland and Ken Schwaber, The Scrum Guide (http://www.scrumguides.org/)

Scott Ambler, Agile Modeling (http://www.agilemodeling.com/)

Evolution of Application Development Technology

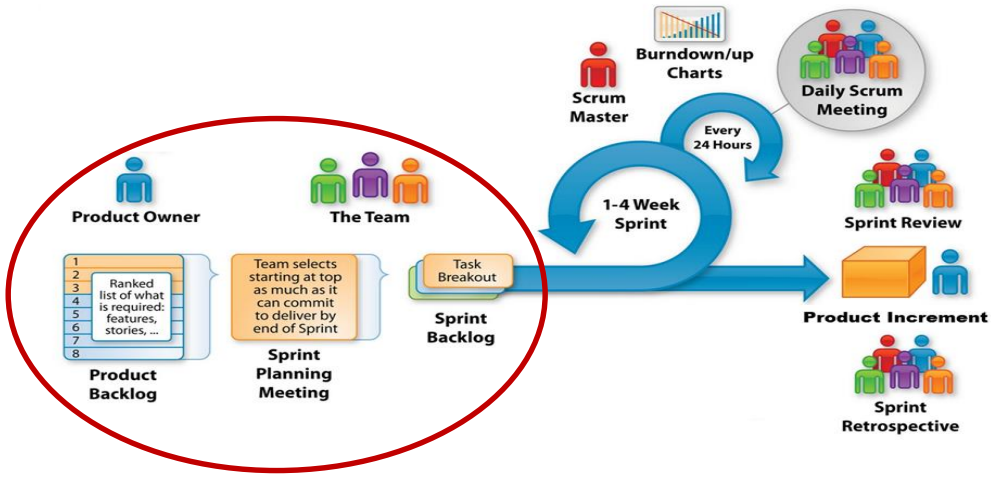
Gartner, IT Market Clock for Application Development, 2013

Evolution of Application Development Method

Asset Class	Recommendations
Architected Model-Driven (AMD) Service-Oriented Design of Applications (SODA)	Slow evolution, as shared business and IT modeling standards that can be used for 100% code generation evolve. Applicable now for early technology adopters, especially those with CASE/code generator experience.
Enterprise-Class Agile Development Methodologies	Look to move from tactical agile development to enterprise-class agile to gain the benefits of long term and sustainable agility.
Cloud-Native Application Design	IT leaders should establish guidelines that define when and where the principles of cloud-native application design should be applied, based on cost, risk and time factors applicable to a given cloud solution delivery initiative.
Business Process Analysis (BPA) Tools	Stable development path. Use business process analysis (BPA) tools to support enterprise architecture/business process management (BPM) initiatives, and as a starting point for architected, model-driven (AMD) AD and integration projects.
Architected Rapid Application Development (ARAD) SODA	Stable development path. To reduce the learning curve and increase the productivity of Java EE or .NET developers or the transition of traditional client/server developers to service-oriented development of applications, consider providing these developers with agile methods and ARAD tools.
Agile Development Methodologies	Stable deployment path. Adopt agile approaches judiciously and on projects with sophisticated teams that are not averse to process discipline.
Composite Applications and Enterprise Mashups	Use service oriented architecture (SOA) to address complexity and governance issues. Enable SOA by using integration, process management and multiprotocol communication technologies, via best-of-breed products or an integrated middleware suite.
Mobile Application Development (AD)	Select tools based on your needs during the next two to five years, but set internal expectations that the choices should be evaluated regularly. Look for AD tools that support different mobile device platforms (Android, iOS, etc.) and formats (phone, tablet, etc.). Plan to take advantage of industry trends toward HTML5 support.
Object-Oriented Analysis and Design (OOA&D) Methodologies	Stable development path. Implement OOA&D methods and tools in conjunction with the development of service-oriented applications and in collaboration with BPA methods and tools.

Key:
- Recommendation should be acted on by 2014
- Recommendation should be acted on by 2015
- Recommendation is less urgent

Software Requirements



1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

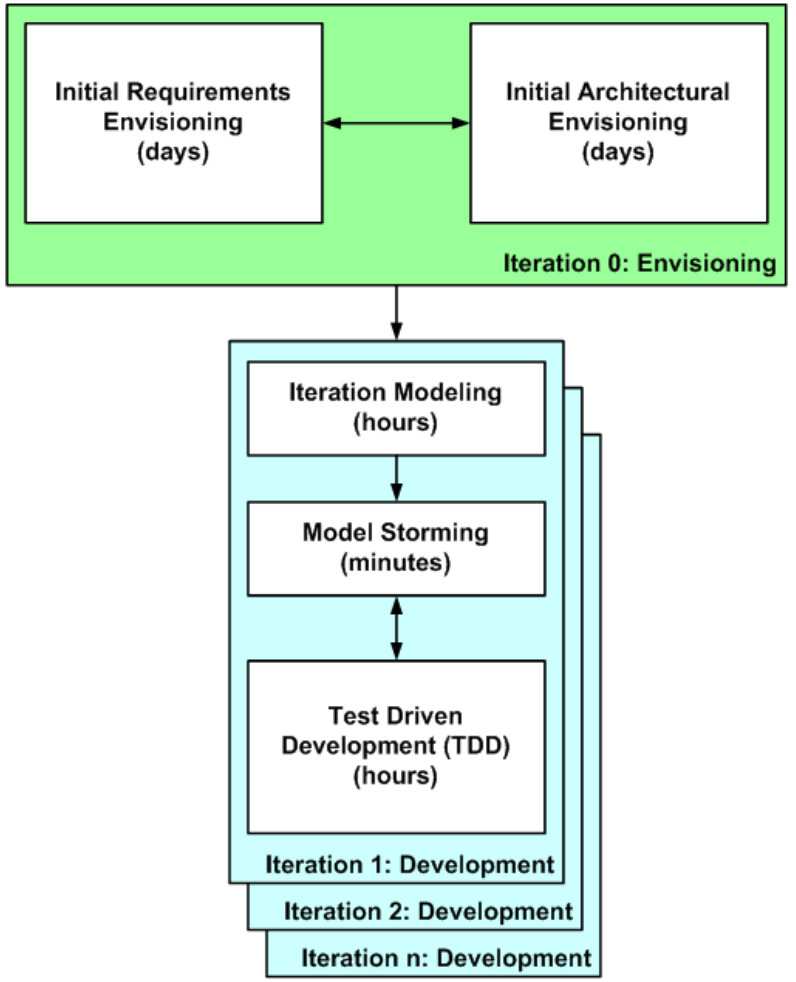
Joel Spolsky, The Joel Test: 12 Steps to Better Code (http://www.joelonsoftware.com/articles/fog0000000043.html)

Software Requirement Models

Abstraction Level of Model		Conceptual Model	Specification Model (Logical Model)	Implementation Model (Physical Model)
What is described		Business Requirements	Software Requirements	Software Design
Modeling Notation	UX	Persona, User Concept Map, User Journey Map, User Story	Information Architecture, Interaction Design	UI Wireframe, Navigation Structure, Contents Design
	Process	Conceptual BPMN Model	Executable BPMN Model	BPEL Script
	Data	UML Class Diagram	Database Schema	Physical Database Design
	Use Case	UML Use Case Diagram	User Story (Product Backlog Item), Use Case Scenarios	CRC, UML Class and Sequence Diagram
	Service	Business Service	Service Specification	Web Services

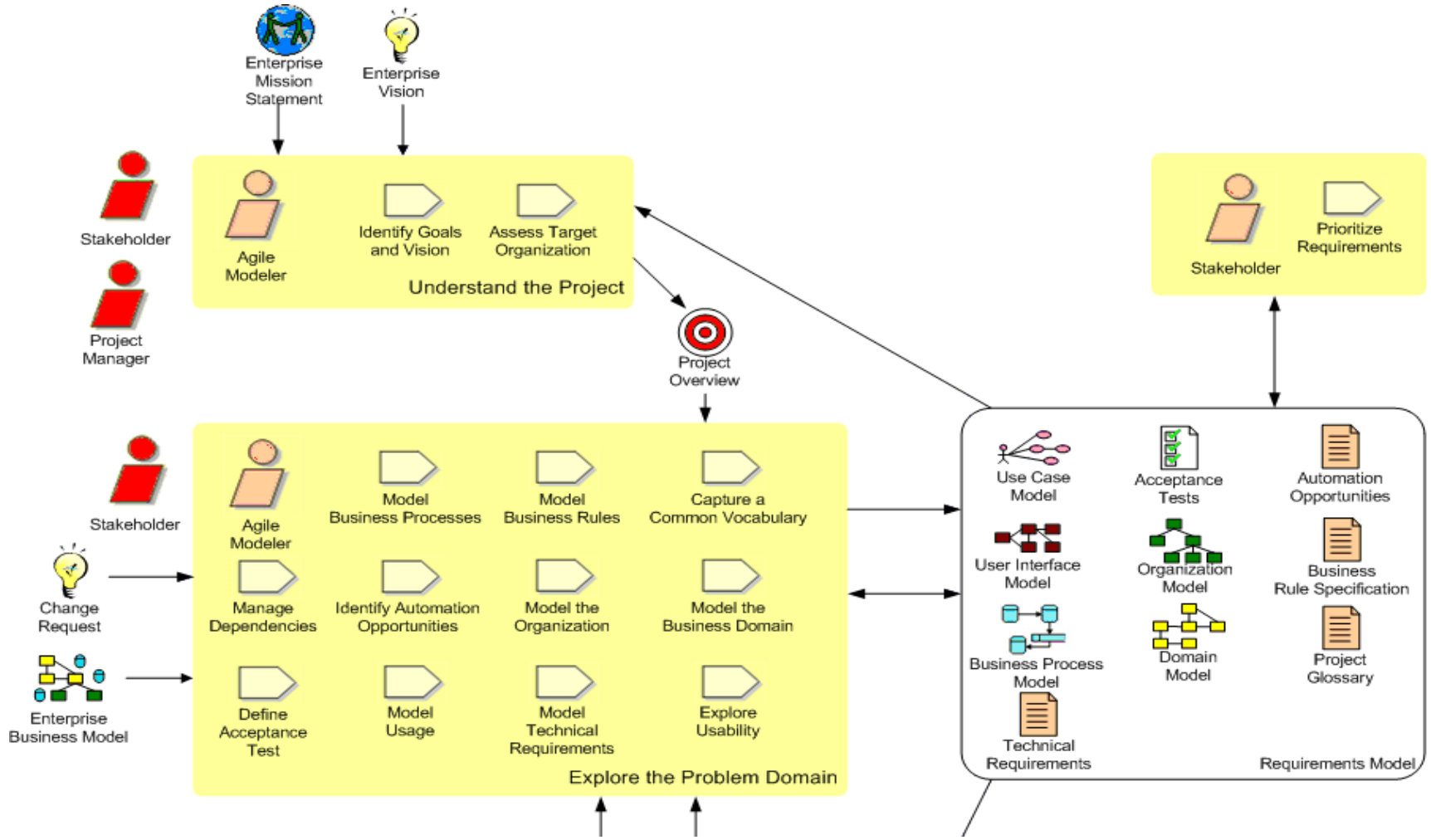
Agile Requirement Modeling @ IBM

- Create several models in parallel in small increments, applying the right artifact for the situation.
- Have stakeholders participate actively.
- Facilitate communication through collective ownership of your project artifacts and through applying modeling standards.
- Apply patterns.
- Do the majority of detailed modeling in the form of executable specifications.
- TDD is a must which includes complete unit tests prior to check-ins and daily builds and integration tests.



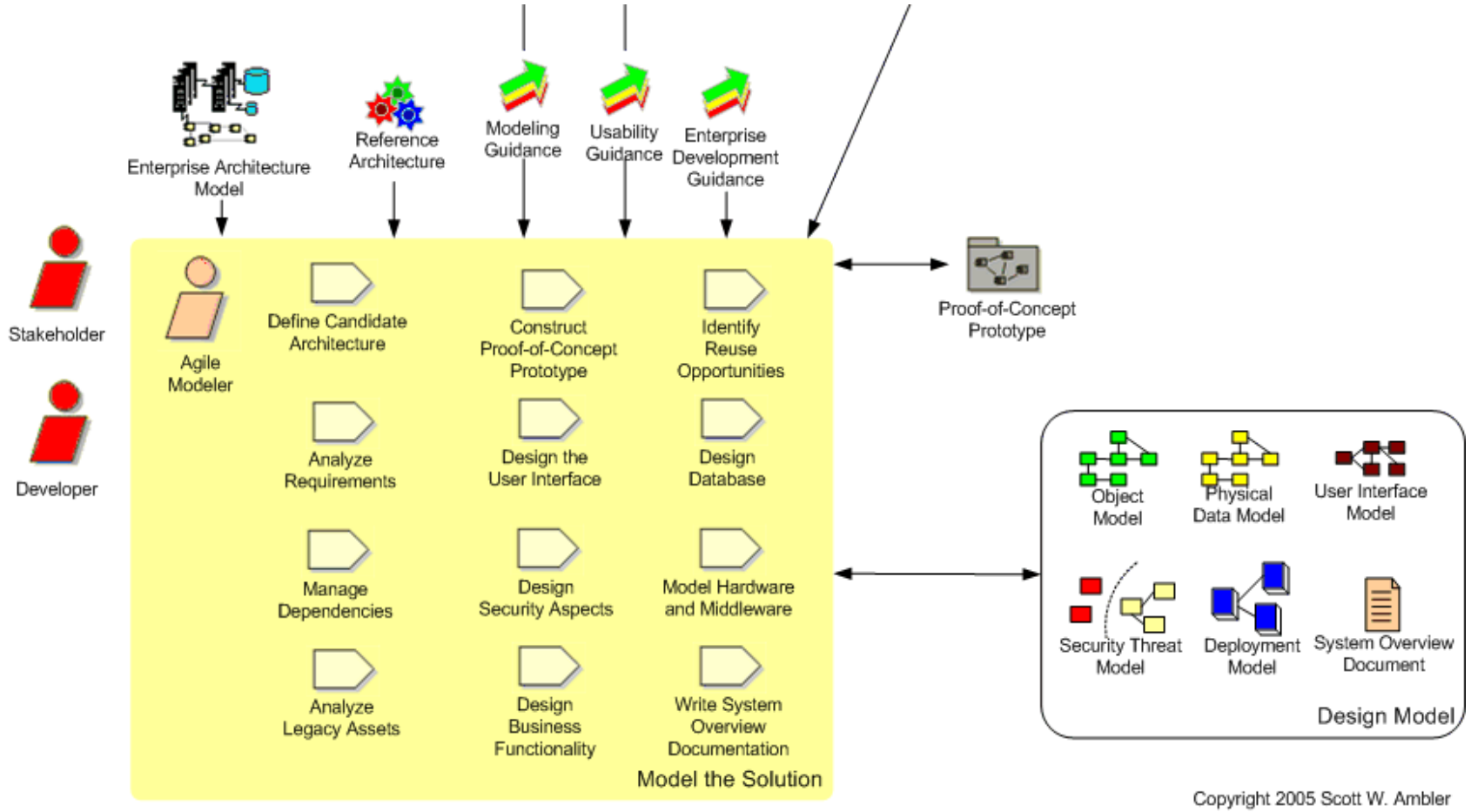
Scott Ambler, Agile Modeling (http://www.agilemodeling.com/)

