Business-driven development

Tilak Mitra

Understand the fundamentals of optimizing business processes to assure information technology (IT) systems adequately meet the business needs of the enterprise before implementing a Service-Oriented Architecture (SOA). This article addresses the need for business-driven development (BDD) and highlights key steps to successfully institutionalize it.

Introduction

In today's increasingly tight economy, more and more businesses are finding that information technology (IT) spending is controlled by various lines of business outside of the IT department. In order for IT departments within an enterprise to survive in and adapt to this controlled financial environment, they need to align themselves with demands of the business. Additionally, business processes undergo constant change and the enterprise needs to swiftly adapt its strategies to reflect these changes. The inherent problem with the enterprise software development process is that it suffers from a lack of agility to match the pace at which the business needs to change in order to keep up with the market trends and competition. In order for enterprise IT departments to survive and adapt to a controlled financial environment and to react to the fast-paced change in business processes, IT needs to enhance its capability and maturity to align itself with the business demands. IT departments must move away from creating IT-centric solutions and move toward creating solutions that realize one or more business process. Business-driven development (BDD) is a methodology for developing IT solutions that directly satisfy business requirements and needs.

This article is an effort to create an understanding of the essential tenets of BDD and proposes a mechanism for institutionalizing it in order to achieve repeatable success.

The need for BDD

Companies today need to keep up with the pace at which the competitive market demands new business capabilities. Approximately 80 percent of a company's IT budget is spent either maintaining or enhancing existing applications. These existing applications were not created with flexibility in mind, and hence, while the business is leapfrogging with new and enhanced processes, the IT backbone is incapable of honoring the required changes. Traditional applications and architectures are not able to keep up with business innovation, primarily because the
Business-driven development processes are not adaptable to on demand business needs. Business requirements often get transformed into siloed IT projects that cannot work together; reusability between artifacts created for different IT projects is often very low. Creating applications that are flexible enough to react to the unknown requires a more systematic approach toward application development. With the business not able to create IT functionality that is capable of reacting to the unknown, it has traditionally been very difficult to justify the deeper budgetary requirement to create flexible IT applications. The traditional inflexibility of application architectures makes even small improvements so expensive that they become virtually impossible to justify.

A mechanism needs to be devised by which IT efforts are interlocked with business strategy and requirements through an execution framework that is standardized, well understood, and can be executed repeatedly and successfully. The enterprise might achieve business flexibility through IT by modeling the business processes that collectively define the way the business executes. The first thing to do is model a business process through its constituent process steps. By measuring a business process or a key use case through return on investments (ROIs), key performance indicators (KPI), or other metrics, the enterprise can use these business process models (BPMs) as an essential mechanism to communicate the business needs to the IT realm. The business and IT can significantly bridge the communication chasm by using well-articulated BPMs that create a link between what the business needs and what IT implements and delivers.

While the starting step for BBD is the creation of BPMs, the IT solution structure also needs to adapt to using the BPMs as input artifacts to the design and development phases of the software development life cycle. The IT architecture needs to be able to design and implement the process activities as software components or services.

By using BBD, the enterprise models and provides new business processes (when conceptualized) to the IT department. Analysis of the new process might reveal either that software services might already exist to address the need and the only work effort required is to wire the existing software services to realize the new business process, or it might reveal that the enterprise needs to implement new software services and add them to the IT service portfolio. Similarly, if changes are needed to an existing process, the BPM is revamped to reflect the change and delivered to IT for subsequent technical revision based on which services might need to be enhanced or modified.

A BDD approach helps increase the agility of the business and also helps prioritize and align IT initiatives with business imperatives. It also indirectly helps in simplifying the process of cost justification for IT budgets within an enterprise.

The execution model

As mentioned in the previous section, enterprise IT should strive to bridge the gap between business needs and IT solutions and also be agile and responsive in creating IT solutions. This need has led to the development of a Services-Oriented Architecture (SOA), which provides an IT framework along with a set of principles and guidelines to create IT solutions as a set of reusable, composable, and configurable services that are independent of applications and runtime platforms. Transitioning an enterprise to SOA requires a BDD approach that uses business goals
and requirements to drive downstream design, development, and testing. This promises to create composite business applications by reusing existing or newly created services, which helps to create adaptable and flexible business solutions. It also brings a much needed flexibility in enterprise IT and helps to align IT solutions with business needs.

