Rational. software

IBM

Redbooks Paper

Michele Galic
Bruce Macisaac
Dan Popescue

# Using a Single Business Pattern with the Rational Unified Process (RUP)

IBM® Rational® Unified Process®, or RUP®, is a commercial software and systems development process framework (Figure 1 on page 2) with key principles and guidance for iterative software and systems development. RUP is composed of:

- Best practices: RUP includes a library of best practices for software and systems engineering, covering everything from project management to detailed test guidance.
- Method delivery tools: RUP is delivered using Web technology, which means that it can be integrated with other software and systems development tools, making it easily accessible to developers.
- Method configuration tool: RUP is made up of components and plug-ins that you can select and configure to meet the needs of different kinds of projects using IBM Rational Method Composer.
- Method authoring tool: An organization can extend or modify RUP by creating its own plug-ins with Rational Method Composer.
- Community and marketplace: The RUP section of the developerWorks® site provides a place for process engineers in the software and systems development community to share their method extensions.

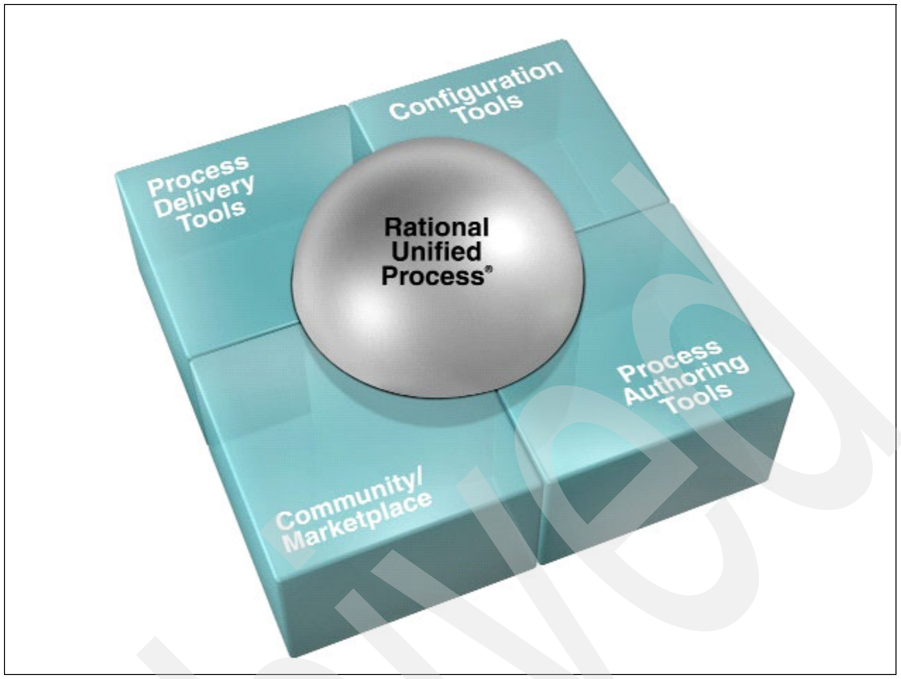


*Figure 1 Rational Unified Process*

RUP is included in Rational Method Composer, which provides tools for method delivery, configuration, customization, and authoring.

## Key principles for business-driven development

The following principles articulate best practices for the broader life cycle of continuously evolving systems, where the primary evolving element is software:

- Adapt the process.

  It is critical to fit the development process to the needs of the project. More is not better; less is not better. Instead, the amount of ceremony, precision, and control present in a project must be tailored according to a variety of factors. These factors include the size and distribution of teams, the amount of externally imposed constraints, and the phase that the project is in.

- Balance competing stakeholder priorities.

  Balancing often conflicting business and stakeholder needs and balancing custom development versus asset reuse in the satisfaction of these needs are critical activities when you develop a new system or improve an existing one.

Finding the correct trade-off between competing forces is the key to building successful applications.

- Collaborate across teams.

  Fostering optimal project-wide communication is very important in software development. This is achieved through proper team organization and the setting up of effective collaborative environments.

- Demonstrate value iteratively.

  An iterative process makes it possible to easily accommodate change, to obtain feedback and factor it into the project, to reduce risk early, and to adjust the process dynamically.

- Elevate level of abstraction.

  Complexity is a central issue in software development. Elevating the level of abstraction helps reduce complexity and the amount of documentation that is required by the project. This can be achieved through reuse, high-level modeling tools, and early stabilization of the architecture.

- Focus continuously on quality.

  An iterative process is particularly suited to achieving quality because it offers many measurement and correction opportunities. Improving quality is not simply "meeting requirements" or producing a product that meets user needs and expectations. Rather, quality also includes identifying the measures and criteria that demonstrate that quality has been achieved and implementing a process to ensure that the product has achieved the desired degree of quality, and that this can be repeated and managed.

## The architecture of RUP

Figure 2 on page 4 shows the overall architecture of RUP, which has two dimensions:

- The horizontal axis represents time and shows the life cycle aspects of the process as it unfolds.
- The vertical axis represents disciplines that logically group activities by nature.

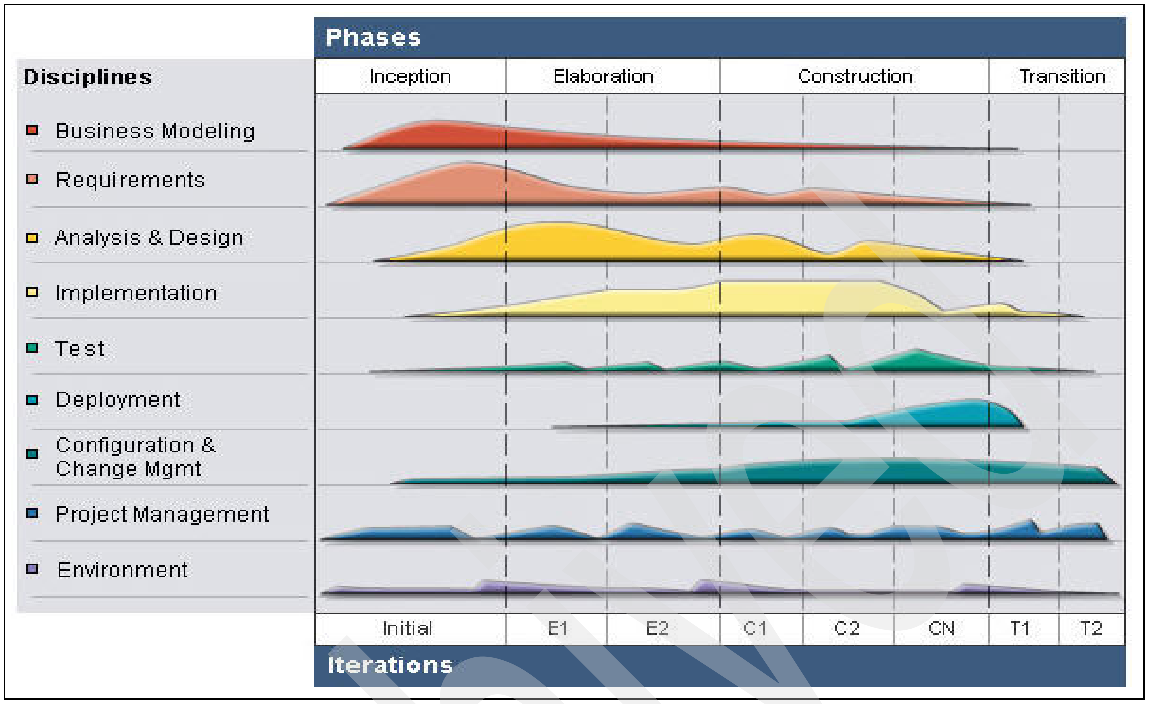


*Figure 2 Architecture of RUP*

The graph shows how the emphasis varies over time. For example, in early iterations you spend more time on requirements; in later iterations, you spend more time on implementation.

RUP follows a standard meta-model[1] for describing software processes that includes the following concepts:

- Work products: What is produced
- Tasks: How to perform the work
- Roles: Who performs the work

## Work products

Work products can take various shapes or forms, including:

- A model, such as the use-case model or the design model, that contain model elements (sub-artifacts) such as design classes, use cases, and design subsystems

[1] See the OMG Standard Software Process Engineering Meta-Model, available from `http://www.omg.org`.

- Databases or other types of tabular information repositories (for example, spreadsheets)
- Source code and executables
- Various types of documents (for example, a specification document, such as the requirements specification, or a plan document, such as the software development plan)

These work products are examples of “artifacts,” the tangible results of doing work. Work products can also take the form of “deliverables,” which are packages of artifacts, or “outcomes,” which are intangibles, such as “team has attended security training.”

## Roles

A role defines a set of related skills, competencies, and responsibilities of an individual or a set of individuals working together as a team within the context of a software engineering organization.

