

## A Rational Development Process

Dr. Philippe Kruchten

*This is a usable outline of the phases and activities of the generic and tailorable Rational Software Development Process. The process addresses high-risk areas upfront, allows for requirement refinements, accommodates change, and keeps the software product and customer satisfaction the main focus.*

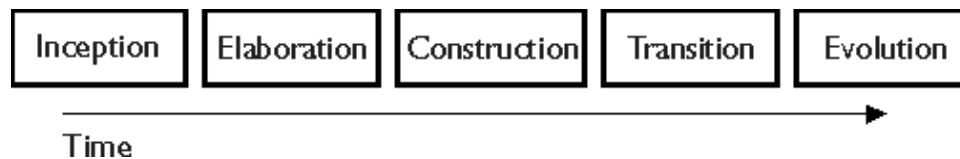
This article provides a high-level description of the philosophy and structure of Rational's software development process. The process is iterative, incremental, object-oriented, managed and controlled. This process provides an organized, flexible approach that is generic enough that it can be applied to a wide variety of software projects and products, regardless of their size or their application domain.

### The Overall Software Lifecycle

#### Two Perspectives

Rational's process may be approached from two different and integrated perspectives:

- A management perspective that deals with the financial, strategic, commercial, and human aspects.
- A technical perspective that deals with quality, engineering, and design method aspects.



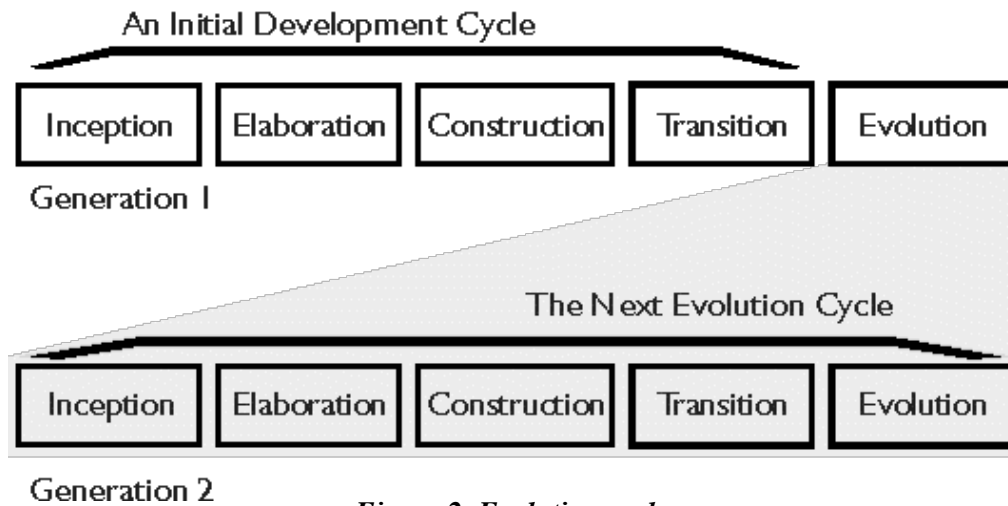
*Figure 1. Management Perspective.*

#### Cycles and Phases

As seen from a management perspective (*Figure 1*), i.e., the business and economics point of view, the software lifecycle is organized along four main phases, indicators of the progress of the project:

- **Inception** - The good idea: specifying the end-product vision and its business case, defining the scope of the project at the beginning of the project.
- **Elaboration** - Planning the necessary activities and required resources; specifying the features and designing the architecture more thoroughly.
- **Construction** - Building the product and evolving the vision, the architecture and the plans until the product--the completed vision--is ready for transfer to its users community.
- **Transition** - Transitioning the product to its user's community, which includes manufacturing, delivering, training, supporting, and maintaining the product until the users are satisfied.

Going through the four phases is called a development cycle, and it produces a software generation. Unless the life of the product stops, an existing product will evolve into its next generation by repeating the same sequence of inception, elaboration, construction, and transition phases, with a different emphasis, however, on the various phases. We call this period evolution (*Figure 2*). As the product eventually goes through several cycles, new generations are being produced.