This section provides a high-level overview of the various steps involved in a typical project that follows the BDD methodology. The section that follows after this one describes the steps in more detail. **Figure 1** depicts the flow of activities that define the high-level steps of BBD.

**Figure 1. The execution model**

![Image of the execution model](image_url)

The first step is to model the business processes that need IT enablement. It is advisable to start by modeling the key business processes and, using the outputs of the modeling activity, to communicate the business requirements to the IT domain. It is the responsibility of the business users and stakeholders to associate the processes and its significant steps with ROI, KPI, and any other pertinent metric. In the later stages of the IT life cycle, this will help to validate that IT delivered what was proposed by the business.

Once the processes are modeled, the outputs of the models can be used as inputs to the requirement gathering phase of an initiative. The activities or process steps that make up a given business process model can be analyzed to form the basis of use case modeling. Developing use cases is a significant step in the requirement gathering phase of a project. Based on the use cases, the application architecture is structured and the enterprise services are identified, designed, developed, and subsequently wired together as service composites that realize the business processes. After development, the project moves to the deployment stage during which the developed components are exposed as publishable, location-transparent, and discoverable services. These software services are deployed to an execution runtime, such as an application server.

In post deployment, the project enters the monitoring or management phase. Once they are up and running, business processes can be monitored for real-time performance and data capture, reporting, and analysis. For this to happen, the steps in the business process (as modeled in the first step) need to be assigned various business metrics (for example, ROI and KPI) against which run-time performance, latency, and other factors can be measured. Measurement is essential to
validate whether the IT solution meets the needs of the business as defined by a service level
greement (SLA).

Data obtained from the run-time monitoring is analyzed against the expected SLA or other
benchmark performance metrics and criteria. The captured information is provided to the
architects, designers, and developers who analyze the data and find out innovative ways
of optimizing or improving the process through enhancements and performance tuning of
implementation code. Sometimes the changes might also be made by the business users by
changing business rules using external interfaces, which requires no code changes. If the analysis
suggests changes to be made in the business process, the corresponding process models
can be modified and the same steps (that is, develop-deploy-monitor) repeated to enhance
the implementation. This completes the execution loop with analysis and process adaptation
techniques feeding back to the modeling step. This mechanism helps both the business and
IT to adapt to the changing business needs with quick turnaround time. It helps justify cost
enhancements to existing system functionality by directly associating business needs and
justification to a given IT initiative.

Executing an IT project with BDD

So far you've seen the various high-level steps that make up BDD. How do you translate the
execution model into organized and planned work activities that can be repeatedly executed in
a typical software development life cycle? This section attempts to detail the steps needed for
actual project execution. **IMPORTANT NOTE:** Keep in mind that I am describing the methodology
for BDD here. Complete details for each phase of the methodology are project- and technology-
dependent, and thus beyond the scope of this article.

**Business requirements analysis**

The first and foremost step during any IT project initiative is to understand the business
requirements. The high-level business requirements reside in the minds of the key business
stakeholders and inside existing legacy systems. Unless they are documented and signed off, the
project itself might have quite a few unknowns at the very onset. Getting the time of the business
executives is often a challenging endeavor, but it is a necessity and needs to be well planned and
executed. The following (at a minimum) need to be documented:

1. Business vision
2. Business goals (long- and short-term) that realize the enterprise vision
3. High-level business requirements that help attain the goals
4. Problems with existing business processes (such as customer pain points, high costs,
schedule issues, and so on)

It is also essential to have an understanding of the business environment and the enterprise
organizational structure. The business domain model also needs to be analyzed and well
understood. Formalization of this understanding might be documented as a **business domain
matrix**. It is also very important to have an understanding of the high-level business functions
that a given business domain is expected to provide. Having a business domain matrix with
its associated high-level business functions is a good way of concluding the business analysis activities.

**Business process modeling**

BPM is the technique used to visually model a business process through a sequence of activities, use cases, and decision points. The purpose of BPM is to create fully executable models that an engineering group can implement as technical services. BPM involves the activities of modeling the *as-is* and *to-be* business processes and allocating resources to implement each process. Process optimizations are performed through simulations that help in attaining the *ideal* business process state. The to-be state is finalized as a first feasible step -- both from budget and timeline -- toward the ideal process state.