Note that roles are not individuals; instead, roles describe responsibilities. An individual typically takes on several roles at one time and frequently changes roles over the duration of the project.

## Tasks

A task describes a unit of work. Every task is performed by specific roles. Tasks are usually defined as a series of steps that involve creating or updating one or more artifacts.

Some examples of tasks are:

- *Find actors and use cases*: A task performed by the system analyst role to identify high-level functional requirements in terms of actors and use cases
- *Describe distribution*: A task performed by the software architect role to describe software distribution across multiple processors

## Process concepts

A process describes the sequence of work to be performed, structuring tasks into higher level groupings called *activities*. Reusable chunks of process are referred to as *capability patterns*, while complete processes, ready to be applied on a project, are called *delivery processes*.

Figure 3 is a screen capture from the delivery process called “Classic RUP Lifecycle” that shows the breakdown into phases and iterations and includes a diagram of the sequence of activities in the inception phase.

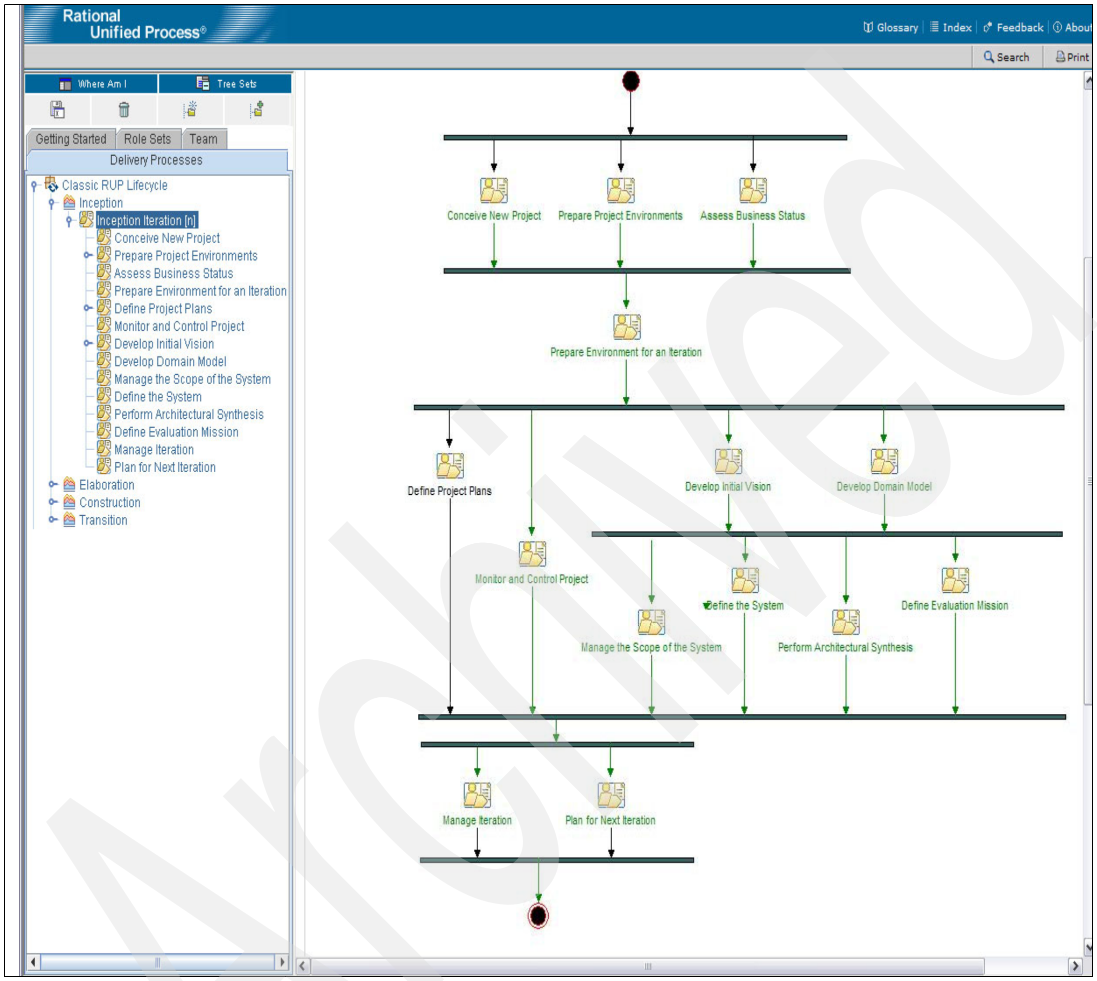


*Figure 3 Inception iteration*

## Phases

Another key concept in RUP is phases. Phases provide project milestones that ensure that iterations make progress and converge on a solution, rather than iterate indefinitely. Phases and iterations are special activities for which specific attribute values have been set with predefined values.

The phases of RUP are:

- Inception: The scope has been defined, main set of requirements have been captured, cost/schedule estimates are in place, risks have been identified and mitigation strategies exist for each one.
- Elaboration: The software architecture is established and validated with an executing architectural version of the system.
- Construction: The focus is on completing the development of the system.
- Transition: The software is deployed and made acceptable to its users.

The objectives of each phase are achieved by executing one or more iterations in the phase. Each phase concludes with a milestone at which the phases are assessed to determine if the project has achieved the specified objectives of the phase. The project cannot move to the next phase until these objectives are achieved. See Figure 4.

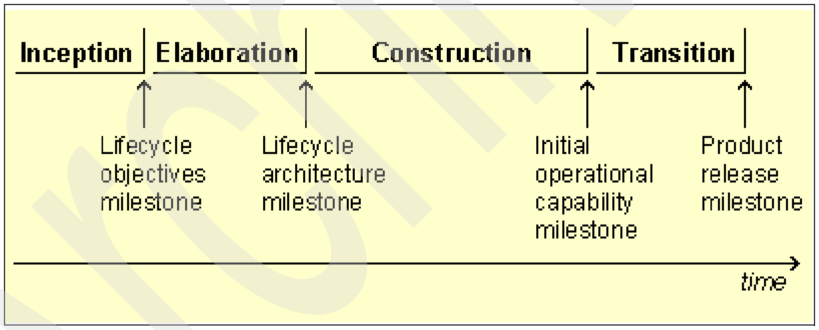

Inception
Elaboration
Construction
Transition
Lifecycle objectives milestone
Lifecycle architecture milestone
Initial operational capability milestone
Product release milestone
time

*Figure 4 The phases and milestones of a project*

The objectives of each phase are described in more detail in the next section.

## Disciplines

RUP tasks are organized into disciplines, which include a process (to be precise, a capability pattern) that shows the typical workflow for the discipline throughout a project. Discipline workflows provide a focused perspective for learning about the process.

Figure 5 shows the workflow for the analysis and design discipline.

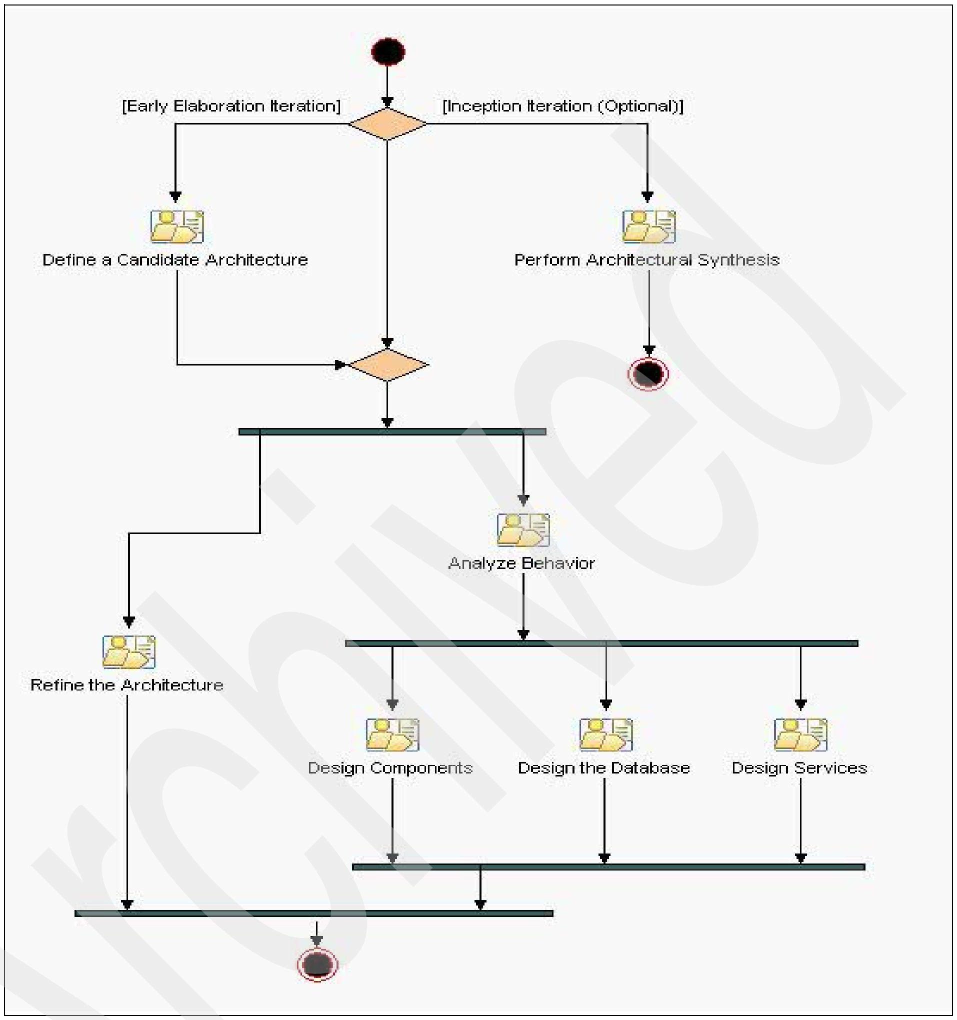

[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Refine the Architecture
Design Components
Design the Database
Design Services