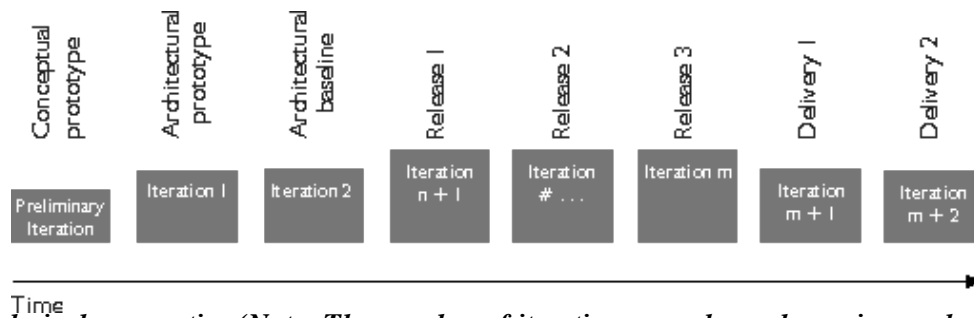
*Figure 2. Evolution cycle.*

For example, evolution cycles may be triggered by user suggested enhancements, changes in the users' context, changes in the underlying technology, reaction to the competition, etc.

In practice, cycles may slightly overlap: the inception and elaboration phase may start during the trailing part of the transition phase of the previous cycle.

**Iterations**

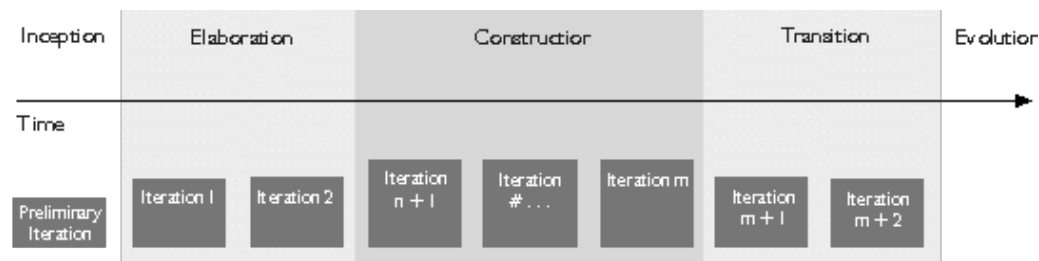
From a technical perspective, the software development is seen as a succession of iterations through which the software under development evolves incrementally (*Figure 3*). Each iteration is concluded by the release of an executable product, which may be a subset of the complete vision, but useful from some engineering or user perspective. Each release is accompanied by supporting artifacts: release description, user's documentation, plans, etc.



*Figure 3. Technical perspective (Note: The number of iterations per phase shown is merely for illustration purposes).*

An iteration consists of the activities of planning, analysis, design, implementation, and testing in various proportions, depending on where the iteration is located in the development cycle.

The management perspective and the technical perspective are reconciled and in particular, the end of the phases are synchronized with the end of iterations (*Figure 4*). In other words, each phase is broken down into one or more iterations.



**Figure 4. Management and Technical Perspectives Combined.**

However, the two perspectives--management and technical--do more than just synchronize on a few well-identified milestones; they both contribute to a common set of products and artifacts that evolve over time. Some artifacts are more under the control of the technical side, some more under control of the management side.

The availability of these artifacts and the satisfaction of the established evaluation criteria for the product and the artifacts are the tangible elements that constitute the milestones, much more than mere dates on a calendar.

Like cycles, iterations may slightly overlap, e.g., the planning or architecture activities of iteration N may be started toward the end of iteration N-1. In some cases, some iterations may proceed in parallel: a team, working on one part of the system, may have no deliverable for a given iteration.