Each organization or business domain has business functions that it is expected to support or provide. The terms *business function* and *business use case* can be used interchangeably. BPM techniques are used to model the aspects of behavior, organizational structure, and business domain objects. Each task is assigned to a role. A role is an entity (a person, computer, or any other type of actor) or group of entities that have the same rights and obligations with respect to performing a task or a group of tasks. A role might be assigned to any number of tasks and an entity might act in any number of roles.

Each business process, when analyzed, can be represented as a sequence of activities or tasks. A task is the smallest unit of work that makes sense to a user. Sometimes a part of the process is involved or complicated enough that it needs to be represented as a use case. An example of a BPM can be seen in **Figure 2**.

**Figure 2. Sample process model**

![Sample process model](image)

The constituent parts of a process (activities or use cases) can be assigned an organizational domain. Sometimes it is practical to assign functional domains or business domains to process activities. This provides an understanding of which business/functional domains are responsible for which steps within a business process. This view provides insight into the architecture and design activities that follow.

The activity steps need to be assigned inputs and outputs. The inputs and outputs should be identified in terms of the business items (that is, the business domain entities such as person, customer, and so on).

A business process needs to be measured against performance criteria. The process itself, along with the activities that comprise the process, can be assigned parameters that can be used to measure the run-time performance. These parameters might be added and simulations might
be run to collect data on how the process needs to perform. Simulation results also might help in tweaking the process model to make simulation runs perform closer to the expected performance measurements.

**What can the enterprise gain from this work effort?**

The process models, when formalized and validated, serve as the main input for the next phase of a project, for example, use case modeling. The data architecture team can consolidate a single view that includes all of the business domain entities. With subject matter experts helping to identify the relationships between the various business entities, this business domain view can quickly evolve into the conceptual data model on which the system is to be built.

**Use case modeling**

The use case model is a view of all the various use cases and their interdependencies, as well as a list of use case actors. A business process role is mapped to a use case actor. A use case can either specify a complete system or any model element that describes behavior, for example, a subsystem. Some modeling even goes to the extent of mapping a use case as a business process. Hence, there surely is flexibility when it comes to identifying use cases from process models.

**So how do you identify use cases from business processes?**

There are no hard and fast rules for identifying use cases from process models, and hence this step has no formalized standard. I will make an attempt to rationalize a mechanism to define the use case model with BPMs as inputs.

If one maps a single activity or task in a process to a use case, each use case would only model a single interaction, since a task is assumed to be equivalent to a single interaction. However, the Unified Modeling Language (UML) definition of a use case describes it as a complete sequence. This implies that the use case specifies all the interactions that have to be carried out to bring the system to a state in which the same use case can be performed again. Hence, limiting a use case to a single interaction would be too constraining.

When mapping an entire process to a use case, the resulting use case might turn out to be too complex. Not only will there be multiple roles, but the number of alternative paths will be very high. Hence, this approach would not be practical.

An alternative might be to define a concept of a step. A step is a sequence of tasks that can be performed without interruption by the same role. As an example, save reservation might be a use case whereas save reservation and send vacation fliers might not be a use case, because there is an elapsed time between the time a reservation is saved to the time vacation fliers might be created, personalized, and sent to the customer. Hence, a different view of a business process is that of a sequence of steps with each step being a use case with a desirable output and behavior.

Once use cases are being identified, the set of activities in a use case can be documented as major steps in a use case. This allows a use case to be documented as a sequence of steps. Transition between process steps within a business process is mapped as dependencies.
between use cases. Use case identification and documentation into a use case model (with actor association) is the first iteration of this overall activity.

With a level of analysis performed at this point, a formal use case model can be created and documented.

**So what is the advantage of this approach?**

Since the use cases might be identified as steps within a business process, every single use case has a purpose for existence. Each use case realizes (partially or fully) at least one business process. No use case is left unidentified and at the end of this work effort and every single step in the business process portfolio has an assigned use case name and major steps of execution. Also, no use case is orphaned, which in other terms means that every use case is directly traceable to at least one business process.

With this work effort being executed successfully, the level of confidence in the success of the IT project is usually increased.

**Service modeling -- SOA**

An enterprise that is moving toward a SOA needs to carefully consider creating an enterprise service portfolio. A service portfolio is a list of various services that are offered by an enterprise, either for external consumption or for internal use. Any IT initiative that falls under the enterprise SOA umbrella needs to create a list of services. Enterprises are at various levels of maturity in their goal to embrace SOA. Most companies are heading towards Web services implementation projects. To become a true SOA-based enterprise, there are much higher levels of maturity to attain than just a Web services implementation. An analysis of the maturity level of an enterprise’s SOA can be a formal process in itself.