*Figure 5 Analysis and design discipline workflow*

## Using IBM Patterns for e-business during inception

Key goals of the RUP inception phase are:

- A vision that establishes the key needs
- A business case that justifies the investment
- Initial budget and schedule for the project for moving forward

Table 1 is an overview of how IBM Patterns for e-business apply to RUP during the inception phase. We focus on those parts of RUP for which the assets of Patterns for e-business are of most value by accelerating the development of the associated artifacts. This is only a small subset of a complete RUP-based software development process.

*Table 1 Inception and Patterns for e-business*

| **RUP elements: Disciplines, activities, tasks and artifacts** | **Added value of Patterns for e-business** |
|---|---|
| Discipline: Environment<br><br>Activity: Prepare environment for project<br>Task: Prepare guidelines for the project<br>Artifact: Project-specific guidelines | Patterns for e-business are one of the intellectual assets that should be considered during the initial identification of useful project guidelines. |
| Discipline: Business modeling<br><br>Activity: Explore process automation<br>Artifact: Business vision | You can use Patterns for e-business as inspiration for identifying and characterizing business processes. Also, analyzing the business goals and business processes for the business as a whole provides valuable input to deciding how and what to automate. |
| Discipline: Requirements<br><br>Activity: Define the system<br>Task: Develop vision<br>Artifact: Vision | The need to collect the necessary information for navigating the asset catalog creates an opportunity for a focused analysis of the requirements.<br><br>The navigation of the Patterns for e-business asset catalog starts from a broad categorization of business requirements. Not only do these guide the asset selection process, but they also offer additional insights into the overall business area and the main business drivers. |
| Discipline: Analysis and design<br><br>Activity: Perform architectural synthesis<br>Task: Architectural analysis<br>Artifact: Architectural proof-of-concept<br>Artifact: Software architecture document (inception draft) | Reference architectures, of which Patterns for e-business are an example, contribute significantly to the definition of an architectural proof-of-concept. |
| Discipline: Project management<br><br>Activity: Evaluate project scope and risk<br>Activity: Plan the project<br>Artifact: Business case<br>Artifact: Software development plan | The architectural proof-of-concept, created using Patterns for e-business, provides valuable input to the business case and project plans. |

We now discuss each of the elements outlined in Table 1 in the context of the First Financial case study. First Financial, a fictional company used for this example, is a large financial services company that wants to deliver better services to their customers and to do so more efficiently.

## Business modeling

Business modeling (Figure 6) is an optional discipline in RUP that looks at the broader scope of the business. It is used to understand the current business processes and determine how they can be improved. Identifying opportunities for automation is one way you can improve business processes.

You can perform business modeling as part of a project to help gain better understanding of the business context or as a separate project that spawns multiple software development projects.

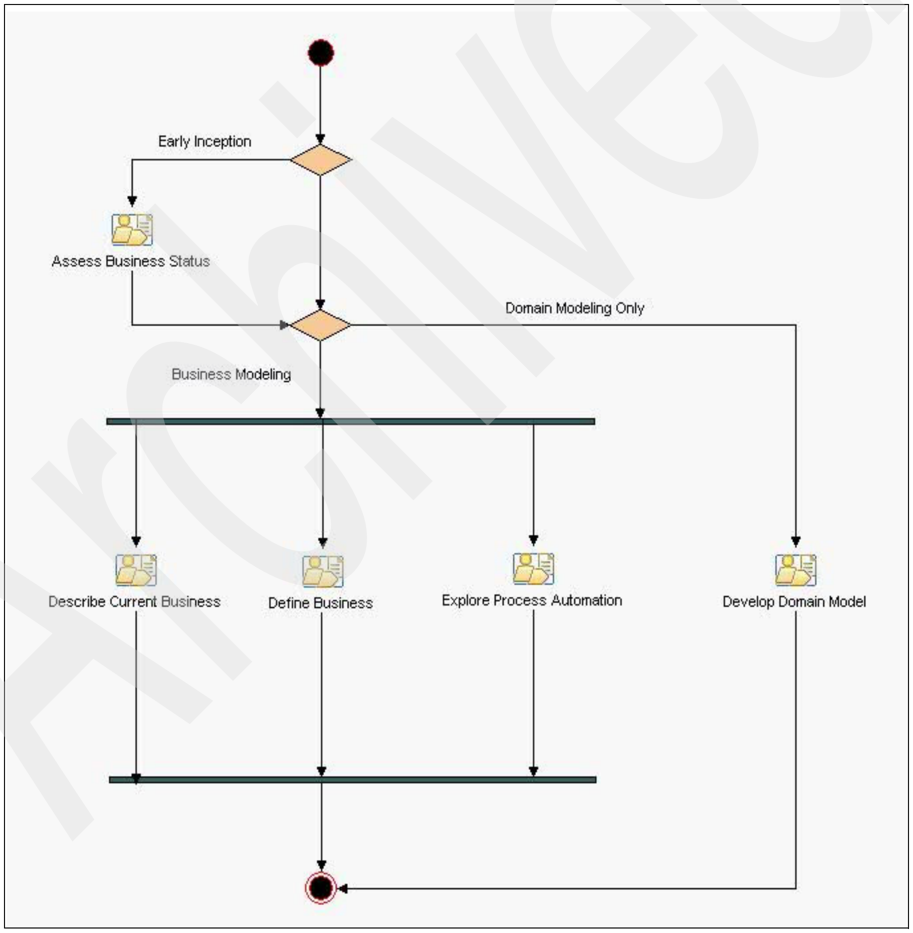


*Figure 6 Business modeling discipline*

In the case of First Financial, business modeling can describe how services are currently provided to customers and identify opportunities for improvement. One way of identifying possible improvements is to look for opportunities where particular business patterns (from the Patterns for e-business repository) are applicable.

For example, after modeling some key First Financial business processes, we could look for processes where the self-service business pattern might be applied. The self-service business pattern addresses direct interactions between interested parties and a business.

We recognize that the business processes involved in applying for, approving, and managing credit card accounts fit this pattern and that automation in this area would be in line with the business goals of providing better services with more efficiency.

We decided that Web enablement of the existing credit card system is a first step toward giving customers access to some of the core First Financial applications that cross multiple access channels. These core applications are supported by IBM @server® zSeries® mainframes and maintained by the predominantly CICS®-skilled First Financial staff.

## Environment

Patterns for e-business are one of the assets to consider during the initial tailoring of processes for an e-business project. Process tailoring is part of the prepare environment for project activity (Figure 7 on page 12).