### Discriminants

The emphasis and importance of the various phases, the entry and exit criteria, the artifacts involved along a development cycle, and the number and length of the iterations may vary depending on four major project characteristics that are process discriminants. In order of decreasing impact, the most important process discriminants are

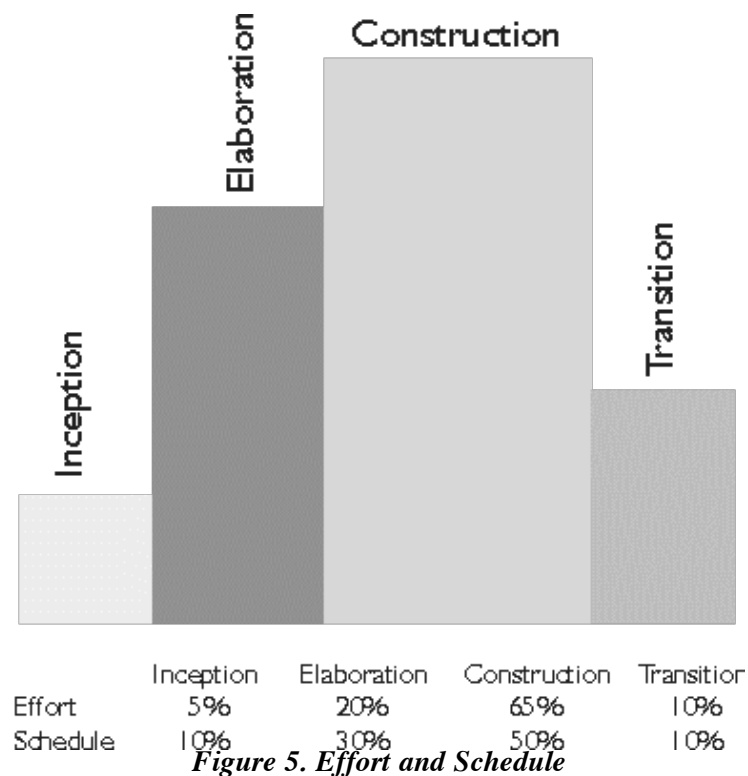
The business context:

- Contract work, where the developer produce the software on a given customer specification and for this customer only.
- Speculative or commercial development, where the developer produce software to be put on the market.
- Internal project, where customer and developer are in the same organization.
- The size of the software development effort-as defined by some metrics, such as Delivered Source Instructions, or in functions points, etc., or number of person-months, or merely in cost.
- The degree of novelty-how "precedented" this software effort may be relative to the development organization, and particularly whether the development is in a second or subsequent cycle. This discriminant includes the maturity of the organization and the process, its assets, its current skill set, and issues such as assembling and training a team, acquiring tools, and other resources.
- The type of application, the target domain-MIS, command and control, embedded real-time, software development environment tools, etc., especially with respect to the specific constraints the domain may impose on the development: safety, performance, internationalization, memory constraint, etc.

This article first describes the generic process, i.e., the part of the process that applies to all kinds of software developments, across a broad and general range of these discriminants. It then describes some specific instances of the process for some values of the discriminants, as examples.

### Effort and Schedule

All phases are not identical in terms of schedule and effort. Although this will vary considerably depending on the project discriminants, a typical initial development cycle for a medium size project should anticipate the ratios in *Figure 5*.



But for an evolution cycle, the inception and elaboration phases can be considerably reduced. Also, using certain tools and techniques, such as applications builders, the construction phase can be much smaller than the inception and elaboration phase together.

### Phases of the Process

#### Inception Phase

This phase takes the original vision of a potential product and transforms it into an actual project. Its purpose is to establish the business case for a new product or a major update and to specify the project scope.

For the development of a new product, the main outcome of this phase is a "go-no go-" decision to move into the next phase and to invest time and money to analyze in detail what is to be built, can it be built, and how to build it.

For the evolution of an existing product, this may be a simple and short phase, based on users' or customers' requests, on problem reports, and on new technological advances.

For a contractual development, the decision to proceed is based on experience of the specific domain and on the competitiveness of the development organization in this domain or market. In this case, the inception phase may be concluded by a decision to bid, or by the bid. The idea may be based on an existing research prototype of which its architecture may or may not be suitable for the final software.

Entry criteria:

The expression of a need, which can take any of the following forms:

- An original vision.
- A legacy system.
- An RFP (request for proposal).
- The previous generation and a list of enhancements.
- Some assets (software, know-how, financial assets).

- A conceptual prototype, or mock-up.