Agile Requirement Modeling @ IBM

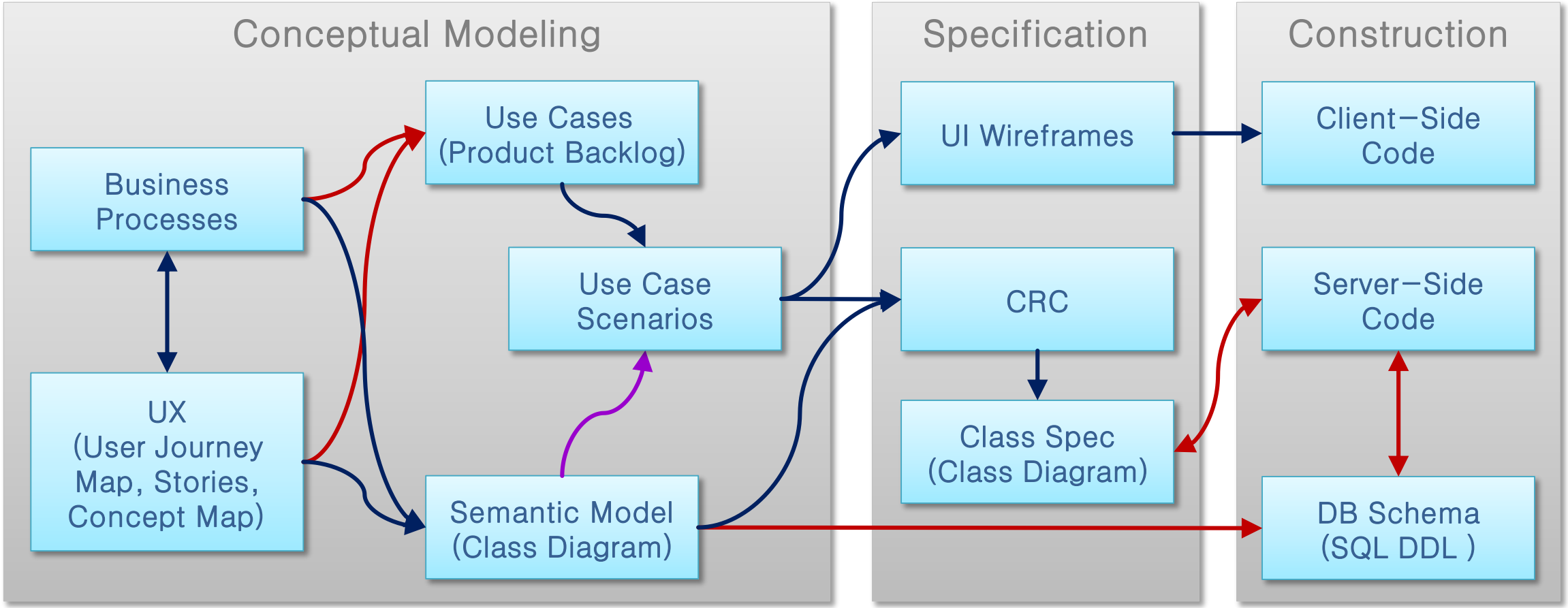


Scott Ambler, Agile Unified Process v1.1 (http://www.ambysoft.com/unifiedprocess/agileUP.html)

Agile Requirement Modeling @ IBM

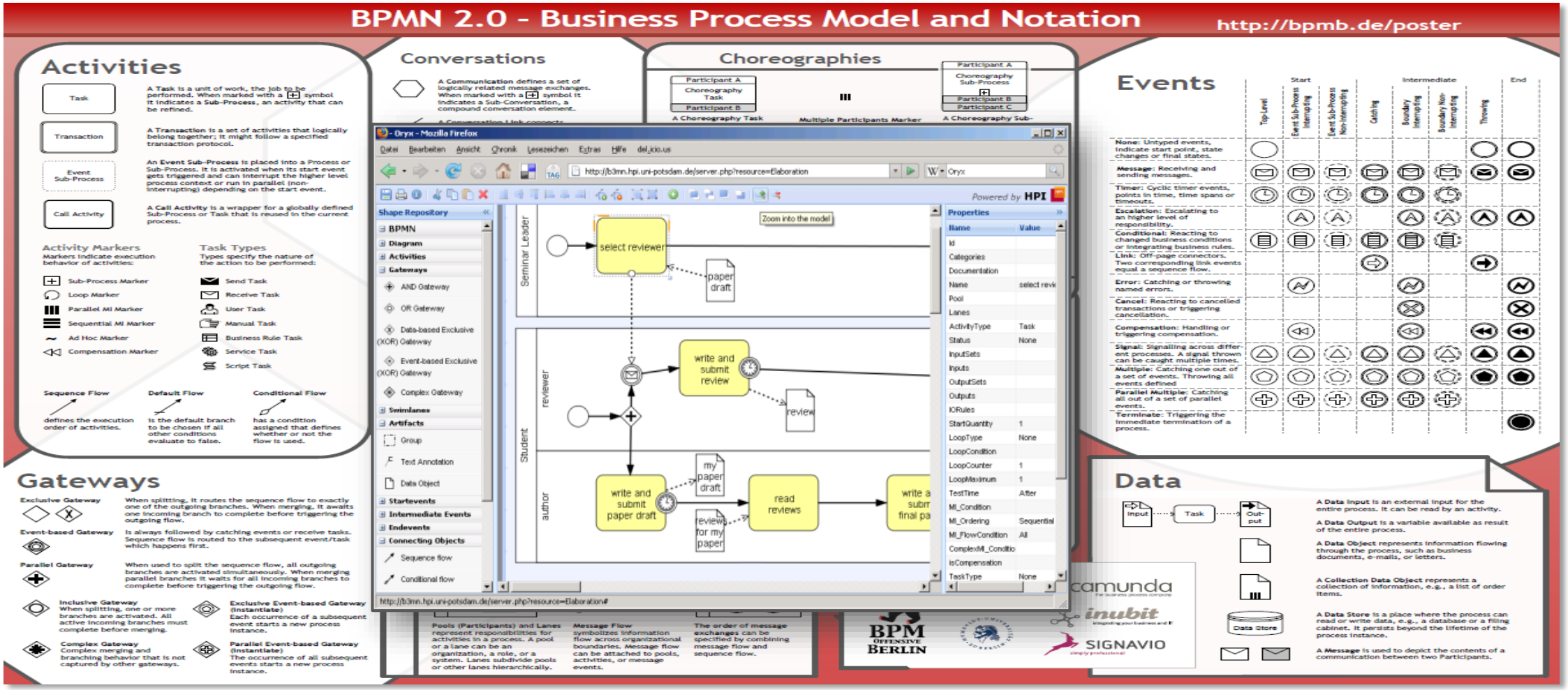


Agile Requirement Modeling



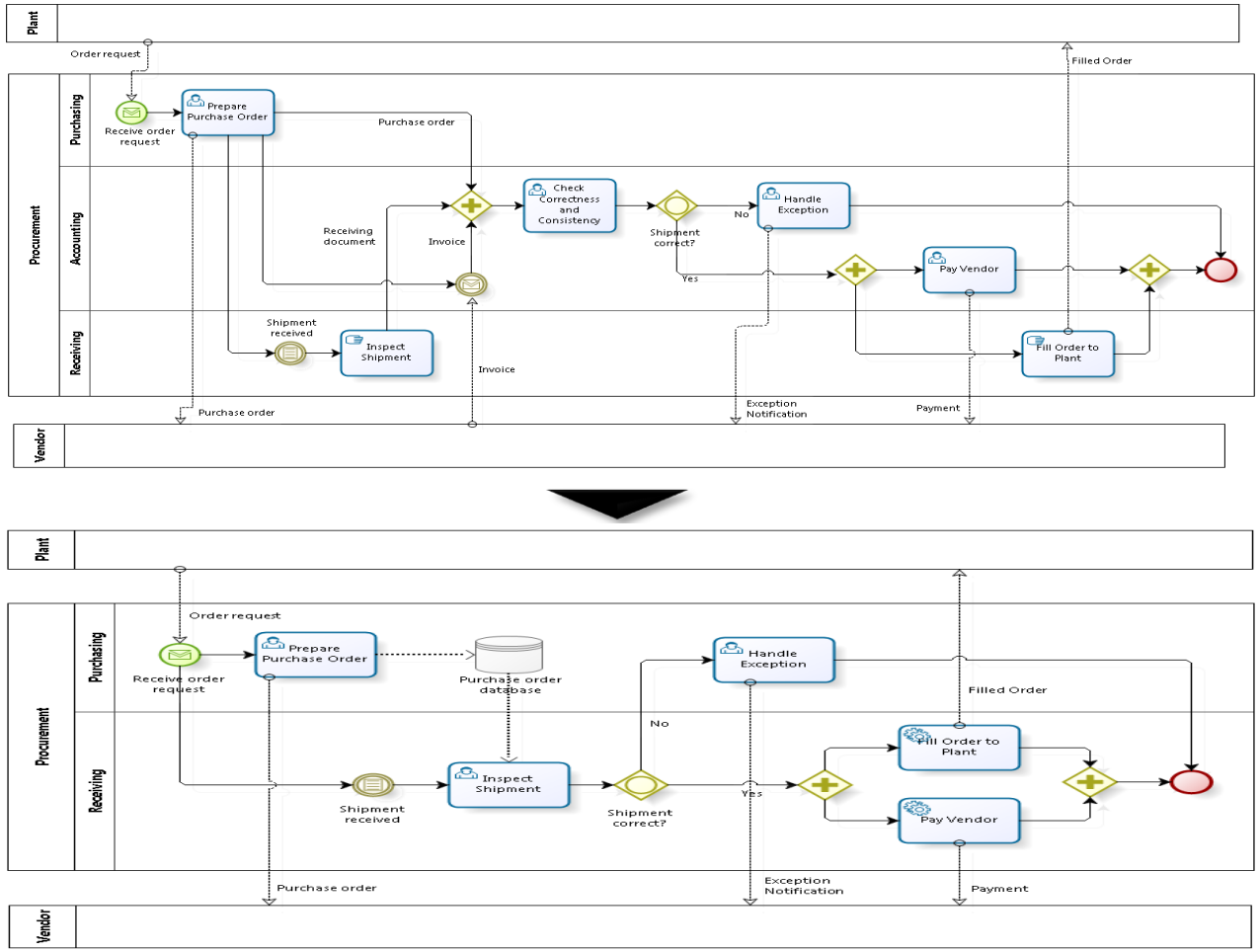
Visual Paradigm (http://www.visual-paradigm.com/features/)

Business Process Modeling



Business Process Reengineering

- Ford 's BPR of Procurement Process in early 1980s
- By adding a purchase order application and database and a network connecting its part vendors, Ford eliminated invoice.
- Payment authorization, used to be performed by accounting, was then done by the receiving clerk—Empowerment!
- Head count was reduced immediately from 500 to 125 in the a/p dept, and further to only 5-10 after a few years who were needed for handling exceptional situations.

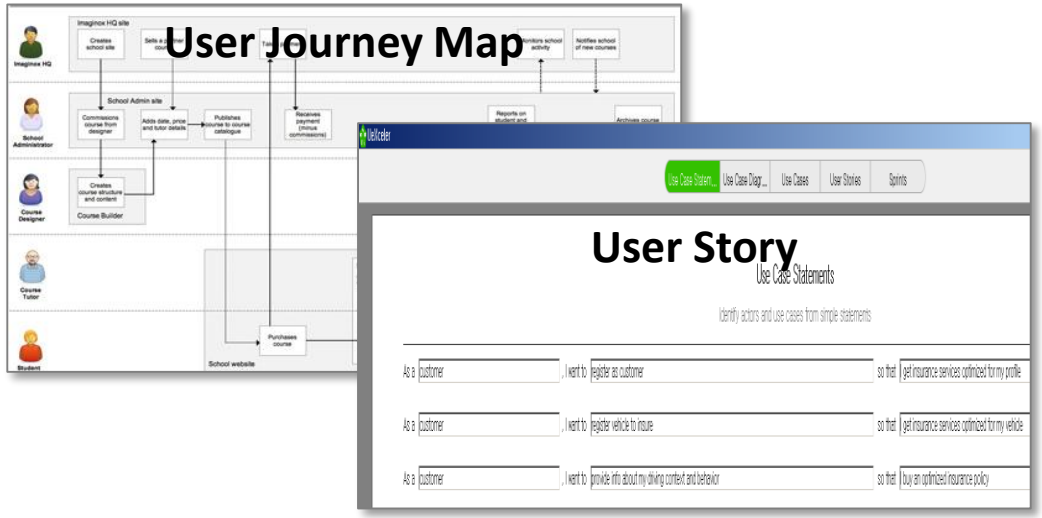


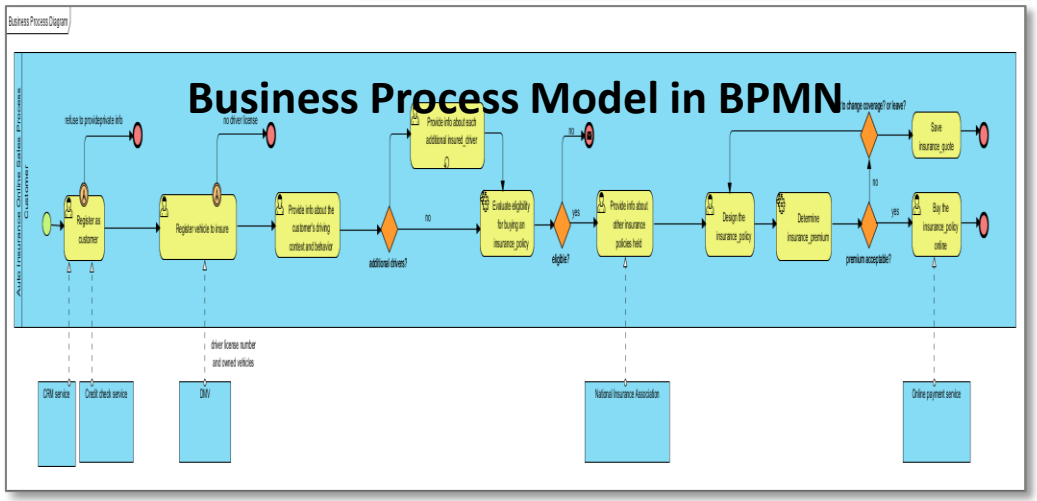
M. Hammer and J. Champy, Reengineering the Corporation—A Manifesto for Business Revolution, HarperBusiness, 1993.