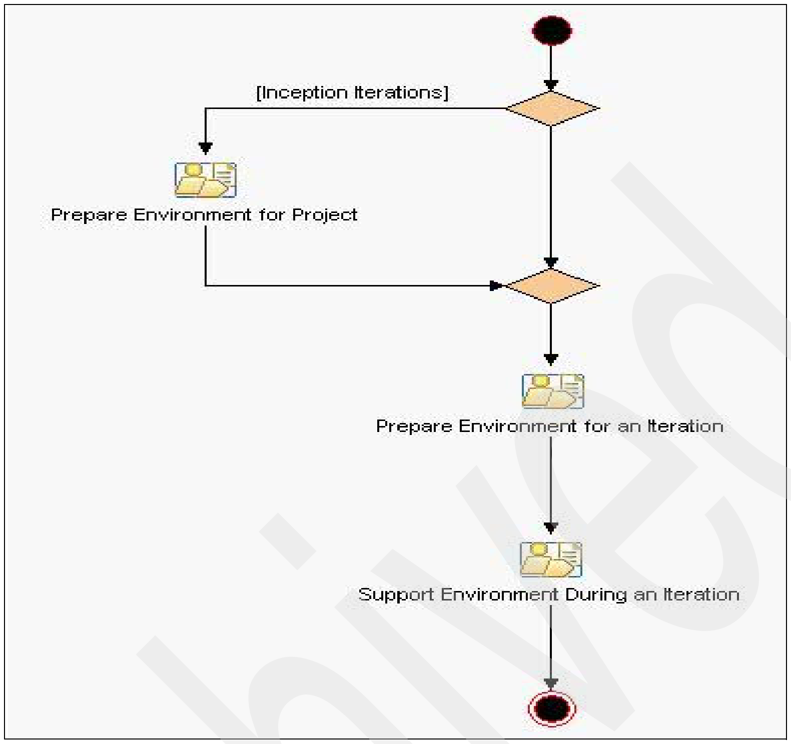

[Inception Iterations]
Prepare Environment for Project
Prepare Environment for an Iteration
Support Environment During an Iteration

*Figure 7 Environment discipline*

At this stage, Patterns for e-business are evaluated for relevance to the project and process. The recommendation to use Patterns for e-business and any additional guidance related to using the patterns can be incorporated into a RUP configuration as additional guidelines associated to the activities and tasks that are identified in this paper. See the prepare guidelines for the project task for more details.

Because First Financial is embarking on an e-business project, we decided to use Patterns for e-business for a fast-path definition to a suitable and proven solution. Note that patterns are rarely a perfect fit; some customization is usually needed to address the unique requirements of each project. Patterns, even when they fit, are only part of the solution. The specific functionality for the project still needs to be defined, designed, and delivered.

## Requirements

A key goal for the requirements discipline (Figure 8) during the inception phase is to baseline a vision for the project. The RUP activities of *analyze the problem*, *understand stakeholder needs*, and *define the system* indicate how to describe the problem to be solved, identify the key stakeholder needs, and capture the key functional and non-functional requirements of the system.

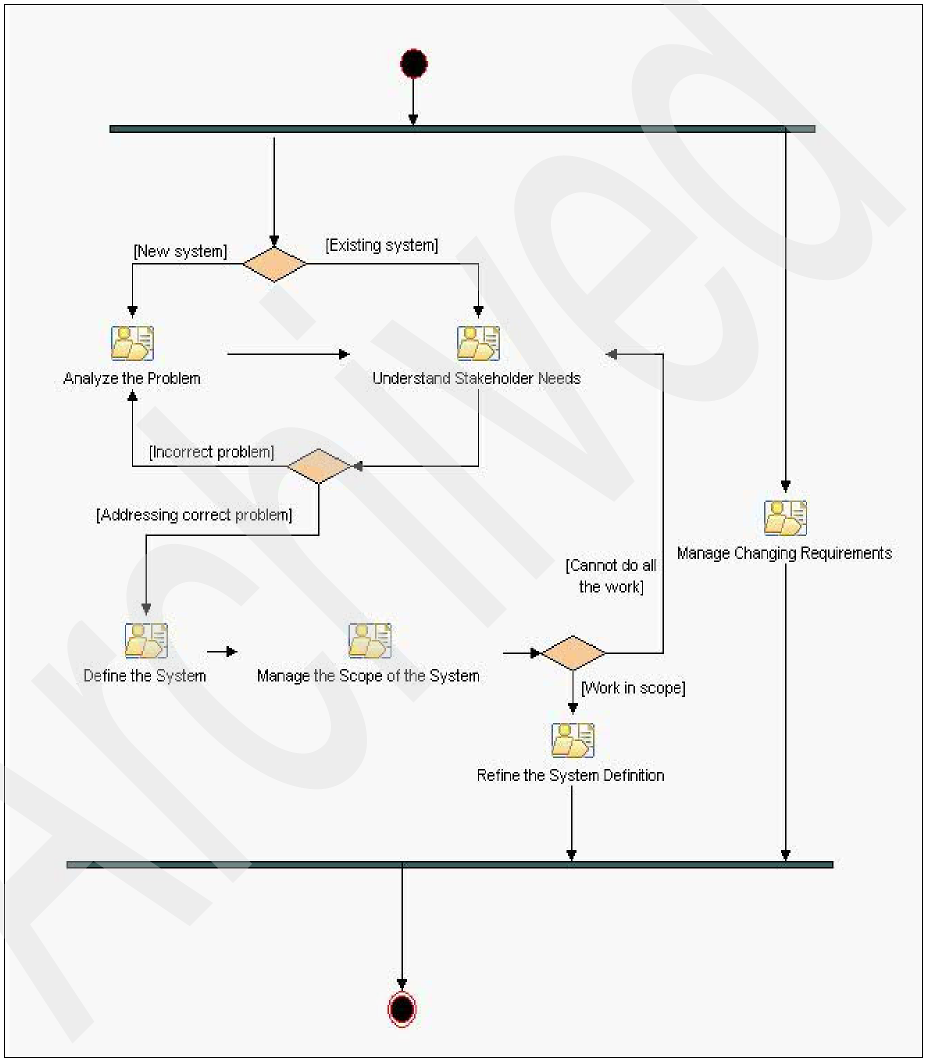

[New system]
[Existing system]
Analyze the Problem
Understand Stakeholder Needs
[Incorrect problem]
[Addressing correct problem]
Manage Changing Requirements
[Cannot do all the work]
Define the System
Manage the Scope of the System
[Work in scope]
Refine the System Definition

*Figure 8 Requirements discipline*

The First Financial goal is to open up the existing enterprise transactions and data to external customers. The first step toward meeting this goal is the Web enablement of their existing credit card system.

The vision identifies the top business issues, challenges, or priorities for the enterprise and provides an insight into the trade-offs made and priorities set by the client. These drivers and priorities are an important input to the application pattern selection process (described later in this paper), which is based, among other things, on an analysis of the key business and IT drivers.

The First Financial business objectives that flow down from the business modeling efforts are:

- Reduction of time to market
- Tactical focus on only supporting the Web channel
- Strategic focus on supporting multiple access channels such as call center and telephone access
- Make the most of past investments and the existing CICS skill base

Figure 9 shows the use cases and actors for First Financial summarized in a system context diagram.

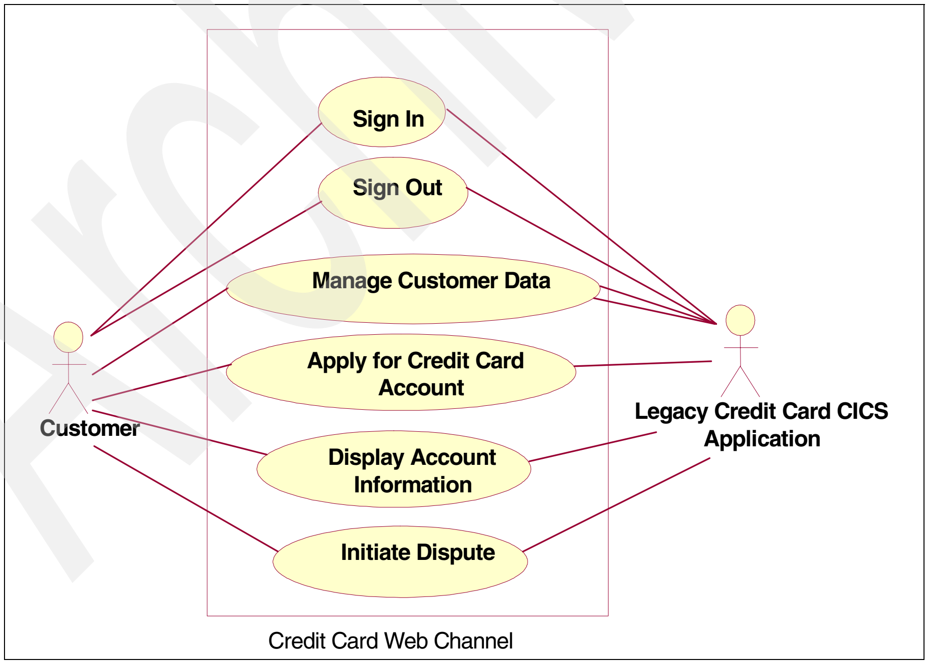

Sign In
Sign Out
Manage Customer Data
Apply for Credit Card Account
Display Account Information
Initiate Dispute
Customer
Legacy Credit Card CICS Application
Credit Card Web Channel

*Figure 9 System context diagram: First Financial*

Supplementary specifications include:

- A requirement to access an existing CICS-based application
- A strong emphasis on the privacy of the exchanges between the front end and the existing applications and the authentication of a client before any access to an existing application is allowed
- A preference for Microsoft® Windows® 2000/IBM xSeries® servers to host the Web infrastructure, because there is a current installed base of Windows 2000/xSeries and zSeries servers

For First Financial, we identified the opportunity for automation from the self-service business pattern. However, it is equally valid and common to start with requirements and determine what pattern or patterns fit already specified software requirements. In this case, an analysis of the high-level business requirements that were described earlier and captured in the vision suggests that the Self-Service business pattern is applicable. First Financial is interested in extending an existing credit card application through a Web channel to its customer and user base. The self-service business pattern is applicable because users need to interact directly with existing data (Figure 10).