It is evident from the above that a methodology needs to be understood, adopted, and followed if an enterprise is to evolve to a true SOA-based enterprise capable of exposing business services. Although this topic is beyond the scope of this article, I shall make an effort to link the outputs of the previous work efforts (that is, business analysis, BPM, and use cases) to the SOA work efforts. I shall also try to illustrate a set of activities (at a broad level) that need to be carried out to complete a SOA initiative.

**Leverage inputs from previous work activities**

The business analysis work effort provides a clear understanding of the business vision and goals of the company. The business processes that need to be automated through IT must be categorized according to some priority criteria. IT might work closely with the business to evaluate key issues facing the business, such as the following:

1. Primary process weaknesses and bottlenecks
2. Customer pain points
3. Processes that need to scale with increasing transaction volumes and so forth

Based on such an evaluation, a plan might be chalked out that identifies the processes to be implemented first. Processes are owned by business domains. Hence, a few initial business
domains might be identified that need primary IT focus to obtain immediate short-term benefits. Any service that might be identified needs to be adequately supported by the business goal it is trying to solve (either fully or partially). This helps in chalking out a strategy for prioritizing business domains for IT enablement. In many cases, the high-priority domains will be ones for which business services need to be identified and defined. Problems with existing business processes might also be used as primary targets for service enablement. In any case, prioritization of IT work efforts needs to be kept in line with the business needs, especially time-to-market concerns.

BPM, as discussed previously, models both the "as-is" and "to-be" business processes. The process model identifies a sequence of tasks (either atomic in nature or represented by use cases) that realize an end-to-end business process. The to-be process models form the basis of the top-down process decomposition technique. In this technique, every business process as well as the use cases that are identified is a candidate to become a service. In addition, every single use case that is identified through the use case modeling work effort might be added to the list of candidate services. (Note: It is very important to understand that these are only "candidates" to be services and might not necessarily make it to the final service portfolio.)

The essence of SOA is to be able to come up with a service portfolio that might be reused to realize multiple higher-level business processes. The implementation of the service also needs to be well thought out so that the implementation artifacts (for example, components) are also inherently reusable across multiple service implementations.

**Figure 3. SOA layers**

![Figure 3](image-url)

Figure 3 gives a high-level view of the various SOA layers. The figure illustrates how steps within a business process can be realized through either a single service or through a composition of multiple services. It also shows how a service might be realized or implemented by components. These components might be assets harvested from existing assets (such as legacy systems), components from packaged or commercial off-the-shelf (COTS) applications, and components developed completely from scratch.
Starting with a candidate service portfolio is an essential step toward the ultimate goal of building an enterprise service portfolio. There are many other steps that need to be exercised to arrive at a service model for an enterprise. One analysis that plays a vital role in strengthening reusability is the analysis of legacy and existing systems to find out possible services that might make their way into the candidate service list. The main point to make here is that the service modeling work activity can use the work artifacts of the previous steps as input and a starting point.

The essential work activities that might be performed for outlining and building on top of the service model are:

1. Top-down analysis (through process modeling)
2. Analysis of legacy systems and applications
3. Creation of an initial service model from the candidate service list using criteria to expose services
4. Clear articulation of service descriptions, capabilities, and quality of service (QoS) -- once the services are selected to be exposed
5. Mapping of every service to a business goal and making sure that a service participates in at least one business process

A SOA system can be broken down into multiple layers and its work activities identified, understood, and executed at each layer to help realize the benefits of becoming a true Services-Oriented Enterprise. **Figure 4** identifies the various SOA steps and how each of the steps is geared in the direction of creating artifacts at various layers of a SOA.

**Figure 4. SOA steps**

The following list provides a brief description of each of the steps as it appears in the figure above:

1. **Business componentization analysis.** This is the work that is essentially done in the business analysis phase of the project execution. The business domain map is created and analyzed at this level.
2. **Service identification.** This is the step during which candidate services are identified by performing both top-down and bottom-up (existing systems) analysis and evaluating the services against a business goal-based criteria matrix.
3. **Service specification.** This work defines the dependencies, composition, exposure decisions, messages, QoS constraints, and decisions regarding the management of state
within a service. It also describes the relationships and the contractual dependencies between services in the model.