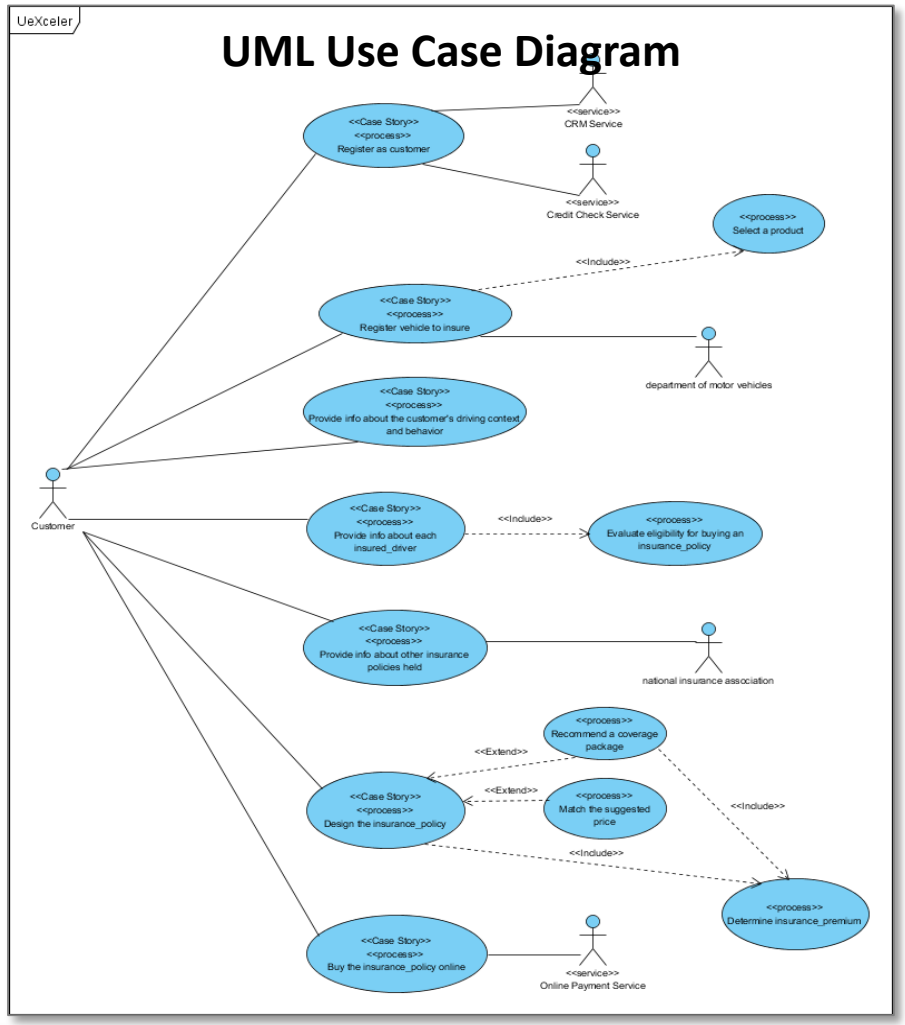
BPR Patterns

- Quick Win
- Parallelization
- Flattening and Empowerment
- Automation and Exception Handling
- Refactoring and Standardization
- Horizontal Integration
- Disintermediation
- Downstreaming and Self-Service
- Upstreaming and Partner-Managed Process
- Lean Process and Pull System
- Intelligent Process
- Social Business Process
- Mobile Business Process
- Context-Sensitive Event-Driven Process

UX & Process Analysis ➔ Use Case Discovery



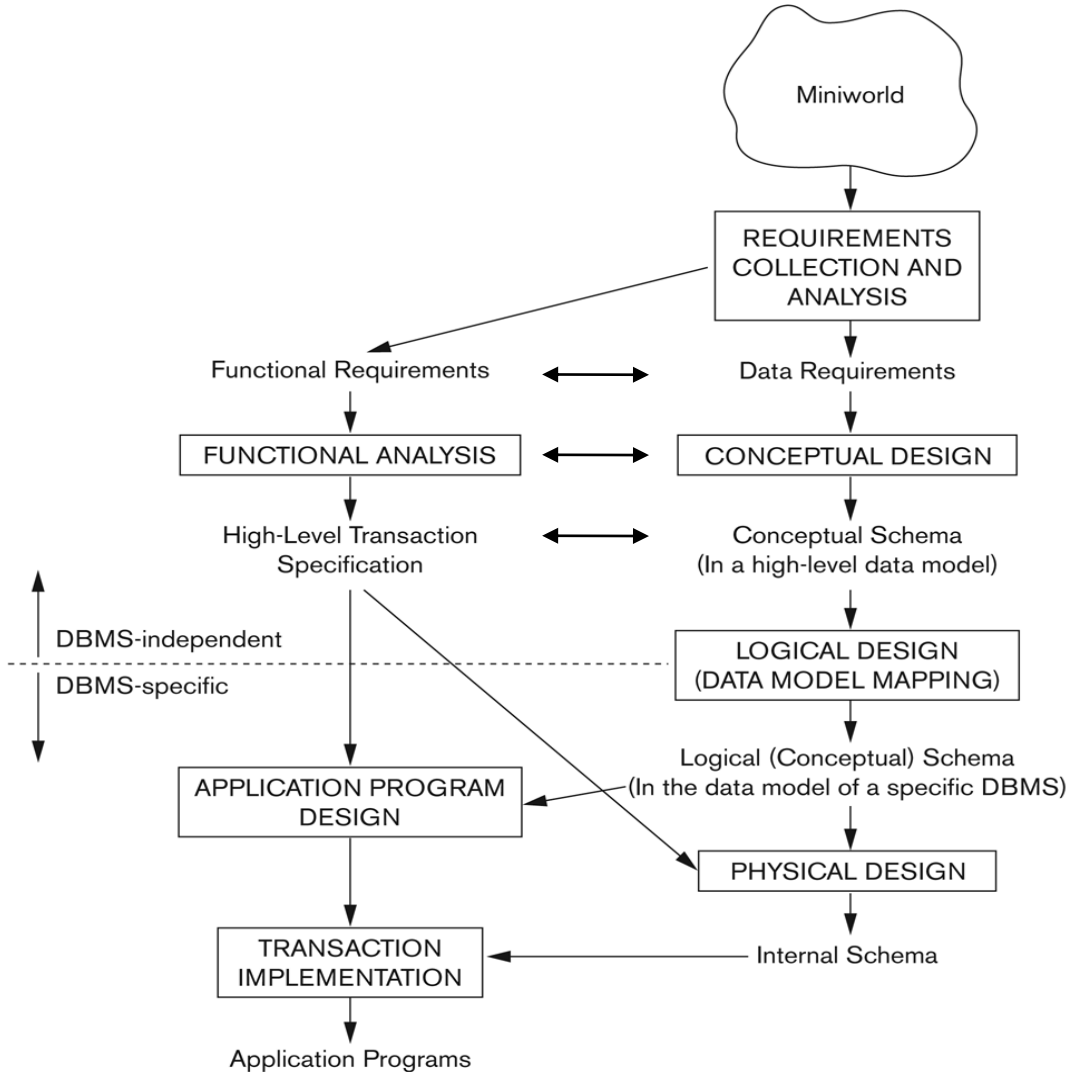




Data Modeling

Database Design Process

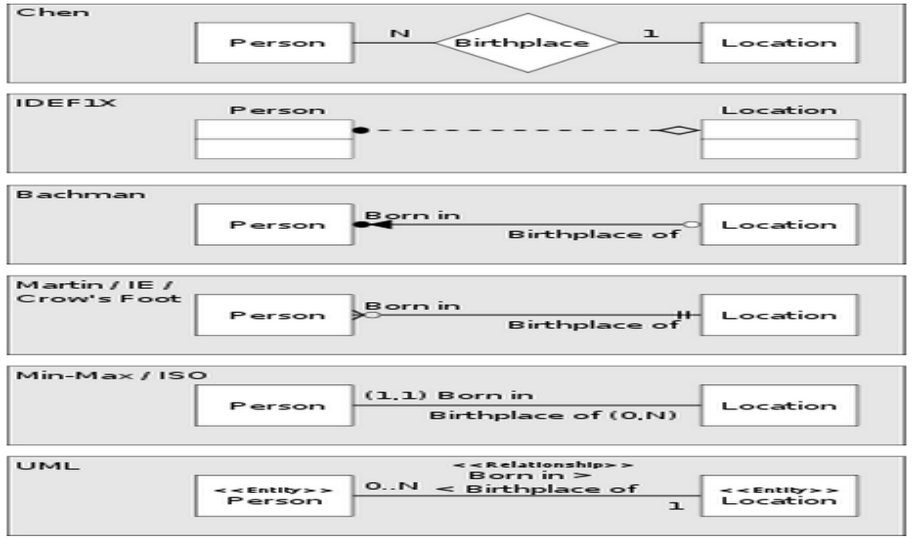
- Analysis of data requirements
- Semantic data modeling
- Mapping to logical database
- Verification of 5th normal form
- Transaction design
- Physical database design

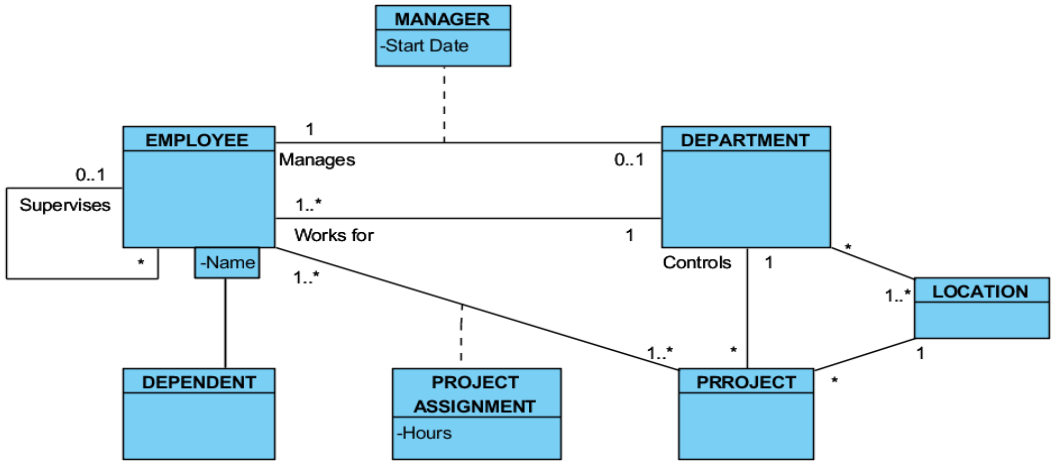


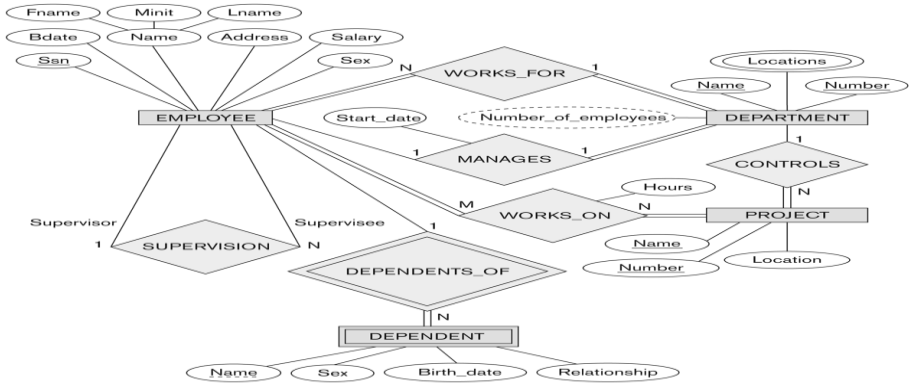
R. Elmasri and S. Navathe, Fundamentals of Database Systems (6th ed.), Addison Wesley, 2011.

UML Class Diagram for Sematic Modeling

- Since Entity-Relationship Diagram was proposed by Peter Chen in 1976, many variants have been developed.
- UML offers a richer set of notation to support advanced concepts of semantic modeling.
- ERDs and Class Diagrams are mechanically convertible to each other.



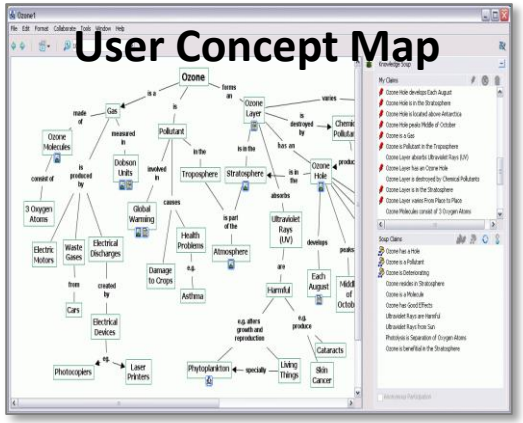




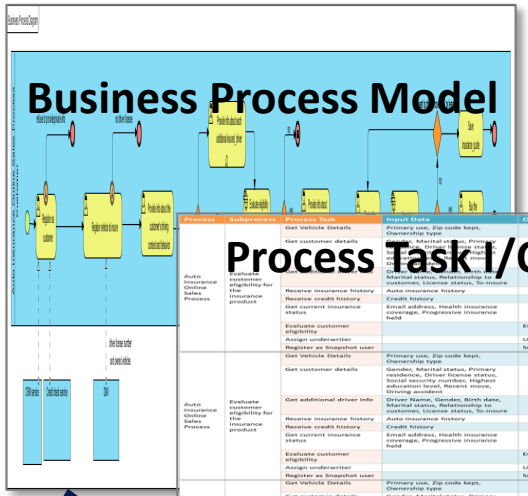
OMG, Unified Modeling Language Superstructure Version 2.4.1, August 2011 (http://www.omg.org/spec/UML/)

Semantic Modeling ➔ Database Design

User Concept Map

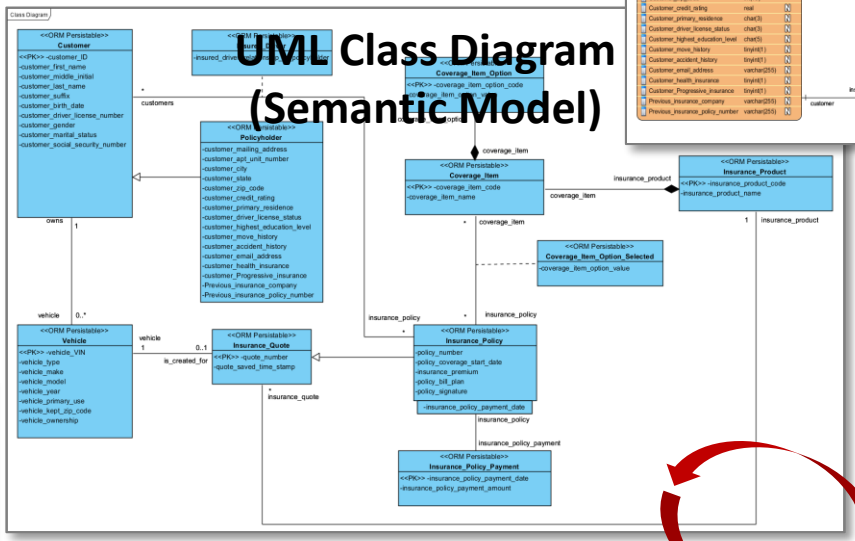


Business Process Model

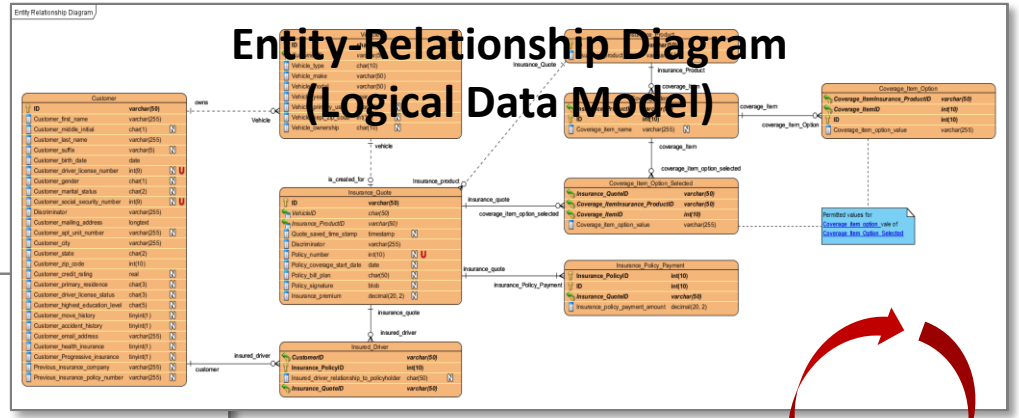


Process Task I/O

UML Class Diagram (Semantic Model)



Entity-Relationship Diagram (Logical Data Model)



SQL DDL (database Schema)

```
CREATE TABLE Coverage_Item_Option_Selected (
    Coverage_ItemInsurance_ProductID    varchar(50)     NOT NULL,
    Coverage_ItemID                     int(10)         NOT NULL,
    Insurance_PolicyID                  int(10)         NOT NULL,
    Coverage_Item_Option_Value          varchar(255)    NOT NULL,
    PRIMARY KEY (Coverage_ItemInsurance_ProductID,  Coverage_ItemID,
    Insurance_PolicyID));

ALTER TABLE Coverage_Item_Option_Selected
    ADD INDEX FKCoverage_I223695 (Coverage_ItemInsurance_ProductID,
    Coverage_ItemID),
    ADD CONSTRAINT FKCoverage_I223695 FOREIGN KEY
    (Coverage_ItemInsurance_ProductID, Coverage_ItemID) REFERENCES
    Coverage_Item (Insurance_ProductID, ID);

ALTER TABLE Coverage_Item_Option_Selected
    ADD INDEX FKCoverage_I252938 (Insurance_PolicyID),
    ADD CONSTRAINT FKCoverage_I252938 FOREIGN KEY (Insurance_PolicyID)
    REFERENCES Insurance_Policy (ID);
```

Use Case Scenario

Use Case ID:	2		
Use Case Name:	Evaluate customer eligibility for the insurance product		
Created By:		Last Updated By:	
Date Created:		Date Last Updated:	
Actors:	Customer, National Insurance Association, Credit Bureau, Underwriter		
Description:	Customer enters detailed information about herself and additional drivers; then the system obtains her insurance and credit history, and evaluate her eligibility for the selected insurance product. If the customer is not eligible for the online application but eligible for an underwriter's review, she may choose to apply for an underwriter's evaluation.		
Preconditions:	Basic information has been obtained about a registered customer insurance product.		
Postconditions:	The customer is accepted so that she can proceed to design her in		
Normal Flow:	2.0 Evaluate the customer's eligibility for an online insurance appli 1. System asks Customer about Social security number, Primary r education level, Driver license status, Recent move and Driving 2. Customer enters the requested information. 3. System asks Customer about Driver's Name, Gender, Birth date License status and To-Insure for each additional driver of the ir 4. Customer enters the requested information. 5. System requests National Insurance Association and Credit Bu Customer's insurance history and credit history, respectively. 6. System receives replies from the two institutions. 7. System asks Customer about her Email address, Health insurar held. 8. System uses Use Case 7 (Evaluate customer eligibility) to deter 9. System terminates the use case.		