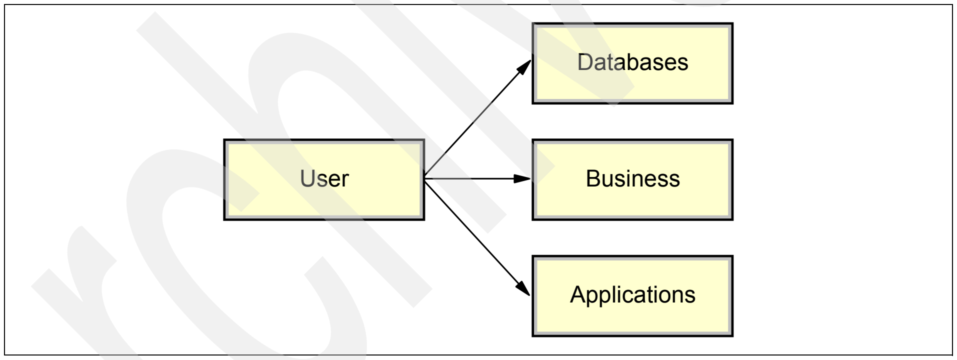


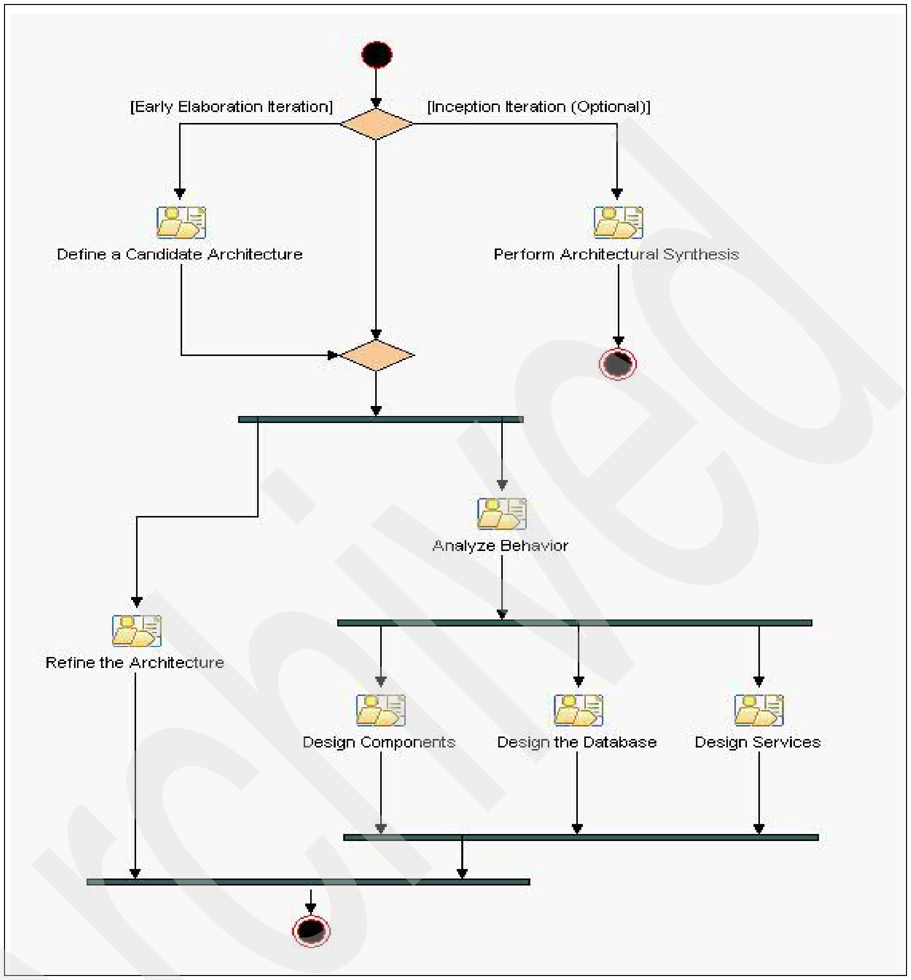
*Figure 10 Self-service, user-to-multiple applications*

## Analysis and design

In the inception phase, the analysis and design discipline is optionally performed, as necessary, to convince stakeholders that the project is worth investing in. Specifically, the perform architectural synthesis activity can be employed to create an architectural proof-of-concept.

Patterns for e-business can contribute significantly to creating an architectural proof-of-concept. The navigation of the Patterns for e-business asset catalog is guided by the requirements collected as part of the requirements discipline.

Figure 11 shows a diagram of the analysis and design discipline.



[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Refine the Architecture
Design Components
Design the Database
Design Services

*Figure 11   Analysis and design discipline*

Note that the architectural proof-of-concept is elaborated to the extent that is required to convince stakeholders that the project should proceed to the next phase.

For many systems, it is sufficient to demonstrate that there are one or more suitable solution patterns that match the business pattern that is identified from the requirements. In other cases, there might be particular feasibility risks that require more design exploration, such as:

- ► A list of additional technologies (frameworks, patterns, executable architectures) that seem appropriate to the solution

- A sketch of a conceptual model of a solution using a notation such as UML
- A simulation of a solution
- An executable prototype to address design beyond what the application patterns express, produce a simulation of a solution, or create an executable prototype

It is not uncommon to find that multiple patterns and assets are relevant to the proposed solution. In this case, additional work is required to integrate the different patterns and assets to produce a coherent proof-of-concept.

The software architect steps through the architectural analysis task. The first step is to develop an architecture overview that is ideally based on an existing reference architecture. Patterns for e-business are immediately applicable from the Patterns for e-business Web site:

`http://www.ibm.com/developerworks/patterns`

The Web site guides the architect through a series of steps described in the sections that follow.

## Step 1: Selection of a business pattern

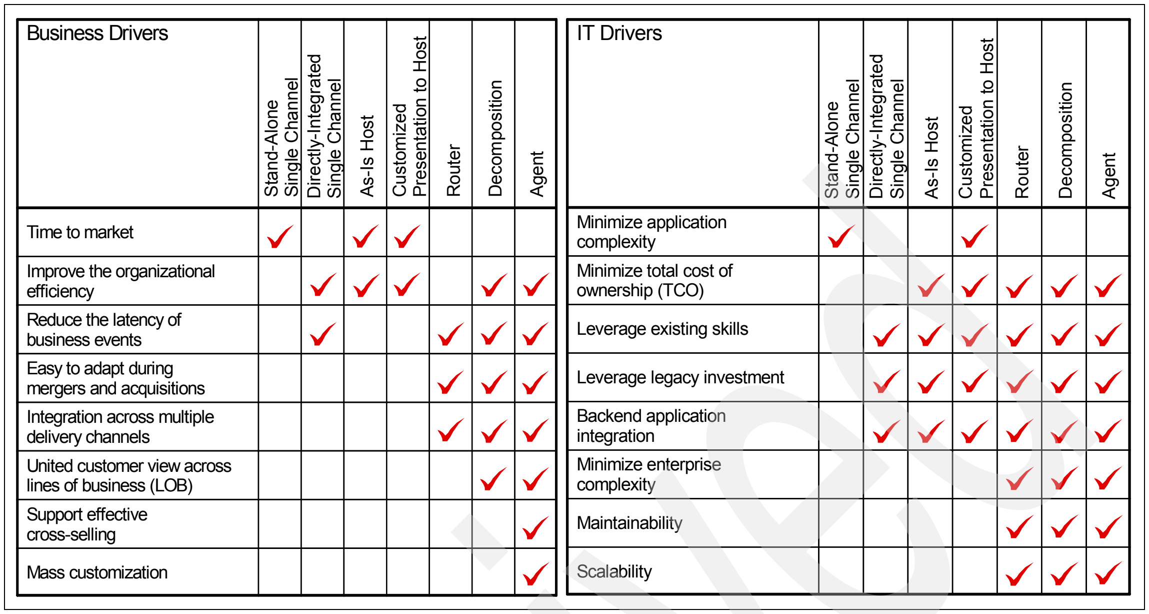
In the case of First Financial, the self-service business pattern has already been identified as applicable.

## Step 2: Selection of an application pattern or patterns

The application pattern focuses on the application-level functionality that is required to support the identified business pattern or patterns.

First Financial plans to provide access to its existing back-end systems through different channels (initially there will only be a Web channel, but the call center and telephone have been identified as future extensions).