Exit criteria:

An initial business case that contains at least

- A clear formulation of the product vision--the core requirements--in terms of functionality, scope, performance, capacity, technology base.
- Success criteria (for instance revenue projection).
- An initial risk assessment.
- An estimate of the resources required to complete the elaboration phase.

Optionally, at the end of the inception phase, we may have

- An initial domain analysis model (~10 percent-20 percent complete), identifying the top key use cases, and sufficient to drive the architecture effort.
- An initial architectural prototype, which at this stage, may be a throw-away prototype.

### **Elaboration Phase**

This phase is to more thoroughly analyze the problem domain, defining and stabilizing the architecture, and addressing the highest risk elements of the project. So that at the end of the phase, we can produce a comprehensive plan that shows how the next two phases will be done:

- A baseline product vision, i.e., an initial set of requirements, based on an analysis model.
- Evaluation criteria for at least the first construction iteration.
- A baseline software architecture.
- The resources necessary to develop and deploy the product, especially in terms of people and tools.
- A schedule.
- A resolution of the risks sufficient to make a "high fidelity" cost, schedule, and quality estimate of the construction phase.

In this phase, an executable architectural prototype is built, in one or several iterations depending on the scope, size, risk, novelty of the project, which addresses at least the top key use cases identified in the inception phase and which addresses the top technical risks of the project.

This is an evolutionary prototype of production quality code, which becomes the architectural baseline, but it does not exclude the development of one or more exploratory, throwaway prototypes to mitigate specific risks: refinements of the requirements, feasibility, human-interface studies, demonstrations to investors, etc.

At the end of this phase, there is again a "go-no go-" decision point to actually invest and build the product (or bid for the complete development of the contract). The plans produced must be detailed enough, and the risks sufficiently mitigated to be able to determine with accuracy the cost and schedule for the completion of the development.

Entry criteria:

- The products and artifacts described in the exit criteria of the previous phase.
- The plan was approved by the project management and funding authority, and the resources required for the elaboration phase have been allocated.

Exit criteria:

- A detailed software development plan that contains
  - An updated risk assessment.
  - A management plan.
  - A staffing plan.
  - A phase plan that shows the number and contents of the iteration.
  - An iteration plan that details the next iteration.

- The development environment and other tools required.
- A test plan.
- A baseline vision in the form of a set of evaluation criteria for the final product.
- Objective, measurable evaluation criteria to assess the results of the initial iteration(s) of the construction phase.
- A domain analysis model (80 percent complete), sufficient to be able to call the corresponding architecture complete.
- A software architecture description (that states constraints and limitations).
- An executable architecture baseline.

### **Construction Phase**

This phase is broken down into several iterations, fleshing out the architecture baseline and evolving it in steps or increments toward the final product. At each iteration, the various artifacts prepared during the elaboration phase (see above) are expanded and revised, but they ultimately stabilize as the system evolves in correctness and completeness. New artifacts are produced during this phase beside the software itself: documentation, both internal and for end users, test beds and test suites, and deployment collaterals to support the next phase: marketing collaterals, for example.

For each iteration there are

Entry criteria:

- The product and artifacts of the previous iteration. The iteration plan must state the iteration specific goals:
  - Additional capabilities being developed, which use cases or scenarios, will be covered.
  - Risks being mitigated during this iteration.
  - Defects being fixed during the iteration.

Exit criteria:

The same products and artifacts updated, plus:

- A release description document, which captures the results of an iteration.
- Test cases and results of the tests conducted on the products.
- An iteration plan that details the next iteration.
- Objective measurable evaluation criteria to assess the results of the next iteration(s).

Toward the end of the construction phase, the following artifacts must be produced and are additional exit criteria for the last iteration of the phase:

- A deployment plan, specifying as necessary:
  - Packaging.
  - Pricing.
  - Roll out.
  - Support.
  - Training.
  - Transition strategy, e.g., an upgrade plan from an existing system.
  - Production, e.g., making floppies and manuals.
- User documentation.

### **Transition Phase**

The transition phase is the phase where the product is put in the hands of its end users. It involves issues of marketing, packaging, installing, configuring, supporting the user-community, making corrections, etc.

From a technical perspective, the iterations continue with one or more releases (or deliveries): beta releases, general availability releases, bug fix, or enhancement releases.