Alternative Flows:	None
Exceptions:	2.0.E.1 Bad Credit (at Step 6) 1 System determines rejection of Customer because of a bad credit status. 2 System terminates the use case. 2.0.E.2 Uneligible Customer (at Step 8) 1a System rejects Customer based on an ineligibility decision made by Use Case 7. 2a System terminates the use case. 1b System notifies Customer that she is not eligible for the online application and asks if she wants to have her application reviewed by an underwriter. 2b(i) Customer requests an underwriter evaluation. 3b(i) System assigns an underwriter and notify him of this case. 4b(i) System terminates the use case. 2b(ii) Customer declines an underwriter review. 3b(ii) System terminate the use case.
Includes:	Use Case 7: Evaluate customer eligibility
Priority:	High
Business Rules:	R-1 Business rule that specifies the customer's credit status condition under which the customer is uneligible for an online insurance application. R-2 Business rule to determine a customer's eligibility for an online application of an insurance policy.

Use Case Scenario

What's a Spec?

When you're designing a product, you need to have some real live scenarios in mind for how people are going to use it. …

Details are the most important thing in a functional spec. …

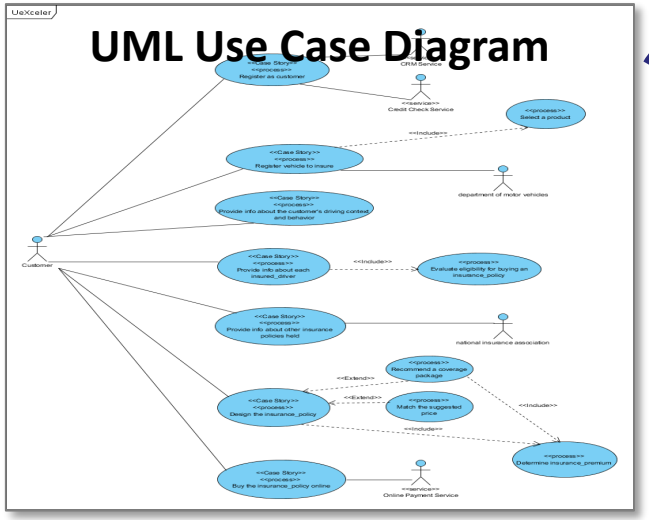
As a program manager at Microsoft, I designed the VBA for Excel 5.0 and completely speced out, to the smallest detail, how VBA should be implemented in Excel. My spec ran to about 500 pages. … Every morning, 250 people came to work and basically worked off of that huge spec I wrote. …

The updating continues as the product is developed and new decisions are made. It is only frozen when the product is code complete

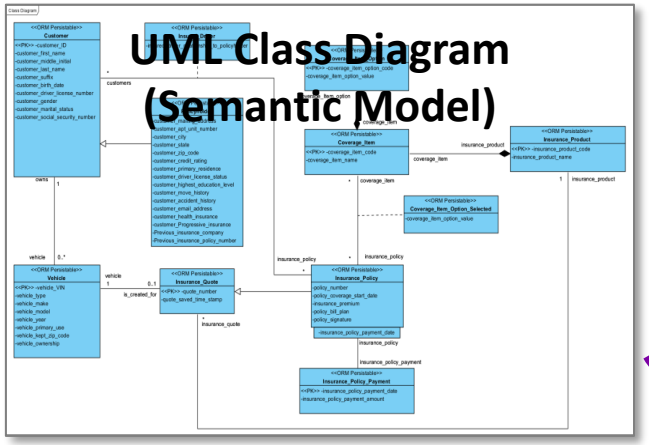
Joel Spolsky, Painless Functional Specifications - Part 1: Why Bother? (http://www.joelonsoftware.com/articles/fog0000000036.html)

Product Backlog ➔ Use Case Scenario

UML Use Case Diagram



UML Class Diagram (Semantic Model)



User Case (Product Backlog Item)

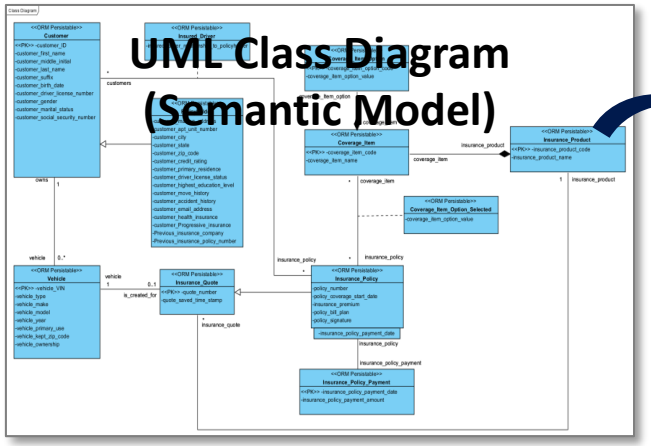
Use Case Scenario

Design the insurance policy

1. SYSTEM generate a unique policy_number and creates an Insurance_Policy
2. SYSTEM displays three policy design options: (1) get recommendation of a coverage package, (2) suggest the price to match, and (3) determine coverage for herself
3. if Customer chooses option (1)
    3.1. Use Recommend a coverage package use case to recommend coverage and insurance_premium
4. else if Customer chooses option (2)
    4.1. Use Match the suggested price use case to recommend coverage that matches the suggested insurance_premium
5. else Customer chooses option (3)
    5.1. SYSTEM displays all Coverage_Item of Insurance_Product and all Coverage_Item_Option of each Coverage_Item
    5.2. Customer selects Coverage_Item_Option_Selected for each Coverage_Item
    5.3. SYSTEM uses Determine insurance_premium use case to compute insurance_premium
   end if
6. SYSTEM displays all Coverage_Item_Option_Selected and insurance_premium
7. if Customer changes Coverage_Item_Option_Selected
    7.1. SYSTEM saves the changes
8. else if Customer wants to leave without continuing to buy Insurance_Policy
    8.1. SYSTEM saves Insurance_Quote, delete Insurance_Policy, and terminates the process
9. Else
    9.1. SYSTEM asks Customer to provide policy_coverage_start_date
    9.2. Customer enter the information
    9.3. SYSTEM updates the Insurance_Policy.
   end if

Construct use case scenario statements using defined classed and attributes

CRC ➔ Server-Side Code Generation



UML Class Diagram (Semantic Model)

Realize use case scenarios by assigning scenario steps to class operations

CRC (Class-Responsibility-Collaboration)

Super Classes: Insurance_Quote

Sub Classes:

Description:

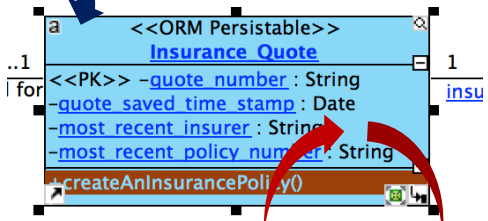
Attributes:

Name	Description
quote_number	
vehicle_VIN	
quote_saved_time_stamp	
policy_flag	
policy_number	
policy_coverage_start_date	
policy_bill_plan	
policy_sinature	
insurance_premium	

Responsibilities:

Name	Collaborator
Create an insurance policy	
Update an insurance policy	Insurance_Product, Coverage_Item, Coverage_Item_Option, Coverage_Item_Option_Selected
Produce the doc[illegible] of an insurance	Vehicle, Policyholder, Insured_Driver, Insurance_Product, Coverage_Item_Option_Selected

Class Spec



Use Case Scenario

Design the insurance policy

1. SYSTEM generate a unique policy_number and creates an Insurance_Policy
2. SYSTEM displays three policy design options, (1) get recommendation on a coverage package, (2) suggest the price to match, and (3) determine coverage for [illegible]
3. if Customer chooses option (1)
   - 3.1. Use Recommend a coverage package use case to recommend coverage and insurance_premium
4. else if Customer chooses option (2)
   - 4.1. Use Match the suggested price use case to recommend coverage that matches the suggested insurance_premium
5. else Customer chooses option (3)
   - 5.1. SYSTEM displays all Coverage_Item of Insurance_Product and all Coverage_Item_Option of each Coverage_Item
   - 5.2. Customer selects Coverage_Item_Option_Selected for each Coverage_Item
   - 5.3. SYSTEM uses Determine insurance_premium use case to compute insurance_premium

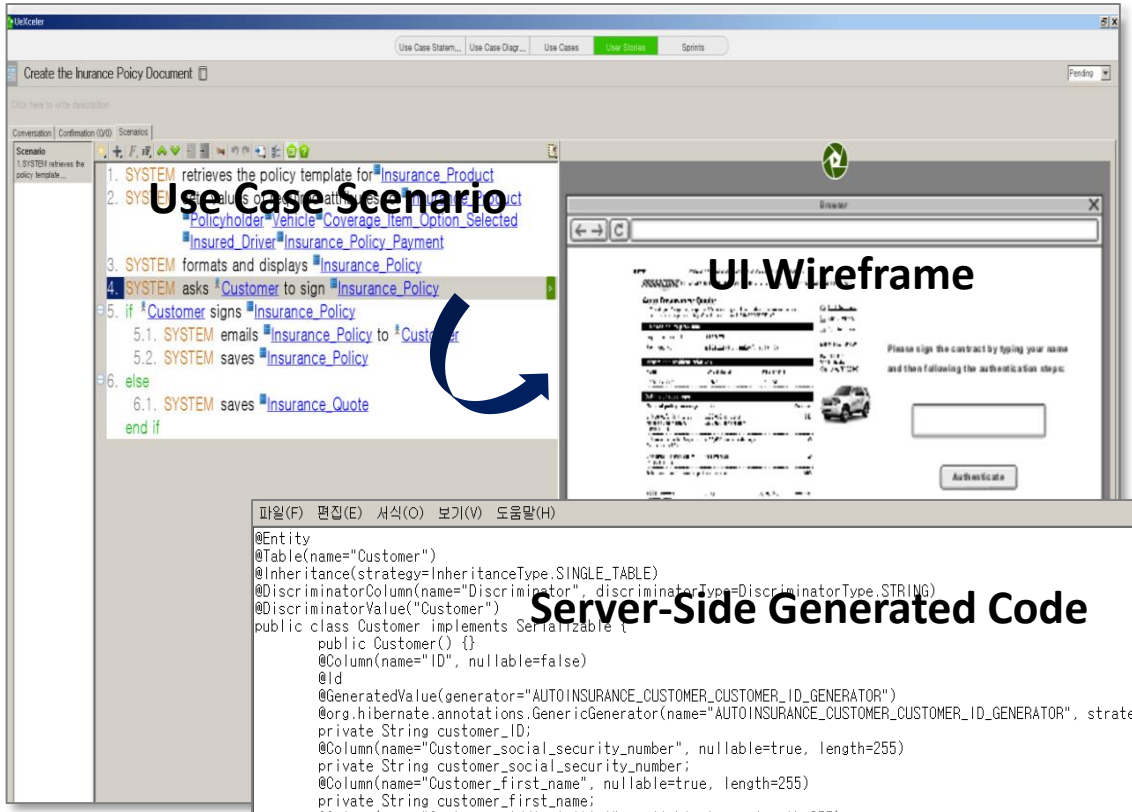
   end if
6. SYSTEM displays all Coverage_Item_Option_Selected and insurance_premium
7. if Customer changes Coverage_Item_Option_Selected
   - 7.1. SYSTEM saves the changes
8. else if Customer wants to leave without continuing to buy Insurance_Policy
   - 8.1. SYSTEM saves Insurance_Quote, delete Insurance_Policy, and terminates the process
9. Else
   - 9.1. SYSTEM asks Customer to provide policy_coverage_start_date
   - 9.2. Customer enter the information
   - 9.3. SYSTEM updates the Insurance_Policy.

   end if

Code Generation

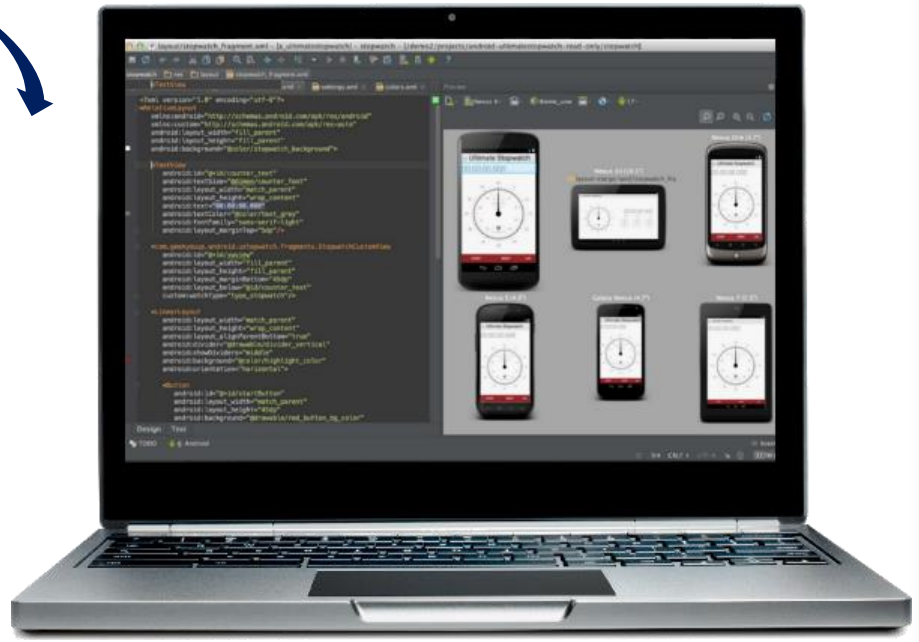
```
@Entity
@Table(name="Customer")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="Discriminator", discriminatorType=DiscriminatorType.STRING)
@DiscriminatorValue("Customer")
public class Customer implements Serializable {
	public Customer() {}
	@Column(name="ID", nullable=false)
	@Id
	@GeneratedValue(generator="AUTOINSURANCE_CUSTOMER_CUSTOMER_ID_GENERATOR")
	@org.hibernate.annotations.GenericGenerator(name="AUTOINSURANCE_CUSTOMER_CUSTOMER_ID_GENERATOR", strategy="native")
	private String customer_ID;
	@Column(name="Customer_social_security_number", nullable=true, length=255)
	private String customer_social_security_number;
	@Column(name="Customer_first_name", nullable=true, length=255)
	private String customer_first_name;
	@Column(name="Customer_middle_initial", nullable=true, length=255)
	private String customer_middle_initial;
	@Column(name="Customer_last_name", nullable=true, length=255)
	private String customer_last_name;
	@OneToMany(mappedBy="owns", targetEntity=autoinsurance.Vehicle.class)
	@org.hibernate.annotations.Cascade({org.hibernate.annotations.CascadeType.SAVE_UPDATE, org.hibernate.annotations.CascadeType.LOCK})
	@org.hibernate.annotations.LazyCollection(org.hibernate.annotations.LazyCollectionOption.TRUE)
	private java.util.Set vehicles = new java.util.HashSet();
}
```