Figure 12 on page 18 presents the application patterns (shown as column headers) for the self-service business pattern and the typical business and IT drivers (shown as row headers) that drive the application pattern selection.



| Business Drivers | Stand-Alone Single Channel | Directly-Integrated Single Channel | As-Is Host | Customized Presentation to Host | Router | Decomposition | Agent |
|---|---|---|---|---|---|---|---|
| Time to market | ✓ | | ✓ | ✓ | | | |
| Improve the organizational efficiency | | ✓ | ✓ | ✓ | | ✓ | ✓ |
| Reduce the latency of business events | | ✓ | | | ✓ | ✓ | ✓ |
| Easy to adapt during mergers and acquisitions | | | | | ✓ | ✓ | ✓ |
| Integration across multiple delivery channels | | | | | ✓ | ✓ | ✓ |
| United customer view across lines of business (LOB) | | | | | | ✓ | ✓ |
| Support effective cross-selling | | | | | | | ✓ |
| Mass customization | | | | | | | ✓ |

| IT Drivers | Stand-Alone Single Channel | Directly-Integrated Single Channel | As-Is Host | Customized Presentation to Host | Router | Decomposition | Agent |
|---|---|---|---|---|---|---|---|
| Minimize application complexity | ✓ | | | ✓ | | | |
| Minimize total cost of ownership (TCO) | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| Leverage existing skills | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Leverage legacy investment | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Backend application integration | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Minimize enterprise complexity | | | | | ✓ | ✓ | ✓ |
| Maintainability | | | | | ✓ | ✓ | ✓ |
| Scalability | | | | | ✓ | ✓ | ✓ |

*Figure 12 Business and IT drivers of the self-service business pattern*

The software architect selects the router application pattern because it provides a structure for applications that require intelligent routing of requests from multiple delivery channels to one of multiple back-end applications. The router application pattern supports multiple delivery channels connecting to one or more existing back-end applications through a router tier. This router tier can route a single message to one or more applications in the same business process. This is in line with the First Financial requirements. The router tier can process a single request and access the required existing systems to produce a response, which, in this case, are the rating and underwriting systems. Figure 13 shows an example of the router application pattern. The selection of an application pattern leads to a set of specific runtime patterns.

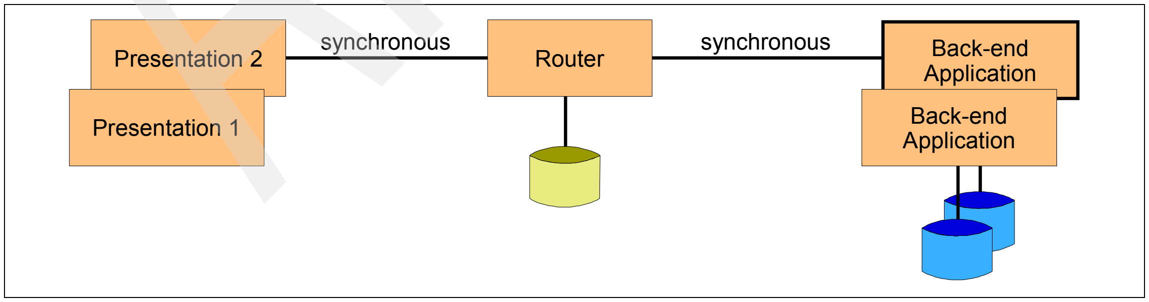


*Figure 13 Self-Service::Router application pattern*

## Step 3: Selection of a runtime pattern or patterns

The runtime pattern focuses on the technical functionality that is needed to support an application pattern.

The software architect selects the basic router runtime pattern because the combined presentation and application server is judged to be sufficient. This is validated in the elaboration phase.

In the router application pattern, the router tier serves as an integration point for delivery channels in the presentation tier, allowing access to individual back-end applications. Figure 14 shows a runtime pattern for *Self-Service::Router*. In this runtime pattern, which supports the router application pattern, the functions of the router tier are performed by an integration server.

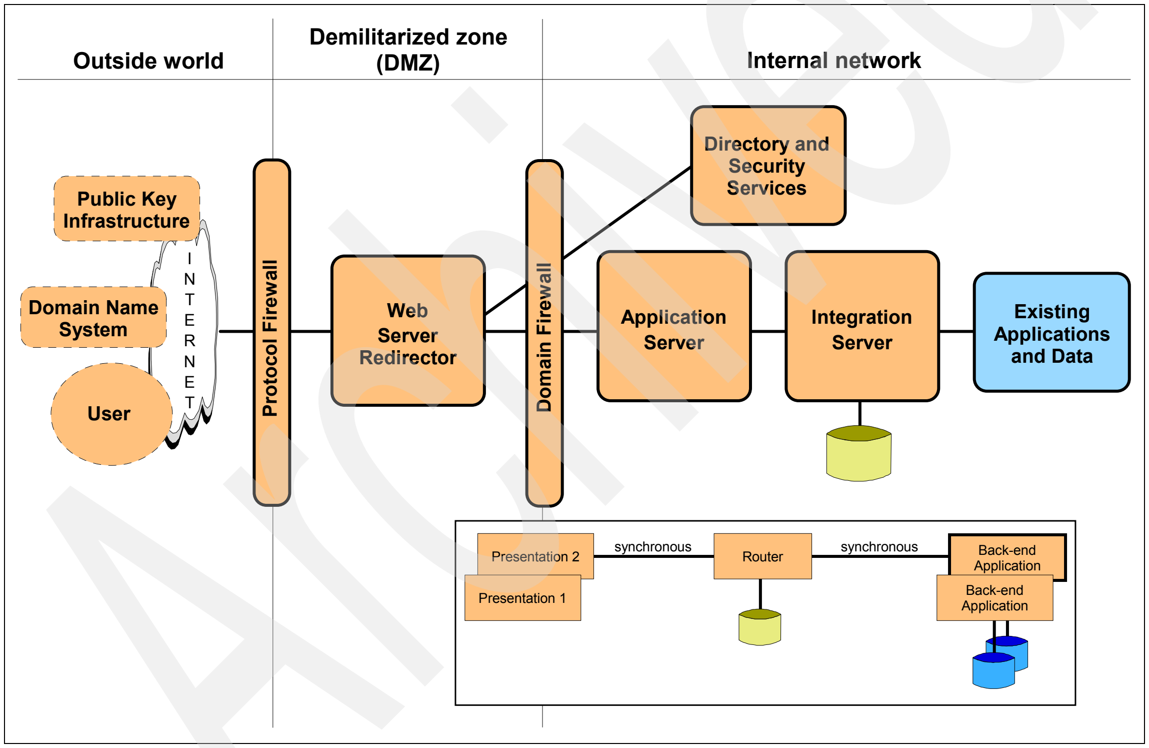


*Figure 14   Self-Service::Router runtime pattern*

Each of the nodes in this diagram is described in more detail on the Patterns for e-business Web site. Refer to the following URL for more information:

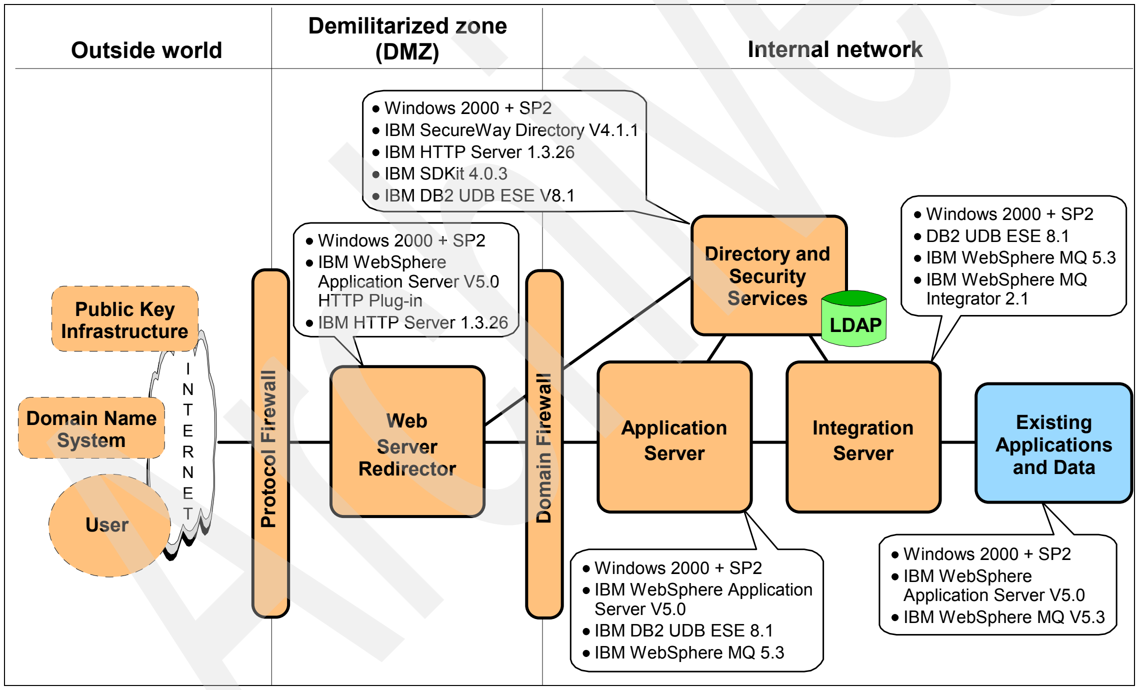
http://www.ibm.com/developerworks/patterns/u2b/at5-runtime.html

**Note:** Runtime patterns indicate the zones where the execution of certain specified implementation components are to take place and where data is to be made available. The precise determination of the involved locations (given that a zone can span several locations) and the specification of the nodes that support the following activities must be determined as part of the definition of the specified deployment model:

- The execution of these components
- The storage of data

## Step 4: Obtaining a product mapping

Given the existing base of servers, the software architect selects a product mapping based on Windows 2000/xSeries servers, which immediately gives us an overview of the recommended products and their relationships (Figure 15).