The phase is completed when the user community is satisfied with the product: formal acceptance, for

example, in a contractual setting, or when all activities on this product are terminated. It is the point where some of the accumulated assets can be made reusable by the next cycle or by some other projects.

Entry criteria:

- The product and artifacts of the previous iteration and in particular, a software product sufficiently mature to be put into the hands of its users.

Exit criteria:

- An update of some of the previous documents, as necessary, the plan being replaced by a "post-mortem" analysis of the performance of the project relative to its original and revised success criteria.
- A brief inventory of the organization's new assets as a result this cycle.

### **Evolution Cycles**

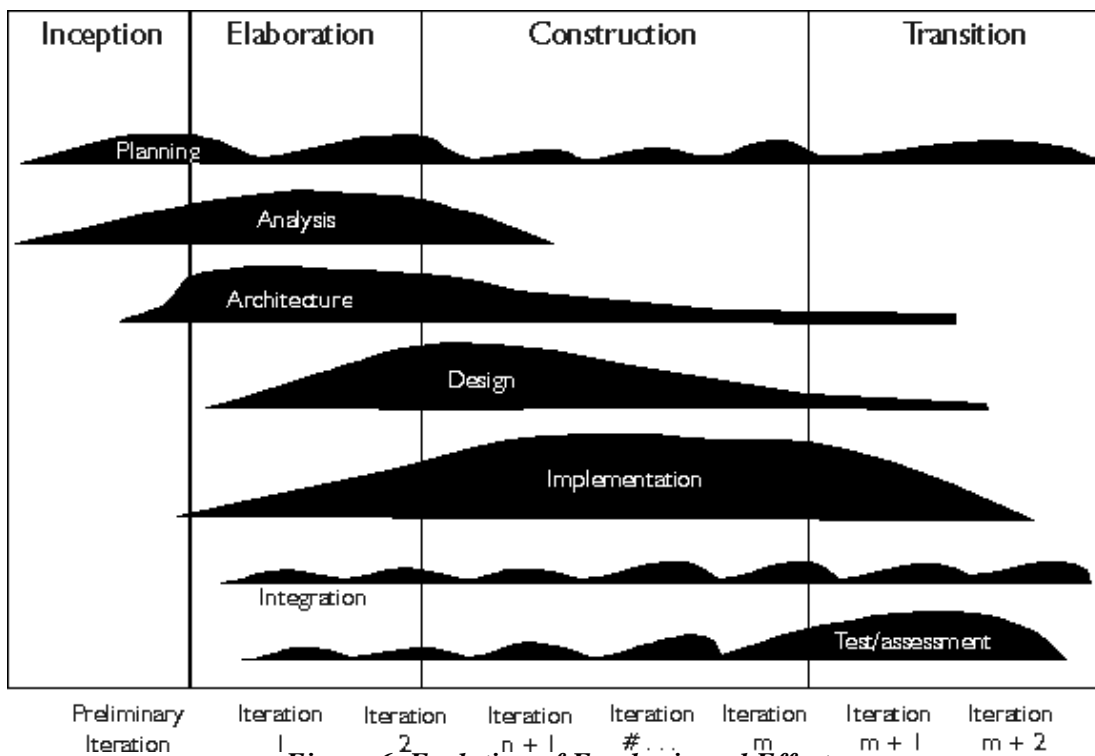
For substantial evolutions, we apply the whole process recursively, starting again at the inception phase, for a new cycle. Since we already have a product, this inception phase may be considerably reduced, compared to an initial development cycle. The elaboration phase also may be limited and focused more on the planning aspects than evolving the analysis or the architecture. Said otherwise: cycles can slightly overlap.

Minor evolutions are done in extending the transition phase, adding one or more iterations.

Alternatively, the transition phase may be concluded by an end-of-life process, i.e., the product does not evolve any more, but some specific actions must be taken to terminate it or retire it.

### **Activities in the Process**

The names of the phases of the process stay away from the terms that describe an intellectual activity: analysis, design, test, etc., so that it will be understood that this activity is not only confined to that phase but also remain independent of terms employed by other authors, standards, and domain-specific jargon. These activities do take place but in varying degree in each phase and iteration. Figure 6 illustrates how the emphasis and effort evolves over time.