UI Wireframe ➔ Client-Side Coding



Server-Side Generated Code

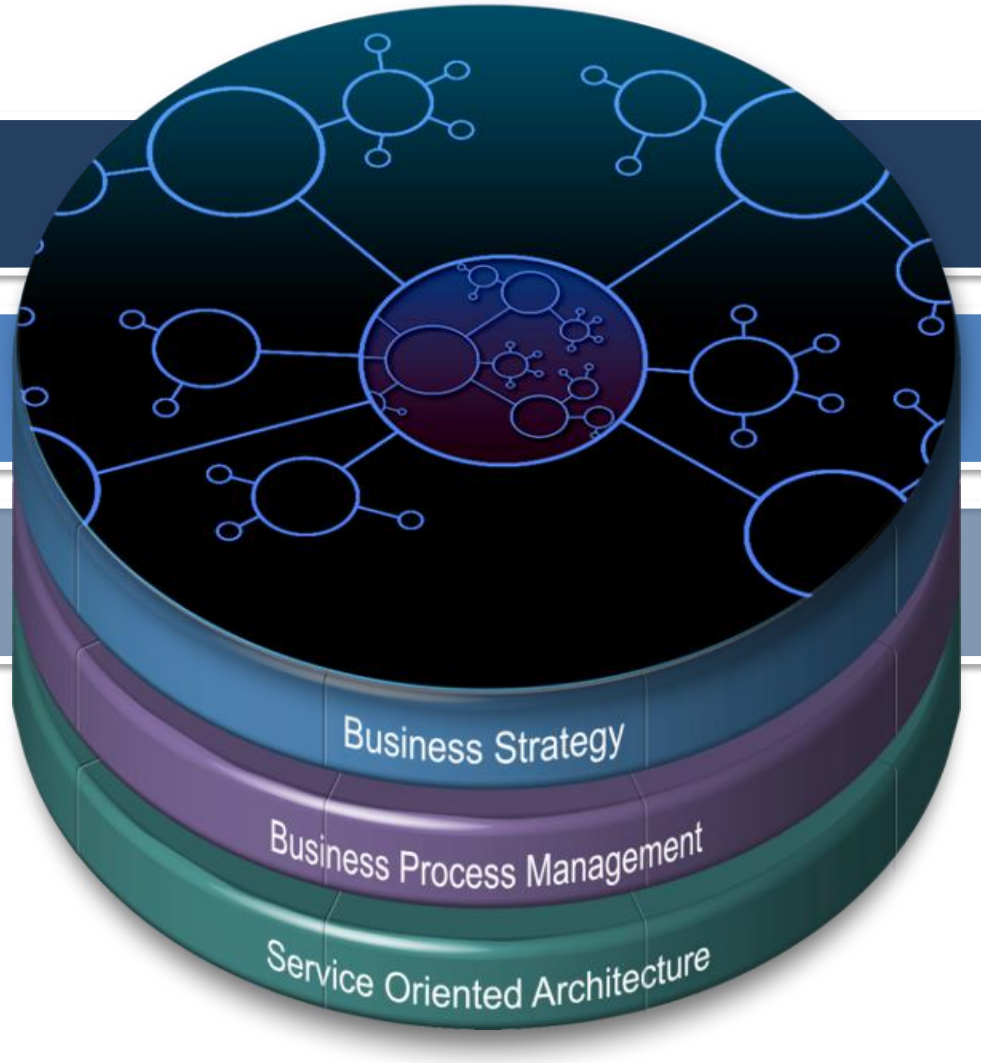
```
@Entity
@Table(name="Customer")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="Discriminator", discriminatorType=DiscriminatorType.STRING)
@DiscriminatorValue("Customer")
public class Customer implements Serializable {
        public Customer() {}
        @Column(name="ID", nullable=false)
        @Id
        @GeneratedValue(generator="AUTOINSURANCE_CUSTOMER_CUSTOMER_ID_GENERATOR")
        @org.hibernate.annotations.GenericGenerator(name="AUTOINSURANCE_CUSTOMER_CUSTOMER_ID_GENERATOR", strategy="native")
        private String customer_ID;
        @Column(name="Customer_social_security_number", nullable=true, length=255)
        private String customer_social_security_number;
        @Column(name="Customer_first_name", nullable=true, length=255)
        private String customer_first_name;
        @Column(name="Customer_middle_initial", nullable=true, length=255)
        private String customer_middle_initial;
        @Column(name="Customer_last_name", nullable=true, length=255)
        private String customer_last_name;
        @OneToMany(mappedBy="owns", targetEntity=autoinsurance.Vehicle.class)
        @org.hibernate.annotations.Cascade({org.hibernate.annotations.CascadeType.SAVE_UPDATE, org.hibernate.annotations.CascadeType.LOCK})
        @org.hibernate.annotations.LazyCollection(org.hibernate.annotations.LazyCollectionOption.TRUE)
        private java.util.Set vehicles = new java.util.HashSet();
}
```



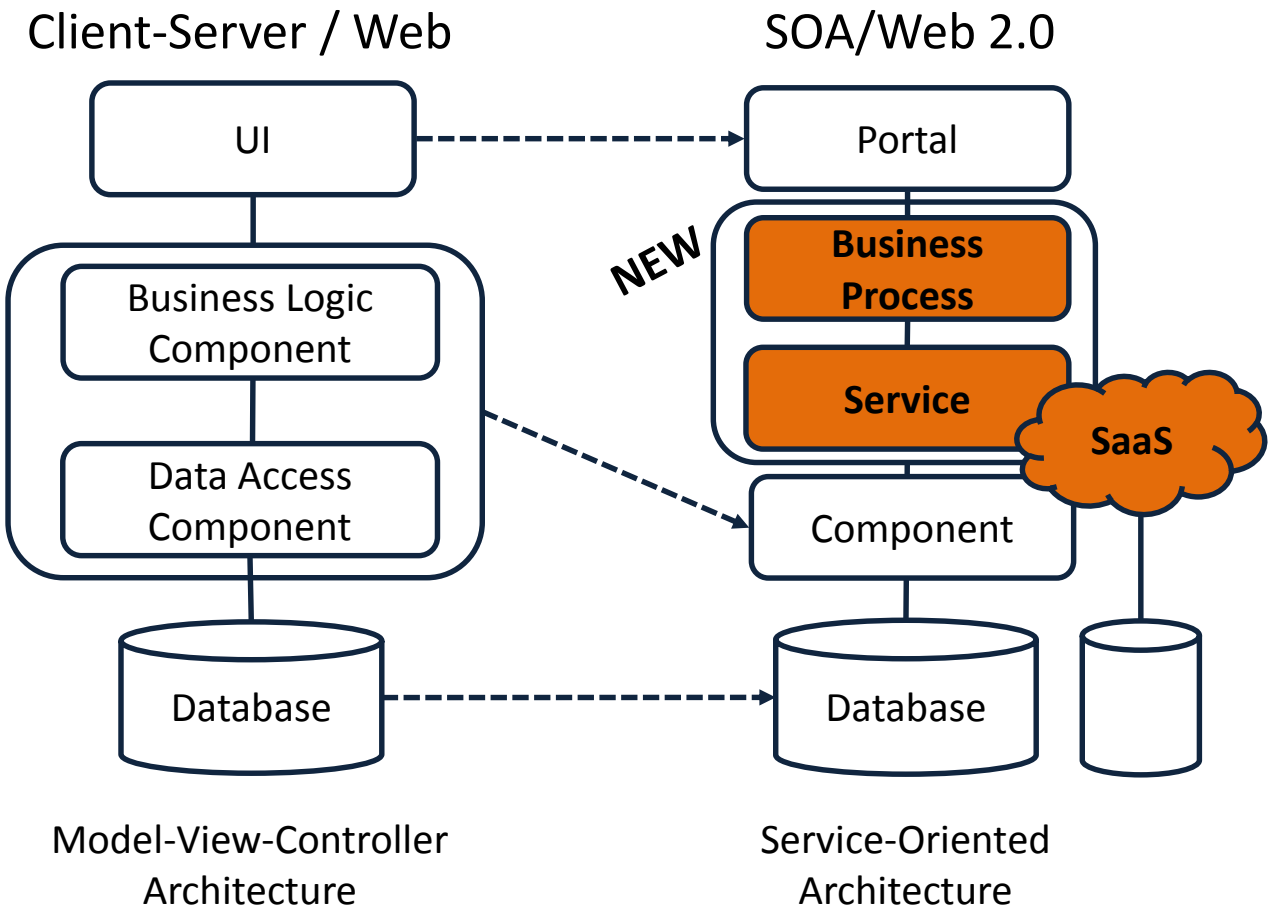
IDE with
Open Source Frameworks

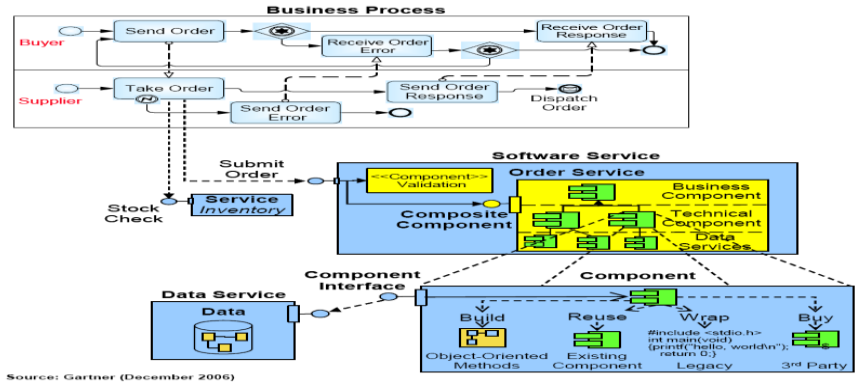
http://www.visual-paradigm.com/support/documents/vpuserguide/2381_ideintegrati.html

Table of Contents

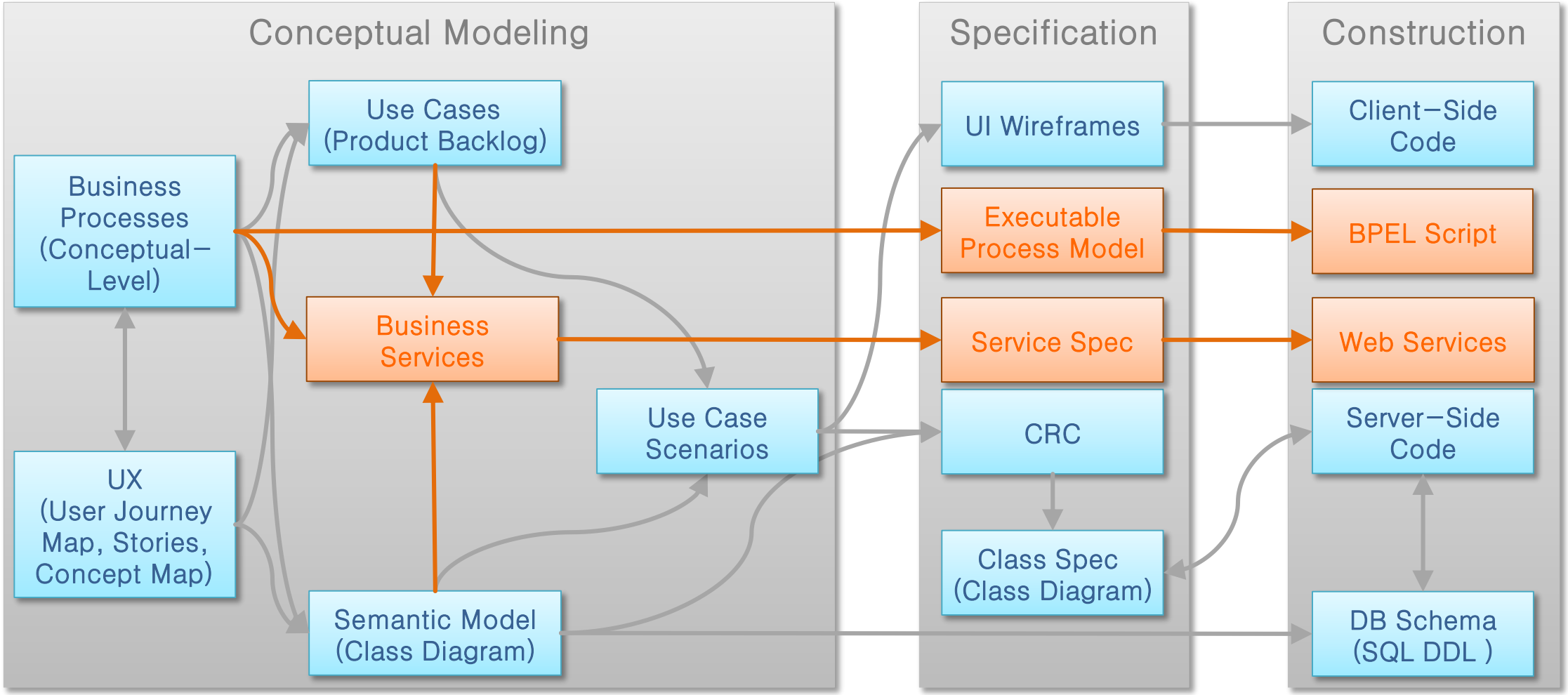


Service-Oriented Architecture





Service-Oriented Requirement Modeling



Business Process Management (BPM)

Design

Graphical modeling, process simulation and business rules

Implement

Code generation

Execute

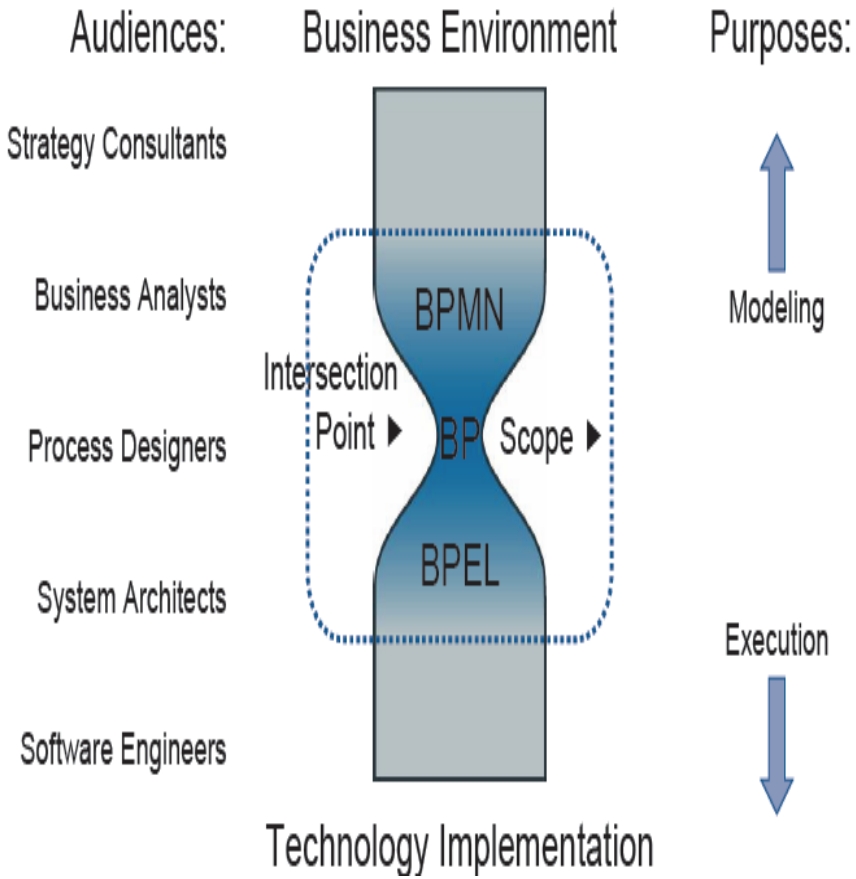
Automation, workflow and integration

Monitor

Business activity monitoring, automated process discovery and dashboards

Optimize

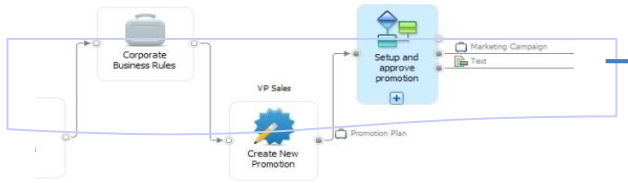
Analyze and dynamically adjust business processes and rules



BPM/SOA Implementation

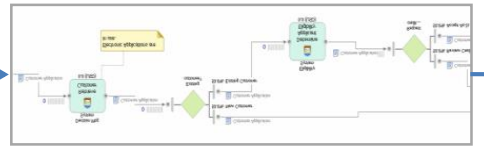
- With a service-oriented enterprise architecture established, companies can perform in-flight process changes in the enterprise applications in response to changes in business environments and strategies.