Outside world
Demilitarized zone (DMZ)
Internal network
Windows 2000 + SP2; IBM SecureWay Directory V4.1.1; IBM HTTP Server 1.3.26; IBM SDKit 4.0.3; IBM DB2 UDB ESE V8.1
Windows 2000 + SP2; IBM WebSphere Application Server V5.0 HTTP Plug-in; IBM HTTP Server 1.3.26
Windows 2000 + SP2; DB2 UDB ESE 8.1; IBM WebSphere MQ 5.3; IBM WebSphere MQ Integrator 2.1
Public Key Infrastructure
Domain Name System
User
INTERNET
Protocol Firewall
Web Server Redirector
Domain Firewall
Directory and Security Services
LDAP
Application Server
Integration Server
Existing Applications and Data
Windows 2000 + SP2; IBM WebSphere Application Server V5.0; IBM DB2 UDB ESE 8.1; IBM WebSphere MQ 5.3
Windows 2000 + SP2; IBM WebSphere Application Server V5.0; IBM WebSphere MQ V5.3

*Figure 15 Self-Service::Router Product mapping*

**Note:** The source for all product mapping information can be found on the Patterns for e-business Web site:

http://www.ibm.com/developerworks/patterns/u2b/at5-product-nt-was4-mb.html

## Follow-on steps

The software architect continues through the steps of the architectural analysis task. This includes creating an architecture overview diagram (Figure 16).

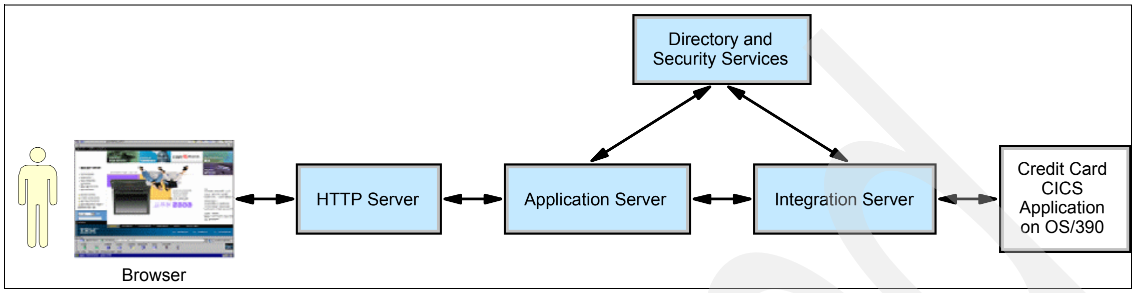

Directory and Security Services
Browser
HTTP Server
Application Server
Integration Server
Credit Card CICS Application on OS/390

*Figure 16 First Financial architecture overview diagram*

While completing the remaining steps of the architectural analysis task, the software architect documents the results in an early draft of a software architecture document:

- Survey available assets: Identify other relevant assets that might be applicable to this project.
- Define the high-level organization of subsystems: Influenced by the selected patterns, you typically define subsystems that do not span processing nodes.
- Identify key abstractions: These are specific to the functionality to be delivered and are not provided by the patterns.
- Identify stereotypical interactions: This is indirectly influenced by the patterns insofar as subsystems have been influenced.
- Develop deployment overview: This is similar to the product mapping diagram (Figure 15 on page 20). The difference is that hardware and software specific to First Financial appear in the diagram, such as their existing systems. Standard UML notation can be used rather than the free-form notation that the patterns use.
- Identify analysis mechanisms: The patterns have already identified implementation mechanisms that suit this application, such as messaging. The architect should determine what other mechanisms this application requires, such as security or transactions. It is likely that many of the required mechanisms are already supported by the selected product mapping.

The architect then performs additional modeling and prototyping as necessary to prove the feasibility of the project. This is described in the construct architectural proof-of-concept task.

In the case of First Financial, the architectural analysis done so far and the good fit of the patterns to the problem at hand provide a sufficient proof-of-concept, convincing the stakeholders that the project can proceed to the elaboration phase.

## Project management

The analysis and design work was performed to confirm technical feasibility. It also serves as input to the business case and project plans, specifically:

- Estimating the software budget, taking into account, among other elements, the costs of suggested products
- Specifying the staffing resources that are required to deliver the solution (among others derived from selected products)

**Note:** Other factors are crucial in being able to budget realistically for the proposed solution, such as sizing estimates, available skills, and other relevant non-functional requirements (availability, performance, and so on), but these are outside the scope of this discussion.

## Summary

This section described how Patterns for e-business can be used effectively to help reach the goals of the inception phase. Patterns for e-business allow us to arrive very quickly at a proof-of-concept. However, Patterns for e-business are not complete architectures by themselves. They provide a basis for the architecture, which must be elaborated to include the specific requirements and software components that are required for a specific system.

The use of Patterns for e-business assets typically increases the overall technical quality of an architectural proof-of-concept because the patterns are based on proven software and hardware product combinations.

# Using Patterns for e-business during elaboration

A key goal of the elaboration phase is to baseline the architecture of the system. In the inception phase, architectural analysis was optional. The architectural proof-of-concept that was produced in the inception phase could be anything from a documented analysis to a set of throw-away prototypes. In the elaboration phase, the goal is to baseline the architecture, including a working subset of the system (not throwaway) that demonstrates that architecture.

Table 2 is an overview of how Patterns for e-business apply to RUP during the elaboration phase. We focused on those parts of RUP to which the assets of Patterns for e-business are of most value by accelerating the development of the associated artifacts. This is only a small subset of a complete RUP-based software development process.

*Table 2 Elaboration and Patterns for e-business*

| **Elaboration disciplines** | **Added value of Patterns for e-business** |
|---|---|
| Discipline: Requirements<br><br>Activity: Refine the system definition<br>Artifacts: Vision (updated), use cases, supplementary specifications | By the end of the elaboration phase, the vision and requirements should be stable. There must be enough detail to define a stable architecture and mitigate all major technical risks.<br><br>Patterns for e-business, which might have been used for proof-of-concept work in inception, are revisited in the elaboration phase. As requirements are better understood, the applicability of the particular business patterns can be confirmed or adjusted. |
| Discipline: Analysis and design<br><br>Activity: Define a candidate architecture<br>Activity: Refine the architecture<br>Artifact: Software architecture document | As the architecturally significant requirements are detailed, Patterns for e-business are revisited to confirm that any choices made during proof-of-concept work are still applicable. Fit-gap analysis determines how much additional work is needed to flesh out a complete architecture. |

## Requirements

Additional requirement details that are obtained in the elaboration phase can result in the discovery of additional business patterns, or in confirming or rejecting the applicability of other business patterns. More typically, however, the architect can make better decisions for selecting applicable solution patterns because of the greater availability of more detailed requirements.

In the case of First Financial, we reviewed their requirements in more detail and updated the artifacts (vision, use cases, and supplementary specifications) that were created during the inception phase. In addition, we prioritized requirements in terms of risk and architectural significance, so that we could implement the highest-priority, architecturally significant requirements first. The self-service business pattern remained the only business pattern that was applicable to this solution. The following additional non-functional requirements were obtained and documented in the supplementary specifications:

- The application must support multiple languages.
- Performance must be enhanced over existing Web-based applications.

- Capacity/logon must handle 500 logons and fifty transactions every minute.
- Scalability must support 10,000 users in two years.
- Security must include secure transaction support, strong encryption, and only authenticated connections to CICS.
- Technology constraints must be taken into account, such as front-end access, which is Netscape and Microsoft Internet Explorer browser-based, CICS Version 4.1.

## Analysis and design

A key goal of the elaboration phase is to create a stable software architecture, documented by the software architecture document artifact. The starting point is any initial work that was done in the inception phase, but as more information is available, the assumptions made in the inception phase are backed up, modified, or elaborated.