4. **Component identification.** This is the step in which enterprise components are identified that will be used to implement the services in the service model. Examples of an enterprise component are an EJB, a component facade, and so on.

5. **Component specification.** This describes the detailed attributes, internal flow, and structure of the component.

There are many other steps involved in coming up with the final service model and portfolio. This section just illustrates how to initiate this work effort by leveraging the work activities of the previous steps.

**System design and development**

During the development phase, the team takes the initial use case and service model and performs a deep dive: The functional specifications of each use case is built by elaborating on the major steps that compose a use case.

**System design**

Use functional specifications as inputs in creating the system's component model. The component model describes the components through the interfaces that they provide, as well as how components depend on each other in the overall system solution. The components are implementation mechanisms that support the exposed services (in the service model). There might also be components that do not directly implement a service; instead, they facilitate implementation of some common utility services (for example, logging, event subscription and broadcasting, and so on). Components that do not expose interfaces to be directly consumed externally are used to facilitate a standardized inter-component communication. Architecturally significant use cases are often used to create sequence diagrams through component interfaces. This proves that the system is designed well to work through the interfaces exposed by the various components in the component model.

One of the activities during this phase is to identify and document the nonfunctional system requirements. These are not only used to create SLAs for the services in the service model (in the previous step), but also are used as inputs into the operational system model. The operational model describes, among other things, the various infrastructure components (such as middleware, messaging, directory services, and so on). The model also describes how to distribute the components across the network and how to deploy them on hardware infrastructure devices.

Before the processes modeled in previous steps can be implemented, the application architects and designers need to define the service implementation, the service description, and service invocation mechanism. Not all process steps can be realized by directly invoking a service. One of the reasons for this can be attributed to strict performance requirements for the step (defined in the SLA); the overhead required for a service invocation might be too costly. These kinds of steps can be implemented by direct invocation of components that do not have a service facade. There are other factors that can result in a hybrid solution for the implementation of the individual process steps. These need to be identified and captured for subsequent reference during development.
This article is by no means an effort to cover system design in detail but gives a high-level description to depict the sequence of steps.

Once the macro design (component and operational models and other design artifacts) is complete, the project execution shifts focus to micro design. Keep in mind that the project activities need to be iterative; the traditional waterfall approach to system development is prone to much higher risk, especially for large and complex projects. In micro design, the class diagrams and sequence diagrams are designed and modeled for each of the components in the component model. This is where proper usage of proven design patterns help in making the design more robust, flexible, and extensible.

**System development**

This is part of the project where you actually implement the design by choosing a programming model (such as Service Component Architecture (SCA) or Java™ 2 Platform, Enterprise Edition (J2EE)), a compatible programming language (for example, Java), and an integrated development environment (IDE) that facilitates comprehensive development. Export the process model artifacts in Business Process Execution Language (BPEL) and use them as a starting point in the development IDE to implement the process activities in the programming language of choice. Also implement the service definitions using a technology of choice. The developed services are wired together through a process choreography engine, which is instrumental in providing the wiring mechanism of the business processes through the service interfaces.

It is very important to understand that a single service might be reused in realizing more than one business process. Each service might realize multiple processes, depending on how it's wired together with other services. In fact, the business gains of BDD are directly realized through service reusability between multiple business processes. Thus, business and IT together attain the agility to adapt to new and changing needs by leveraging IT services through service composition, re-composition, and reusability techniques, thereby adapting to changing business needs in an agile manner.

Infrastructure setup takes place in parallel with development activities. The infrastructure team leverages the output of the operational modeling work efforts to set up the network nodes, hardware, software (with licensing), and other necessary steps to be ready for deployment.

**Deployment**

Once development activities are at a stable state (when each iteration in the build-test continuum has been completed), it's time to deploy the implementation. Synchronize project timelines so that the infrastructure is ready for deployment of the application in the production environment (the one seen by its end users). Also, make sure to carefully plan the distributed deployment of software artifacts. For example, the part of the architectural layer that is going to be used heavily must be fault-tolerant and capable of high loads and volumes, which will likely necessitate a clustered environment for deployment.
Run-time monitoring and analysis

The run-time engine for business processes might be used to analyze the running processes. Unless business processes are monitored at runtime, there is no scientific way to validate whether the business ROI is achieved through the IT solution. The monitoring provides real-time data and analytics. The run-time data collection is possible, since the modeling exercise could associate business and performance metrics to a business process, use case, or activities within processes.