Process Re-Design using BPMN

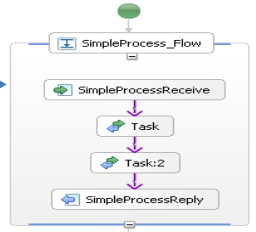


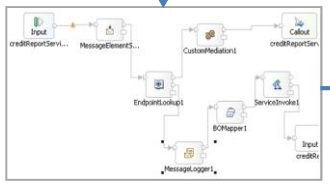
Process KPI Definition

Process Simulation

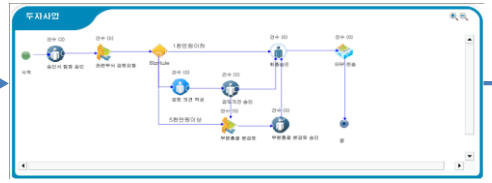


BPEL Process Implementation

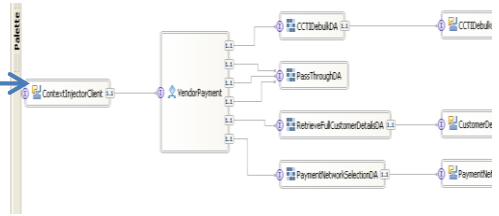




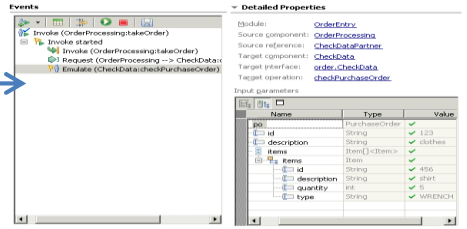
Service Specification



BPM UI and Monitoring Implementation

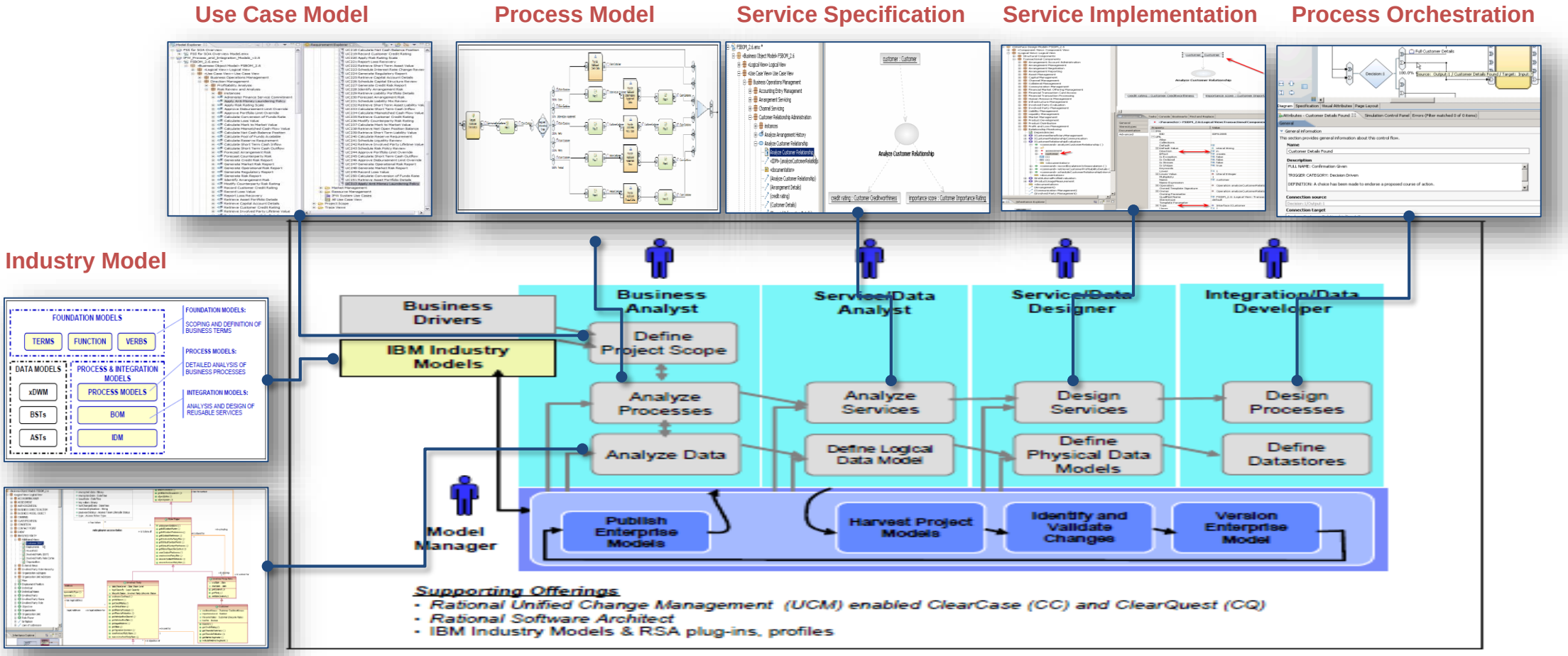


Service Realization



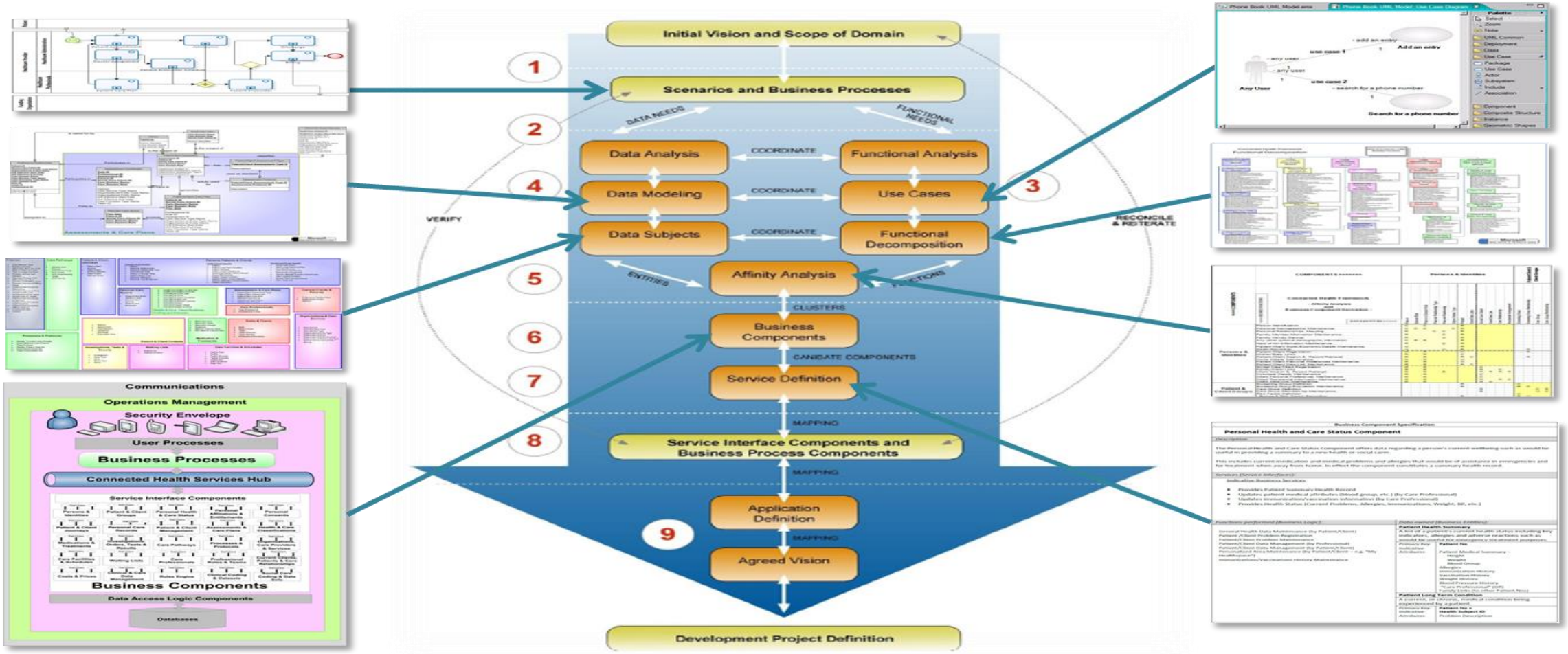
Integration Test and Execution

Service-Oriented Development @ IBM



http://www.redbooks.ibm.com/abstracts/redp4232.html

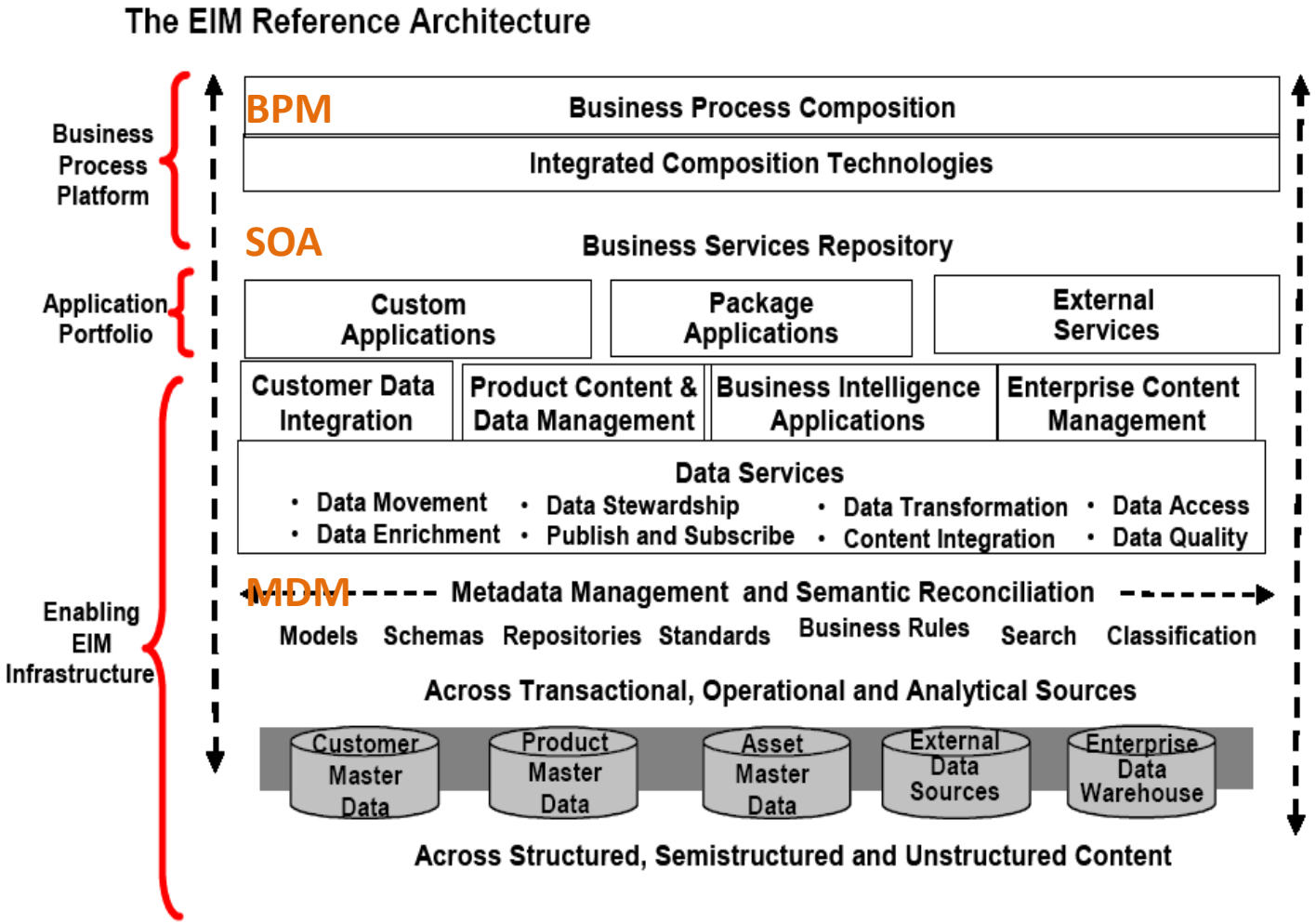
Service-Oriented Development @ Microsoft



Microsoft, Connected Health Framework-Architecture and Design Blueprint, 2nd Ed. March 2009.

Metadata Management

- Business service repository and data federation layer virtualize and synchronize physical apps and data sources to provide an integrated and standardized foundation.
- With the increasing variety of data sources, it is desirable to
  - Document data architectures through models
  - Use modeling tool suites to visualize data definitions
  - Implement metadata repositories and registries to understand and manage metadata.