The define a candidate architecture activity includes the same task, architectural analysis, that was described for the inception phase. However, in the elaboration phase, this task focuses on defining the architecture that is to be the basis for designing and implementing the system. It is no longer an optional task.

In addition, the refine the architecture activity refers to tasks that elaborate all aspects of the architecture. These tasks, along with a description of some of the impacts of Patterns for e-business, are:

- Identify design mechanisms.

  This involves categorizing analysis mechanisms, making an inventory of available implementation mechanisms, and making build and buy decisions. In this task, the architect considers the costs of acquisition and considers alternatives. The product mappings provided by Patterns for e-business are a good starting point, but this task confirms that the selected products are the right solution. These might have been identified by earlier analysis, but it is not likely that many of the required mechanisms are already supported by the selected product mapping. If any gaps are detected at the level of the application patterns, it becomes very likely that both the application and runtime patterns must be enhanced to cater to the additional functionality to be delivered by the application.
- Identify design elements.

  The patterns only provide part of the solution. The architect (in collaboration with designers) must identify the subsystems and their interfaces for the new software to be developed. However, the selected products also influence the design options. For example, if the underlying product set is IBM

WebSphere® Application Server, the design is likely to be composed of J2EE™ components, such as servlets and EJBs.

- Describe the runtime architecture and describe distribution.

  These tasks describe how the functionality of the system is composed of concurrent inter-communicating elements distributed across physical nodes. At this point, an initial deployment model already exists that reflects the selected runtime pattern and product mapping. It describes both the major concurrent elements and the mechanisms for communication. However, the architect now does a much deeper analysis, evaluating memory, disk and computing capacity, communication bandwidth, response and throughput requirements, and so forth, to decide on specific hardware and software configurations. Also, any new design elements identified must be allocated to processing nodes—nodes that might or might not already have been identified by the runtime pattern.

- Incorporate existing design elements.

  Existing design elements include any existing systems with which you want to integrate, commercial software products to be integrated into the system, and any other software that you want to incorporate and adapt to the needs of this specific project. This task describes how to incorporate these elements into the design of the system.

  **Note:** In some situations, it might be useful to document the way the products and technology are to be used with use case realizations. The Patterns for e-business series of Redbooks™ can provide direct input toward creating such models with their detailed descriptions of the interactions between the various components that characterize the adopted products and technology.

- Structure the implementation model.

  This task establishes the structure of the implementation. The implementation model is influenced by the design model, which in turn, is influenced by the selected patterns.

  **Note:** The Patterns for e-business series of Redbooks offer additional guidance for performing product and technology selections. After making the high-level decisions, you can consult the relevant Redbooks for a discussion about the more detailed trade-offs that are involved in deciding how best to use the selected products, technology, or both.

In the case of First Financial, the pattern selections and product mappings that were made in the inception phase continued to hold true, despite the identification of new and more detailed requirements in this phase and a detailed analysis of procurement costs and required capacity.

It should be noted that this is not always the case. Many more factors can enter the process during the elaboration phase, such as:

- The client decides to standardize a particular product or series of products.
- New requirements cause another product to be preferable (for example, a need to support audit trailing facilities that are compliant with specific legal requirements).
- An evaluation of the performance shows that the initial hardware and software configuration is not sufficient.
- There are alternative products with which the staff is familiar.

## Related work in elaboration

In addition to defining and refining the software architecture document, there is related work in the elaboration phase:

- Design and implementation proceeds on the highest-priority use cases.
- Hardware and software that was defined by the product mapping is acquired, installed, and configured.
- An executable subset of the system is built and tested that demonstrates the selected architecture.
- The architecture is assessed based on both the software architecture document and experience with the executable system.

As in the inception phase, the architectural decisions, including hardware and software product decisions that are accelerated by Patterns for e-business, affect all the other disciplines. In addition to affecting project management (in the same way as described in the inception phase), these decisions affect test planning (test discipline), deployment planning (deployment discipline), and the development environment (environment discipline). See RUP for more details.

One responsibility of the environment discipline is to put hardware and software that support the development activities in place. The Patterns for e-business Redbooks offer information that you can use to build this development environment. For each of the involved products, identified through the physical-level technical component model (which takes part of its input from the product mappings), the system administrators and tool specialists should refer to the product installation instruction chapters in the related Patterns for e-business Redbooks.

## Using Patterns for e-business during construction

During the construction phase, you develop the design and implementation in line with the work that was carried out during the elaboration phase. Patterns for e-business further support the activities undertaken at this stage, insofar as new requirements and changes to the architecture are still possible. Typically, such changes should be minor, and the need for reevaluation of these patterns will be minimal.

The emphasis during the elaboration phase was on determining the key requirements influencing the overall architecture, but during construction, you concentrate on creating more detailed implementations, interactively incorporating additional requirements so that you can:

- Make better estimates for the follow-on iterations.
- Detail the overall specification for the structure and topology of the system.

## Using Patterns for e-business during transition

The selection or creation of installation and system management tools typically comes earlier than the transition phase so that these capabilities can be used internally. However, they must be finalized before release to the customer, which occurs in the transition phase.

Several of the Patterns for e-business Redbooks have a chapter on system management guidelines that provide information for:

- Selecting system management tools
- Selecting system management processes
- Selecting and defining job descriptions, based on identified roles, to support the IT system

The Patterns for e-business series of Redbooks can also further support building the production environment by referring to the product installation instruction chapters.

## Summary

This paper shows how Patterns for e-business can be used effectively to help achieve important goals for each phase in a RUP project. With Patterns for e-business, you can arrive very quickly at a candidate architecture through an asset selection process and validate those selections against the requirements. As the project proceeds, guidance, typically in the form of Redbooks for specific patterns, is available for the further refinement in the use of the pattern, such as installation and system management.

The use of the Patterns for e-business assets should increase the overall technical quality of the system given its reliance on proven software and hardware product combinations.

In this Redpaper, we have shown how to start from basic business requirements and define a solution on the basis of a single business pattern. We illustrated how this can be applied initially in the inception phase to create an architectural proof-of-concept, and then refined iteratively through elaboration and later phases. We demonstrated how Patterns for e-business can contribute significant value by accelerating the creation of some of the key RUP artifacts. You can easily integrate the asset selection process into the activities of RUP, and you can use the results directly during the creation of the various RUP models. The assets of Patterns for e-business can deliver a significant part of the proposed solution and lend themselves very well to integration into an overall solution. They also make an important contribution to the overall quality of the solution.

## For more information about RUP

The following publications can help you learn and master RUP quickly:

- Kruchten, P., *The Rational Unified Process: An Introduction, Second Edition*, Addison-Wesley, 2000, ISBN 0201707101
- Kroll, P. and P. Kruchten, *The Rational Unified Process Made Easy: A Practitioners Guide to the RUP*, Addison-Wesley, 2003, ISBN 0321166094
- Jacobson, I., et al., *The Unified Software Development Process*, Addison-Wesley, 1999, ISBN 0201571692
- Royce, W., *Software Project Management: A Unified Framework*, Addison-Wesley, 1998, ISBN 0201309580
- Many articles in *The Rational Edge* online e-zine, available at:

  `http://www.therationaledge.com`

RUP has a variety of getting started resources, including white papers and road maps. You can download a trial version of Rational Method Composer that includes RUP from:

`http://www.ibm.com/software/awdtools/rmc/`

For information about additional resources, training, and consulting services, visit:

`http://www.ibm.com/software/rational/services`

## The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Michele Galic** is an IT Specialist at the International Technical Support Organization, Raleigh Center. Her focus is on the IBM WebSphere family of products and Patterns for e-business. She has 13 years of experience in the IT field. She holds a degree in Information Systems. Before joining the ITSO, Michele was a Senior IT Specialist in IBM Global Services in the Northeast, specializing in the WebSphere field.

**Bruce Macisaac** is the Technical Lead for RUP Development in Cupertino, California. He has 20 years of software development and process development experience. He holds a degree in Computer Sciences and Mathematics. Before joining IBM, Bruce served in a variety of roles, including Software Architect on the Canadian Automated Air Traffic System, and Project Lead for smaller projects at MacDonald Dettwiler in Vancouver, British Columbia.

**Dan Popescue** is a member of the RUP Development team in Vancouver, BC. He has 19 years of software development experience on two continents and holds a Master of Science degree in Electrical Engineering. Before joining IBM Rational, Dan was a member of the 3G/GPRS team in Motorola, Canada, where he served in a variety of roles, including Software Architect, Team Lead, and Process Engineer.

# Notices

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  `ibm.com/redbooks`
- Send your comments in an email to:
  `redbook@us.ibm.com`
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099, P.O. Box 12195
  Poughkeepsie, NY 2455 South Road U.S.A.

Redpaper

## Trademarks