Analyze monitored data and adapt

During BPM, use simulation techniques to find out how the business process can meet the required SLAs. The run-time data collected from the monitoring activities needs to be measured against the simulation results. If the results match closely enough with a reasonable degree of tolerance, then the implementation and entire project might be successful. However, if the simulation results are not matched by the run-time performance, some work needs to be done. The run-time data needs to be analyzed to identify which specific process steps contribute the most to the gap in meeting the SLA.

The first step is to do a careful analysis of the implementation code and identify areas where it might be tuned to achieve better performance. If this exercise is completed and the changes still do not meet the required SLA, then the business process needs to be carefully looked into and changed. The business impact of the changes needs to be analyzed and if the changes are considered to be vital to the business, then the process steps need to change in an effort to close the gap. The infrastructure architecture might also be analyzed to ensure whether changes at that level can satisfy the SLA requirements. It is possible to add redundant infrastructure components to make up for a performance gap, but that runs the risk of exceeding the budget and should probably be the last step to consider.

The important thing to note is that BDD provides a mechanism to adapt and improve business processes. Subsequently, the IT solution that implements the business process can be changed seamlessly, if the process is followed properly.

The roles required to execute

As I said briefly before, the BDD project requires contributions from various people, each with their own unique skillsets. This section introduces various roles required to successfully execute an IT project driven by business goals, vision, and needs. Figure 5 illustrates the few important roles that are required to execute the various phases of a business-driven IT initiative.
Here is description of each key role:

1. **Business analyst.** A high-level role responsible for the business analysis and BPM work activities. The business analyst performs use case identification and creates functional specifications for each use case. This high-level role implies that the analyst might also take on more specialized tasks during the project.

2. **Architect.** A high-level role responsible for the overall work effort of creating the architecture and design of the system. More specialized roles, such as enterprise architect, application architect, SOA architect, lead designer (for micro-design level tasks such as class modeling, sequence diagrams, and so forth), and infrastructure architect, are actually responsible for various architectural work efforts that constitute the design phase of the project.

3. **Developer.** A high-level role responsible for the implementation of the solution. Again, more specialized actors within this role might actually carry out the work, such as a database programmer, a Java developer, a Web developer, and a business process choreographer to name a few. Developers work on specific layers of the application stack and each requires specialized skills for that layer.

4. **Tester.** A role responsible for performing activities required to test the application before it is deployed into a production environment. The tester creates test scripts directly from the functional requirements that illustrate the use case. These test scripts are then executed with various input conditions to validate the desired output conditions. The more thorough the test cases and their execution, the more robust is the application, minimizing the chances of surprises during runtime.

5. **Deployment manager.** Responsible for deploying the application code on the infrastructure in various environments (for example, testing, staging, and production) and all the work efforts required for seamless deployment applicable to all environments (such as developing installation scripts, configuration management, and so on).

6. **IT operations manager.** Responsible for supporting the activities essential for ongoing support of the application when it is up and running, especially in the production environment. This role might also be responsible for collecting the run-time data from the application and analyzing the results against SLA requirements.

This is not meant to imply that the roles mentioned here are the only roles that might be required in a project. These are the key roles that will probably be involved in any typical project, but customize the list based on the client’s organizational structure and on the project’s requirements and complexities.
Tooling support

Technologies, IT methodologies, and processes can only be as good as the maturity of the tooling that supports them. A prime example is the J2EE technology and the related programming model. It is only through the advancement of sophisticated tooling -- much of it provided by IBM and other significant platform and tool vendors -- that the J2EE design and development process has been immensely simplified. Without tooling support for various work efforts in a given project life cycle, processes and methodologies are easier said than done.

IBM has a gamut of tools that maintain its vision of providing end-to-end integrated tooling that supports all aspects of BDD. The following is a description of many of those tools and how they fit into the various phases of BDD:

1. **Rational® Software Modeler** and **WebSphere® Business Integration Modeler and Monitor**. These tools aid in the development of BPMs. The tools use a notation called Business Process Modeler Notation for process modeling. Business analysts use them to formalize business processes into reusable artifacts. Once the analyst identifies a solution as the most optimal "to-be" BPM, he or she defines the business requirements and refines them into software requirements in the form of use cases. **IBM Rational RequisitePro** might be the tool of choice to formally document and maintain these use cases. Well-written use cases act as a solid foundation for architecture, analysis, and design activities. The requirements in IBM Rational RequisitePro can link to process models from other tools, thereby providing full traceability and ensuring that downstream architecture, design, and development activities are driven from business requirements (in the form of process models).