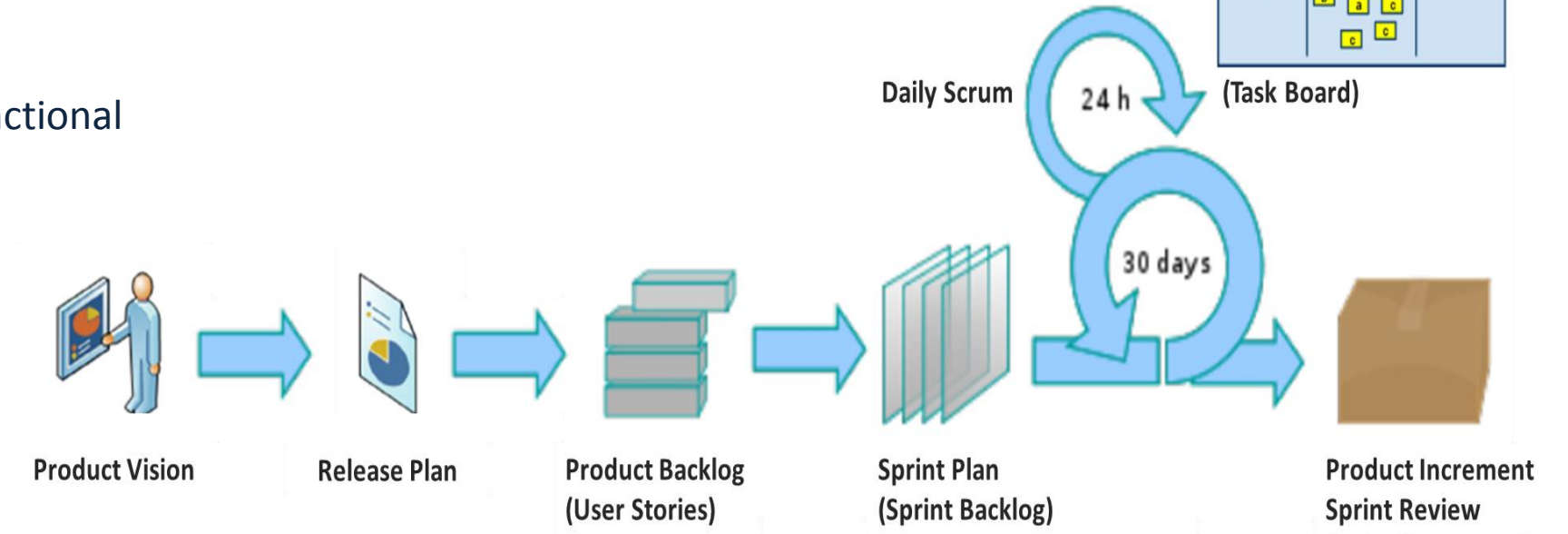


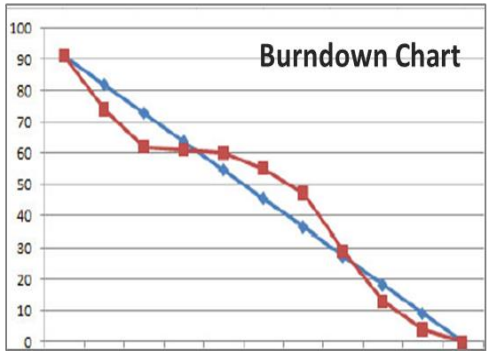
Source: Gartner (September 2005)

Table of Contents

Scrum Project Management

- Product owner
- Scrum master
- Developer: Multifunctional
- Pig and Chicken





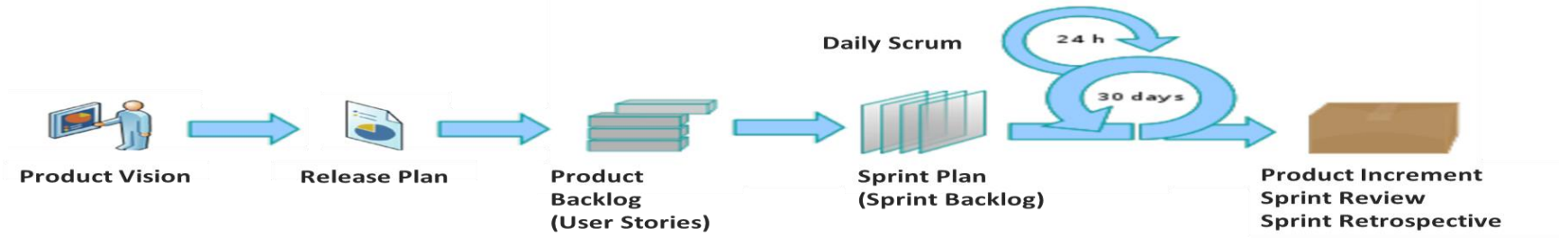
Jeff Sutherland and Ken Schwaber, The Scrum Guide, 2013. (http://www.scrumguides.org/)

Scrum Team

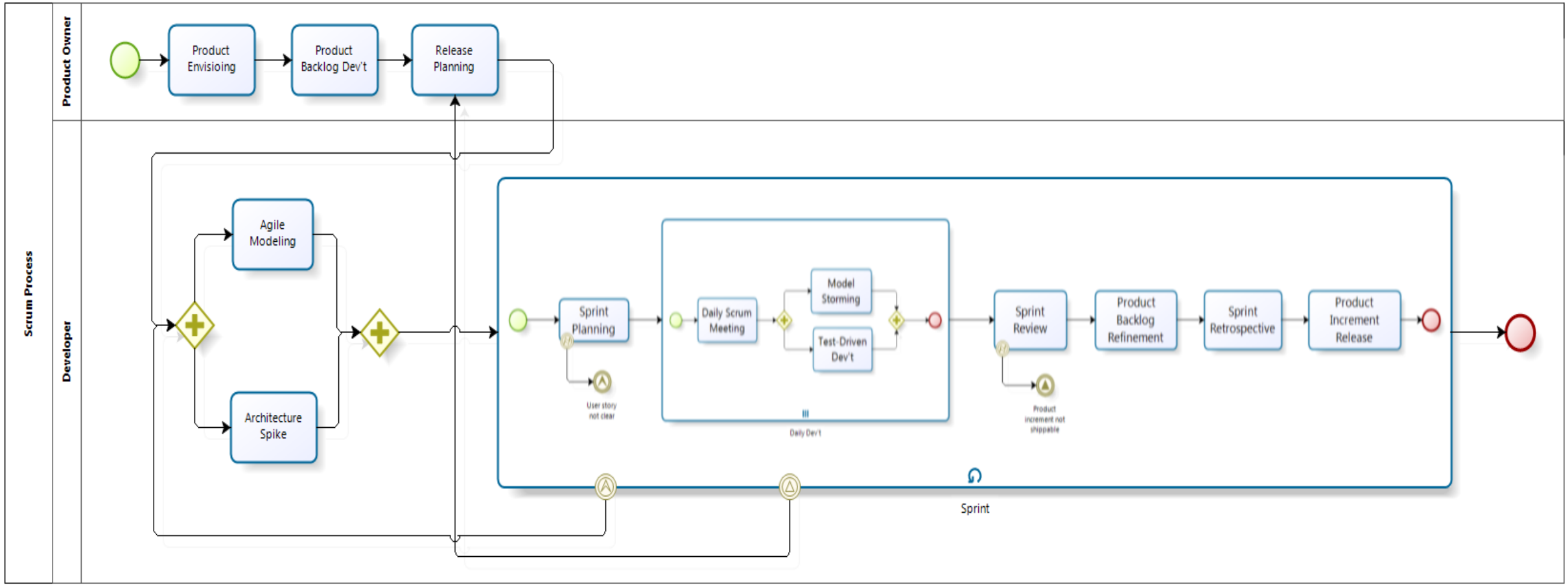
PO Product Owner	▪ Defines the features of the product (Product Backlog), decides on release date and content ▪ Prioritizes features according to market value ▪ Ensures that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next ▪ Is responsible for the profitability of the product (ROI) ▪ Can change features and priority every Sprint ▪ Accepts or rejects work results
SM ScrumMaster	▪ Ensures that the process is followed, and the team is fully productive ▪ Facilitates close cooperation across all roles and removes barriers ▪ Shields the team from external interferences ▪ Invites to daily scrum, iteration review and planning meetings
Team	▪ Cross-functional, seven plus/minus two members ▪ Has the right to do everything within the boundaries of the product groups guidelines to reach the iteration goal ▪ Organizes and manages itself and its work ▪ Reviews work results with the Product Owner

Scrum Project Management

- Release Planning by Product Owner (Product Manager)
  - Select a group of stories to be released together, and decide how many sprints are needed
  - Or select a release date and see how much can be done by then.
- Sprint Planning by Scrum Team
  - Select highest-value stories from the backlog that are to be developed during the next sprint.
  - Have developers estimate story points, and ensure that selected stories can be completed considering the team velocity.
- Daily Scrum meeting
  - At the meeting everyone will announce their plan for the day and then act on it.
  - When you slip your schedule, make a new plan instead of trying to catch up.
  - Changes to requirements are welcome because we have already decided we will be making new plans anyway.

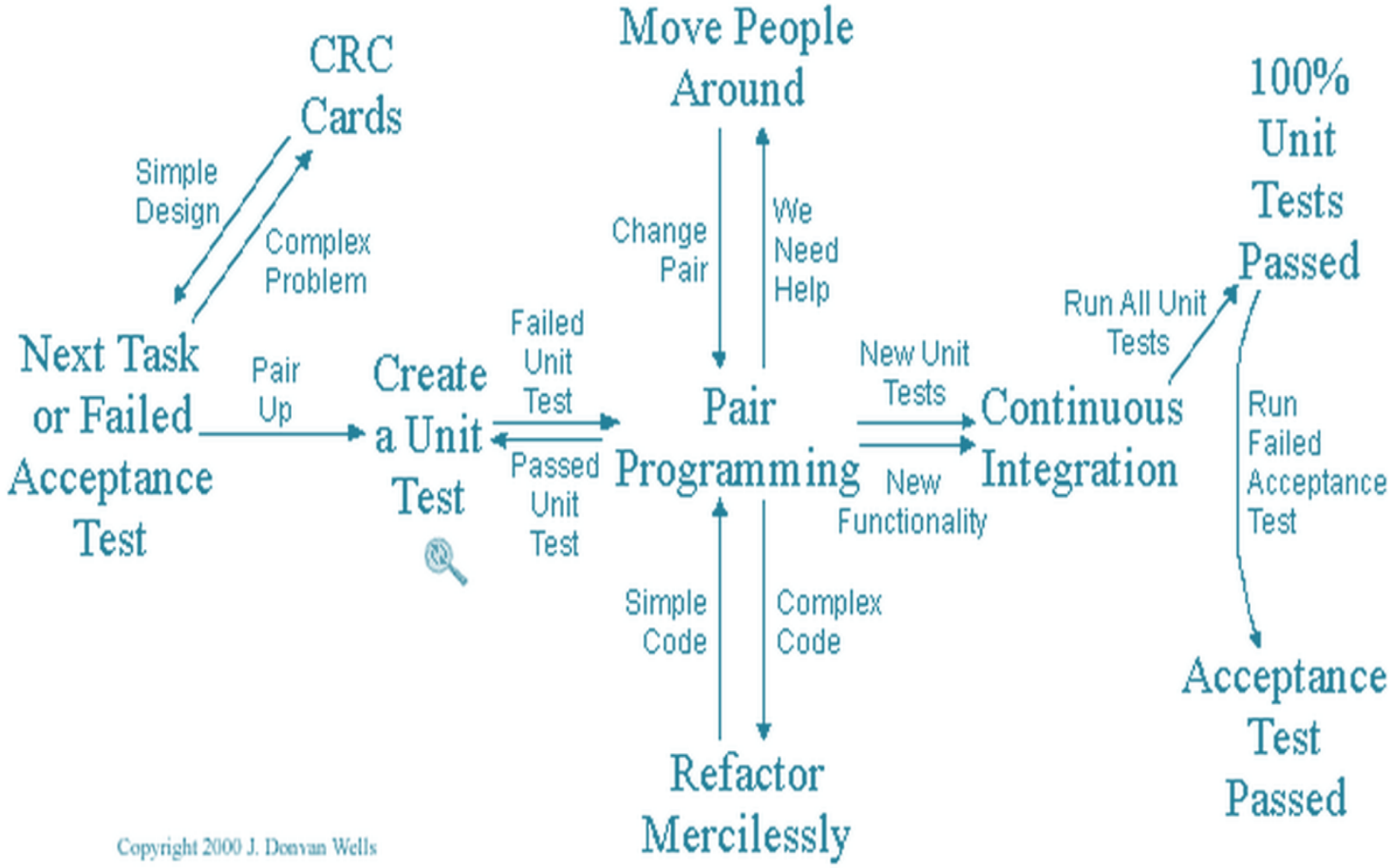


Scrum Process

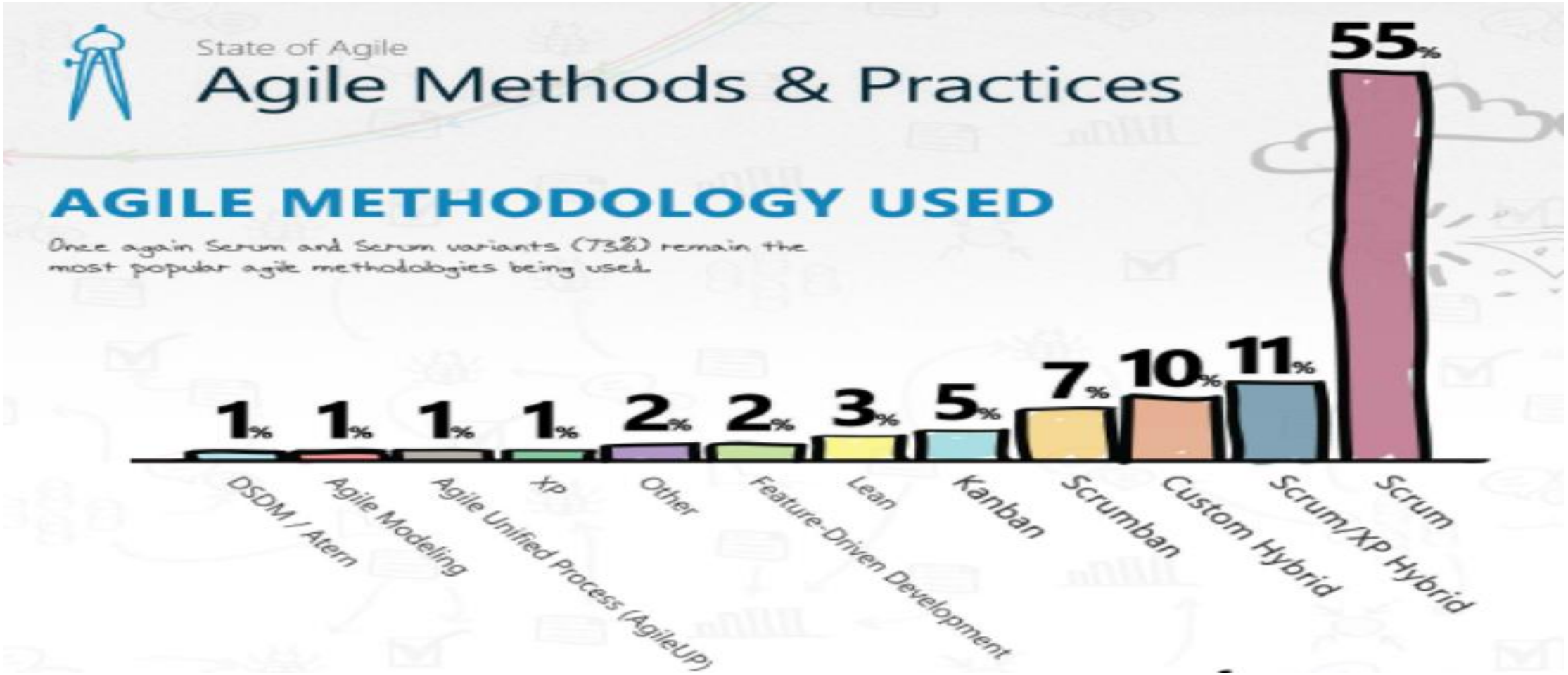


Extreme Programming (XP)

- CRC card
- Coding standard
- Unit test first
- Pair programming
- Refactoring
- Continuous integration
- Bug management
- Frequent acceptance tests



Agile Methods and Practices



VersionOne, 8th Annual State of Agile Survey, 2013.