This change of focus also explains that, although all structured in the same way, the exact nature and contents of the iterations evolves over time.

This also shows that the beginning of an activity is not bound to the end of another, e.g., design does not start when analysis completes, but the various artifacts associated with the activities are revised as the problem or the requirements are better understood.

Finally, in an iterative process, the activities of planning, test, and integration are spread incrementally throughout the cycle, in each iteration, and not massively lumped at the beginning and at the end, respectively. They do not appear as separate steps or phases in the process.

Although this will vary considerably depending on the project discriminants, a typical initial development cycle for a medium size project should anticipate the following ratios for various activities:

- Planning and management 15%
- Analysis and requirements 10%
- Design and integration 15%
- Implementation and functional tests 30%
- Measurement, assessment, and acceptance test 15%
- Tools, environment, and change management 10%
- Maintenance (fixes during development) 5%

### Lifecycle Artifacts

The process is not document driven: its main artifact must remain at all time the software product. The documentation should remain lean and limited to the few documents that bring real value to the project from a management or technical point of view. We suggest the following typical set of documents.

#### Management Artifacts

The management artifacts are not the product but are used to drive or monitor the progress of the project, estimate the risks, adjust the resources, and give visibility to the customer (in a contractual setting) or the

investors.

- An *organizational policy* document - the codification of the organization's process and contains an instance of this generic process.
- A *vision* document - describes the system level requirements, qualities, and priorities.
- A *business case* document - the financial context, contract, projected return on investment, etc.
- A *development plan* document - contains in particular the overall iteration plan and the plan for the current and upcoming iteration.
- An *evaluation criteria* document - contains the requirements, acceptance criteria, and other specific technical objectives, which evolves from major milestone to major milestone. It contains the iteration goals and acceptance levels.
- *Release description* documents for each release.
- *Deployment* document - gathering additional information useful for transition, training, installation, sales, manufacturing, and cut-over.
- *Status assessment* documents - periodic snapshots of project status, with metrics of progress, staffing, expenditure, results, critical risks, actions items, post-mortem.

### Technical Artifacts

These artifacts are either the delivered goods (executable software and manuals) or the blueprints that were used to manufacture the delivered goods (software models, source code, and other engineering information useful to understand and evolve the product).

- *User's manual* - developed early in the lifecycle.
- *Software documentation* - preferably in the form of self-documenting source code and models (uses cases, class diagrams, process diagrams, etc.) captured and maintained with appropriate computer-aided software engineering tools.
- A *Software Architecture* document - extracted (abstracted) from the software documentation, that describes the overall structure of the software, its decomposition in major elements (class categories, classes, processes, subsystems, the definition of critical interfaces), and rationale for the key design decisions.

The artifacts enumerated in the entry and exit criteria can all be mapped onto one of these 11 documents.

Depending on the type of project, this typical document set can be extended or contracted; some documents can be merged. The documents do not have to be paper documents. They can be spreadsheets, text-files, database, annotations in source code, hypertext documents, etc., but the corresponding information source must be clearly identified and easily accessible and some of its history preserved.

### Requirements

The process is not requirement-driven either. The requirements for the product evolve during a cycle, and take different forms:

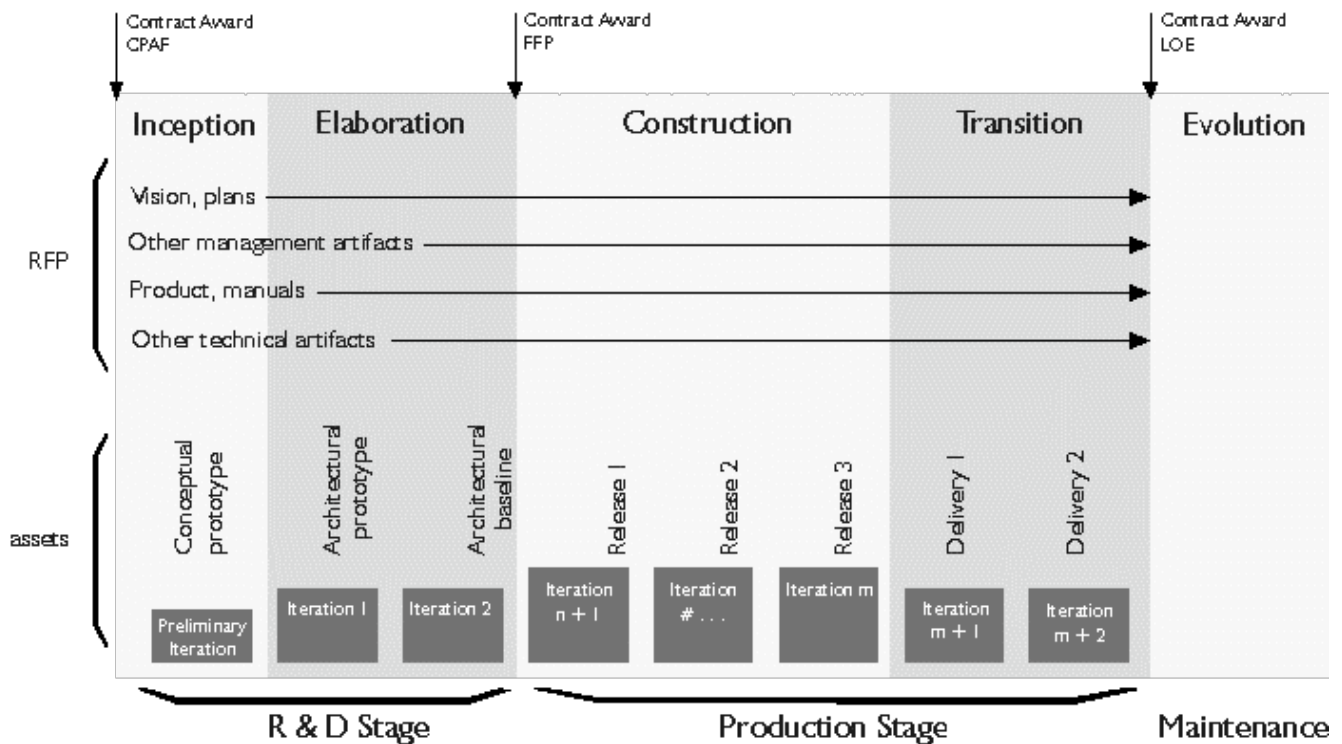
- The business case gives the main constraints, mostly in terms of resources that can be expended.
- The vision document describes only the key requirements of the system from a user's perspective, and it evolves only slowly during the development cycle.
- The more detailed requirements are elaborated during the elaboration phase in the form of use cases and scenarios and are refined incrementally throughout the construction phase, as the product and the users needs become better understood. These more detailed requirements are in the evaluation criteria document; they drive the definition of the contents of the construction and transition iterations and are referenced in the iteration plan.

### Examples of Rational's Processes

The process takes different aspects depending on the discriminants described earlier in this article. Following are two extreme examples:

#### Rational's Process for Large Contractual Software Development

Rational proposes to set up the procurement of large software in three stages, associated with three kinds of contracts (*Figure 7*).



**Figure 7. Software Development Over Course of Large Contract.**

- A research and development stage, comprising the inception and elaboration phase, typically bid in a risk-sharing manner, e.g., as a cost plus award fee contract.
- A production stage, comprising the construction and transition phases, typically bid as a firm, fixed price contract.
- A maintenance stage, if any, corresponding to the evolution phase, typically bid as a level of effort contract.

Due to the higher level of visibility that is required from the customer on the evolution of the project, and because of the larger number of people and organizations involved, more formalism is required in the process, and there may be more emphasis on written artifacts than would be the case in a small, internal project. All 11 management and technical documents are present in some form or name.

### Rational's Process for a Small Commercial Software Product

At the other extremity of the scale of the family of processes, a small commercial development would see a more fluid process with only limited amount of formalism at the major milestones and a more limited set of documents:

- A product vision.
- A development plan, showing schedule and resources.
- Release description documents, specifying the goal of an iteration at the beginning of the iteration, and updated to serve as release notes at the end.
- User documentation, as necessary.