2. **Rational Software Architect**. System architects use this tool for system architecture and design. The output from Rational Software Architect and WebSphere Business Integration Modeler and Monitor import into Rational Software Architect as UML models and hence form the basis of system architecture and design. The transition from the analyst's work in Business Process Modeler Notation or BPEL to the inputs for the architect in UML is seamless, which work together to save time and trouble.

3. **Rational Application Developer** and **Rational Web Developer**. J2EE and Java developers use these tools to convert Rational Software Architect design models into implementation code. This development environment automates many of the tasks performed in the construction and consumption of Web services, and thus facilitates in the development of service-oriented artifacts. The development environments help the developer by providing drag-and-drop widgets for technology components, such as Java Server Pages (JSPs), servlets, Enterprise JavaBeans (EJBs), and so on, and thus can help reduce development time. The packaging of programming modules is now a standard feature for most vendor tools.

4. **IBM WebSphere Integration Developer**. Integration developers use this tool to assemble composite applications that are deployed on a run-time server environment like the **IBM WebSphere Process Server**. WebSphere Integration Developer assists in the choreography of services by wiring them together into composite applications that might be tested and subsequently deployed to a run-time environment.

5. **Rational Functional and Manual Tester** and **Rational Performance Tester**. Application and system testers use these products. Try integrating these tools with Rational RequisitePro.
to create automated test scripts directly from use cases and functional specifications. This allows traceability between the test cases and use cases and helps assure greater levels of success during the entire project life cycle. The tool also supports data gathering of test runs and creates reports from that data -- this data provides feedback to the development team.

6. IBM Tivoli® Monitoring. This tool monitors the execution runtime of the system. It gathers run-time data and analytics that might be subsequently used to analyze and adapt the modeled "to-be" business processes to meet the SLA requirements. IBM Tivoli Configuration Manager configures software applications as well as the deployment environment.

7. IBM Rational Portfolio Manager. Executives and project managers primarily use this tool to gain insight into business benefits, costs, and risks associated with the SOA services portfolio. They also might use this tool to prioritize the creation of proposed services, to track the status of development of services, to forecast the need for future services, and to help manage SOA project team dependencies. The tool also provides sophisticated features for the SOA development life cycle, such as creation, operation, maintenance, enhancement, and retiring of services.

**Conclusion**

The concepts of BDD have been around for quite some time. But it is only recently that the approach has attracted attention and become widely acceptable. The availability of standards for all the activities that make up this methodology is the primary reason for why this approach is realizable. There is a common language in BPEL that companies can use to define the business requirements. UML provides for the common IT language to describe architecture and design artifacts, while many of the development toolsets are built on top of a standard framework. (In the case of IBM and many other vendors, that framework is Eclipse.)

Following the methodology closely helps deliver the value proposition of BBD in terms of reduction in cost of development, decreasing time to market, and improving quality and customer satisfaction. The process addresses the problem of cost reduction not only through reusability and composability of services to realize multiple business processes, but also through legacy modernization techniques. It addresses the problem of reduction of time to market through reuse, as well as by improving the communication mechanism between business and IT through well-articulated business requirements and BPMs. Customer satisfaction can be greatly improved by addressing the customer pain points in the existing business processes. Companies can achieve this by creating IT solutions based upon understanding the strategic impact of creating business-driven solutions, which is the foundation principle of BBD.

This article provided a practical understanding of the essential tenets of BBD and explained how to execute an IT project through various work activities that align the final IT solution with business needs. It also described the high-level roles of key project participants at various project phases. Additionally, it outlined the end-to-end toolset offered by IBM in support of BBD -- further evidence that BBD has arrived in the industry.
Companies now understand the inherent advantages of an approach that takes them toward asset-based, business-centric IT solutions. It is hoped that this article has provided useful insights into BBD, an approach that is rapidly gathering attention and significance.
Related topics

- Learn more about:
  - Rational Software Architect
  - IBM WebSphere Process Server
  - SCA Programming model
- Get your hands on application development tools and middleware products from DB2®, Lotus®, Rational, Tivoli, and WebSphere. You can download evaluation versions of the products at no charge, or select the Linux® or Windows® version of developerWorks' Software Evaluation Kit.