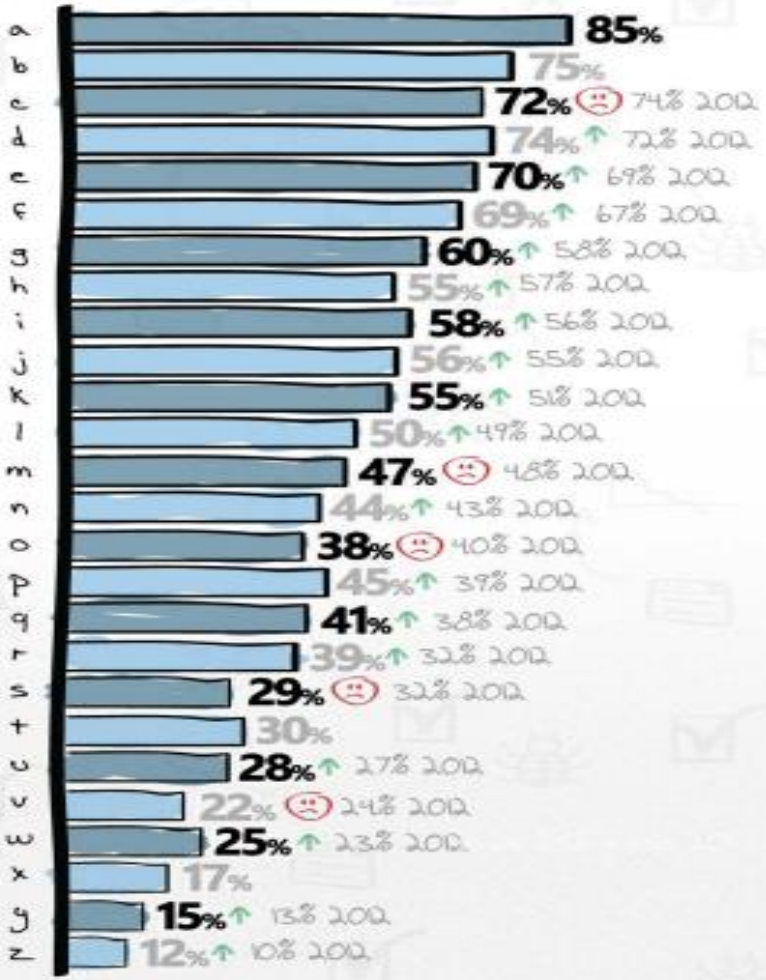
Agile Techniques

AGILE TECHNIQUES EMPLOYED

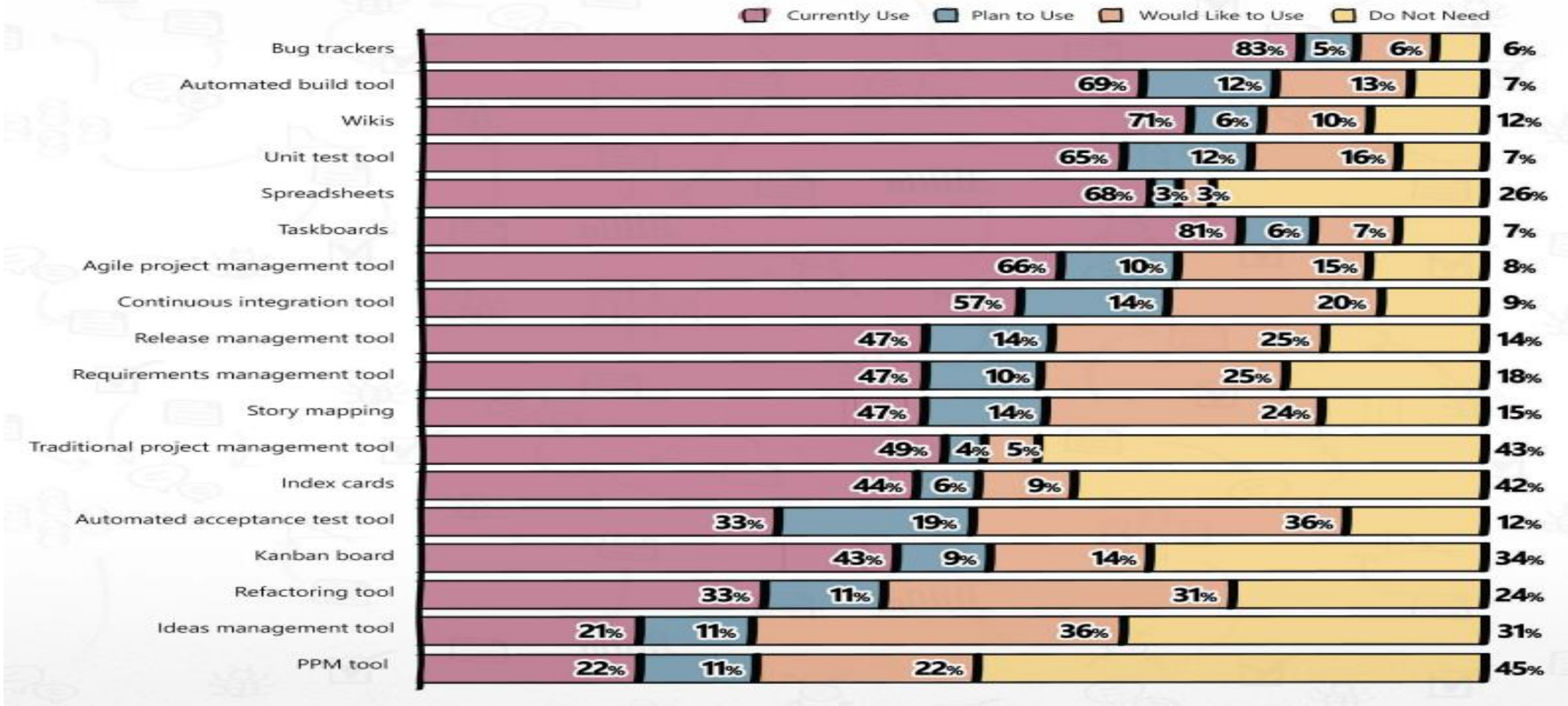
Respondents are making use of a wide variety of different agile management techniques. More than 85% practice Daily Standups, ¾ are using Iteration Planning and Retrospectives, and nearly the same proportion said they maintain Burndown charts. Over the last 2 years we've seen a 10% increase in the use of Retrospectives (from 64% in 2011 to 74% in 2013).

*Respondents were able to select multiple options.

- a Daily Standup
- b Iteration Planning
- c Unit Testing
- d Retrospectives
- e Release Planning
- f Burndown/ Team-Based Estimation
- g Velocity
- h Coding Standards
- i Continuous Integration
- j Automated Builds
- k Dedicated Product Owner
- l Integrated Dev/QA
- m Refactoring
- n Open Workarea
- o TDD
- p Digital Taskboard
- q Story Mapping
- r Kanban
- s Collective Code Ownership
- t Pair Programming
- u Automated Acceptance Testing
- v Analog Taskboard
- w Continuous Deployment
- x Agile Games
- y Cycle Time
- z BDD

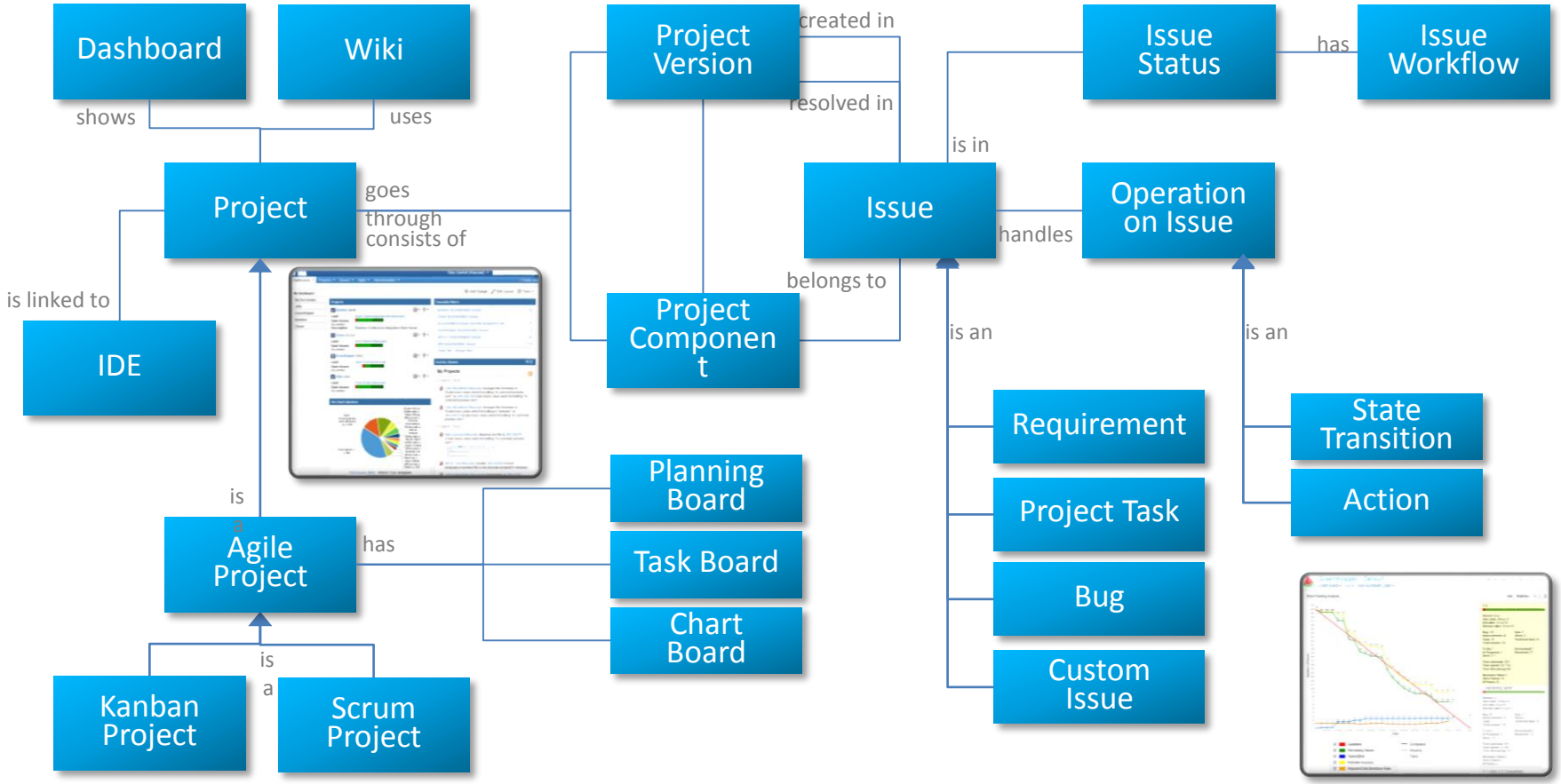


Agile Tools



Agile Project Management Tool

- JIRA, sitting at the center of your team, connects the people and the work being done.
- Track requirements, tasks and bugs, link those issues to related source code, plan agile development, monitor activity, report on project status, and more.

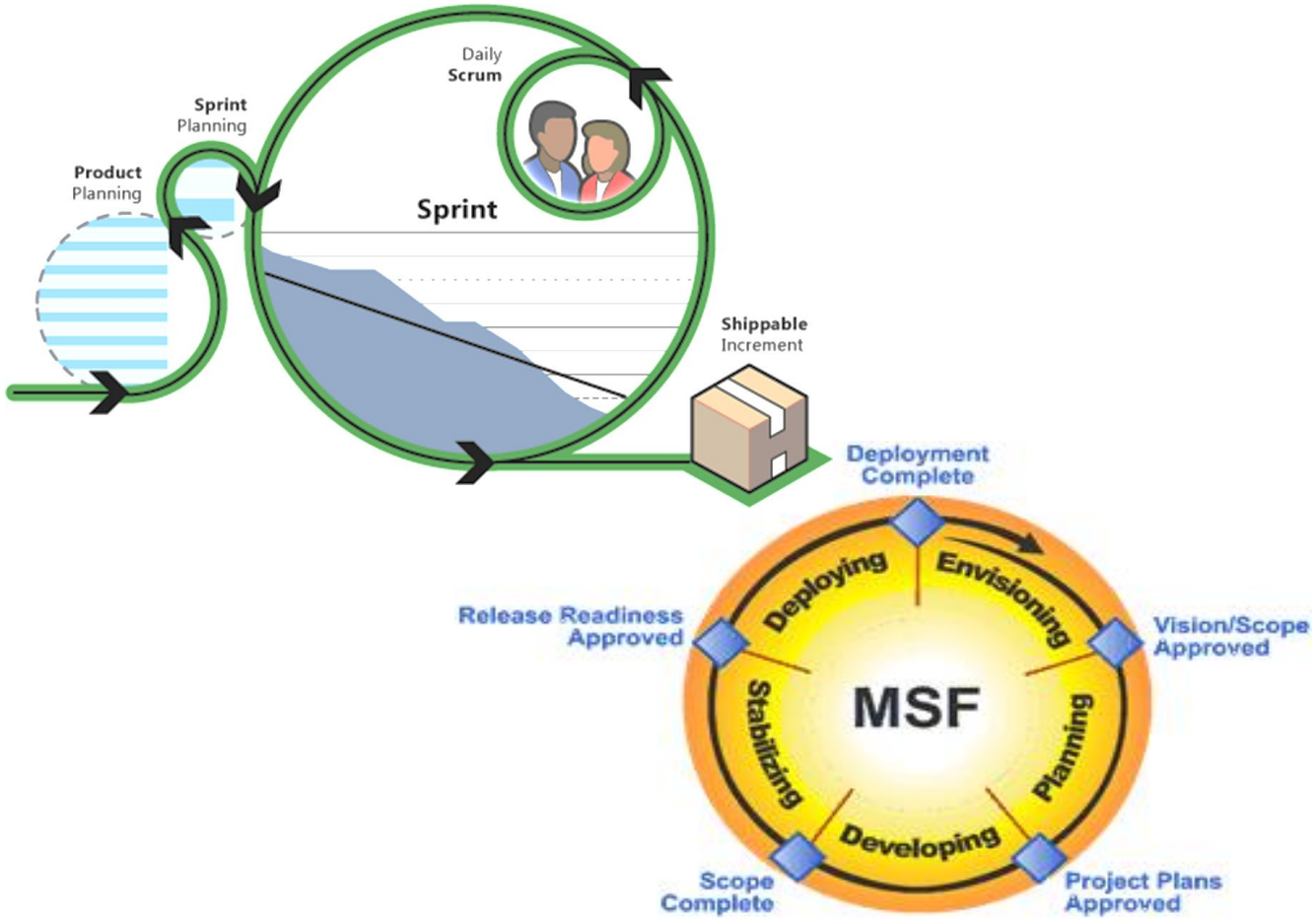


Atlassian, Jira 5.0 Documentation, 2012.

Agile Software Development @ Microsoft

Today Microsoft applies agile software development process consisting of the following practices:

- Basic Scrum
  - 4-week or 2-week sprint with daily scrum
  - JIT design of features using CRC
  - Spike for large or unknown features prior to including in the sprint backlog
- Planning Poker
  - Planning Poker (a.k.a. Wideband Delphi) played in the sprint planning by an extended development team to estimate person-hours
  - Planning Poker sometimes leads to a deadlock when the product owner did not fully described the feature to be delivered.



L. Williams, G. Brown, A. Meltzer and N. Nagappan, "Scrum + Engineering Practices: Experiences of Three Microsoft Teams," *International Symposium on Empirical Software Engineering and Measurement*, Banff, Canada, 2011.

Agile Software Development @ Microsoft

- Requirement Analysis, Architecture Design and Prototyping
  - In such cases, upfront work is requested to the product owner that may include fully defining small, user-visible, user-valued feature requirements, high-level architectural analysis, preliminary user interface design, and a spike.
- XP: Test-Driven Development
  - All public methods must have documentation.
  - Each developer check-in at least daily.
  - Each check-in initiates an automated build.
  - Each build entails automated unit tests and associated test coverage computation.
  - Unit test coverage must be at least 80%. Due to “Done or not done” criteria of scrum, developers often develop and demo only the happy path of a user story in the scrum review. High coverage is needed to force tests to execute alternative flows and error handling.
  - Build must complete with no errors or warnings on the highest level.
- QA
  - Senior developers conduct peer reviews of architecture diagrams and of code when adding new features. When code is checked in, the reviewer(s) names are entered into the code review tool.
  - All code must not have any static analysis errors or warnings.

L. Williams, G. Brown, A. Meltzer and N. Nagappan, “Scrum + Engineering Practices: Experiences of Three Microsoft Teams,” *International Symposium on Empirical Software Engineering and Measurement*, Banff, Canada, 2011.