Software architecture, software design, development process, and procedures can be documented by the code or the software development environment.

### Conclusion

Rational's process puts an emphasis on addressing very early high-risks areas by developing rapidly an initial version of the system, that defines its architecture. It does not assume a fixed set of firm requirements at the inception of the project, but allows it to refine the requirements as the project evolves. It expects and accommodates changes. The process does not put either a strong focus on documents or "ceremonies," and it lends itself to the automation of many of the tedious tasks associated with software development. The main focus remains the software product itself, and its quality, as measured by the degree to which it satisfies its end-users, meets its return on investment objective altogether.

A process derived from the generic process described here would fully conform to the requirements of a standard such as ISO 9000.

### Glossary

**Artifact:** Any document or software other than the software product.

**Baseline:** For business and technology managers. A release that is subject to change management and configuration control.

**Construction:** The third phase of the process where the software is brought from an executable architectural baseline to the point where it is ready to be transitioned to its user's community.

**Cycle:** One complete pass through the four phases: inception, elaboration, construction, and transition. The span of time between the beginning of the inception phase and the end of the transition phase.

**Elaboration:** The second phase of the process where the product vision and its architecture are defined.

**Evolution:** The life of the software after its initial development cycle; any subsequent cycle where the product evolves.

**Generation:** The result of one software development cycle.

**Inception:** The first phase of the process where the seed--idea, RFP, or previous generation--is brought up to the point of being (at least internally) founded to enter into the elaboration phase.

**Iteration:** A distinct sequence of activities with a baselined plan and an evaluation criterion.

**Milestone:** An event held to formally initiate and conclude an iteration.

**Phase:** The span of time between two major milestones of the process where a well-defined set of objectives are met, artifacts are completed, and decisions are made whether to move into the next phase.

**Product:** The software that is the result of the development and some of the associated artifacts (documentation, release medium, training).

**Prototype:** A release that is not necessarily subjected to change management and configuration control.

**Release:** A subset of the end-product that is the object of evaluation at a major milestone (see: prototype, baseline).

**Risk:** An ongoing or upcoming concern which has a significant probability of adversely affecting the success of major milestones.

**Transition:** The fourth phase of the process where the software is turned into the hands of the user's community.

**Vision:** The user's view of the product to be developed.

### About the Author

Philippe Kruchten is director of Process Development at Rational Software Corp., in charge of further development of the commercial version of the process described above. He has been a software designer and a technical consultant for 21 years, involved in large projects around the world in telecommunication, compilation, defense, avionics, and air traffic control. His interests cover software architecture, software process, object-oriented design, and programming. He was a distinguished reviewer for Ada 95. He received a master's degree in mechanical engineering from Ecole Centrale de Lyon (France) and a doctorate in computer science from the French National Institute of Telecommunications in Paris, France.

Rational Software Corp.  
240-10711 Cambie Road  
Richmond BC V6X3G5 Canada  
Voice: 1-604-231-3706  
Fax: 604-231 3720  
E-mail: [pkruchten@rational.com](mailto:pkruchten@rational.com)

### Further Readings

- Boehm, B. W., "A Spiral Model of Software Development and Enhancement," *IEEE Computer.*, 21 (5), May 1988, pp. 61-72.
  - Booch, G., *Object Solutions: Managing the Object-Oriented Project*, Addison-Wesley, Redwood City, Calif., 1996.
  - Devlin, M. T. and W. E. Royce, *Improving Software Economics in the Aerospace and Defense Industry*, Technical paper TP-46, Rational Software Corp., Santa Clara, Calif, 1995.
  - Gilb, T., *Principles of Software Engineering Management*, Addison-Wesley, Wokingham, England, 1988.
  - Humphrey, W., *Managing the Software Process*, Addison-Wesley, Reading, Mass., 1989.
  - Kruchten, Phillipe, "Un processus de developement de logiciel it eratif et centre sur l'architecture," *Proceedings of the 4th International Conference on Software Engineering*, Toulouse, France, December 1991, EC2.
  - Parnas, D. L. and P. C. Clements, "A Rational Design Process: How and Why to Fake It," *IEEE Transactions on Software Engineering*, SE-12 (2), February 1986, pp. 251-257.
-