

[▶ subscribe](#)
[▶ contact us](#)
[▶ submit an article](#)
[▶ rational.com](#)
[▶ issue contents](#)
[▶ archives](#)
[▶ mission statement](#)
[▶ editorial staff](#)

## ▶ The Ten Essentials of RUP



by [Leslee Probasco](#)  
Development Manager  
Rational Unified Process  
Rational Software Canada

*To effectively apply the Rational Unified Process (affectionately known as "RUP"), it is important to first understand its key objectives, why each is important, and how they work together to help your*

*development team produce a quality product that meets your stakeholders' needs.*

### **Camping Trip? Software Project? Identify Essentials First**

The other evening, my neighbor Randy came over to ask for help: He was preparing for a weekend camping and hiking trip and trying to determine what gear to pack. He knows that I often lead and participate in wilderness trips and was impressed with how I'm able to quickly and efficiently determine what items to cram into my limited packing space, while referring to a list of all the equipment and clothing I own. "Do you think I could borrow that list?" he asked.

"Sure, but I'm afraid it won't be much help," I explained. You see, I have literally *hundreds* of items on my outdoor gear list, covering many different types of outings -- from backpacking and climbing, to skiing, snow-shoeing, ice-climbing, and kayaking -- and for trip lengths ranging from simple day trips to multi-day expeditions. I knew that without some guidance, Randy would probably not be able to wade through the multitude of items on my list and figure out what *he* really needed for his relatively simple outing.

#### **Start With Essentials, Then Add the Extras**

Instead, I offered to look through the items Randy had already crammed into his bulging pack. We could see what he might eliminate to lighten his load and also whether any necessary items were missing. Within a short time, I could tell that what he really lacked was an understanding of what were the "essentials" for any wilderness outing.

I pulled out a blank sheet of paper and listed ten items<sup>1</sup>:

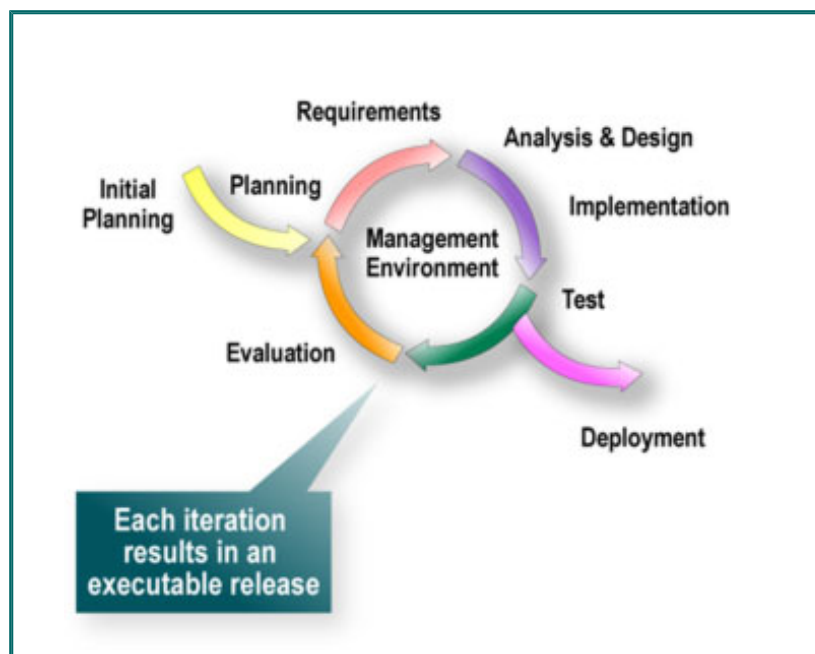
1. Map
2. Compass
3. Sunglasses & sunscreen
4. Extra clothing
5. Extra food & water
6. Headlamp
7. First-aid kit
8. Fire-starter
9. Matches
10. Knife

"Here, Randy. This is the list you really need. If you start with these 'ten essentials,' then for any trip, the other necessary items will become obvious." I memorized this list many years ago, when I first started mountaineering, and I still refer to it -- no matter what type of trip I am preparing for or how long I'll be gone. Each of these items scales up or down, depending on the trip, but starting with a "short list" and expanding it as needed is much easier than starting from a long list and trying to decide what *not* to take.

### Applying This Lesson to RUP

Often, as I help project teams sort through the many elements in RUP, I hear questions such as: "How do I sort through all of these items and determine which ones I need for my project?" "RUP contains so much information. It must be intended only for big projects -- can I really use it for my small one?"

What these folks really need, I've decided, is a "Ten Essentials of RUP," similar to the simple list I gave to my friend Randy -- one that would serve as a reasonable starting point for any project: small, medium, or large. This list would focus on what I call "the



essence" of RUP -- or of any effective software process, for that matter.

For those of you who are unfamiliar with RUP, this diagram represents the incremental and iterative nature of the Rational Unified Process, which covers a vast array of disciplines within the software development lifecycle, including artifacts, guidelines, team member roles, and activities.

Often, projects get bogged down with the details in one particular area before all the participants understand the key process elements required to fully produce and deliver a quality product. Then, when the project falls behind, the blame is placed on some activity that may have been over-emphasized or whose usefulness they don't understand: "See, I told you that requirements management (or use cases, or collecting project metrics, or using configuration management, or using a defect tracking tool, or having status meetings) would slow us down!"

Having an "Essentials" list allows team members to take a more systematic and holistic approach to the overall process of developing software. Once a process framework or "architecture" is in place, *then* team members can more effectively focus on individual problem areas (and often, I'll admit, requirements management is right there at the top of the list). It is also important to identify and prioritize obvious problems and their associated risks at the outset, so the team can apply early mitigation strategies as needed.

## **The Ten Essentials of RUP**

So, what should be on the "Ten Essentials of RUP" list? Here are my choices:

1. Develop a Vision
2. Manage to the Plan
3. Identify and Mitigate Risks
4. Assign and Track Issues
5. Examine the Business Case
6. Design a Component Architecture
7. Incrementally Build and Test the Product
8. Verify and Evaluate Results
9. Manage and Control Changes
10. Provide User Support

Let's look at each of these items individually, see where they fit into RUP, and find out why each made my "short list."

### **1. Develop a Vision**



Having a clear vision is key to developing a product that meets your stakeholders' *real* needs.

The Vision captures the "essence" of the *Requirements Workflow* in RUP: analyzing the problem, understanding stakeholder needs, defining the system, and managing the requirements as they change.

The Vision provides a high-level, sometimes contractual, basis for more detailed technical requirements. As the term implies, it is a clear, and usually high-level, view of the software project that can be articulated to any decision maker or implementer during the process. It captures very high-level requirements and design constraints, giving the reader an understanding of the system to be developed. It also provides input for the project-approval process, and is therefore intimately related to the Business Case. And finally, because the Vision communicates the fundamental "why's and what's" of the project, it serves as a means for validating future decisions.

The Vision statement should answer the following questions, which can also be broken out as separate, more detailed items:

- What are the key terms? (Glossary)
- What problem are we trying to solve? (Problem Statement)
- Who are the stakeholders? Who are the users? What are their respective needs?
- What are the product features?
- What are the functional requirements? (Use Cases)
- What are the non-functional requirements?
- What are the design constraints?

## 2. **Manage to the Plan**

"The product is only as good as the plan for the product."<sup>2</sup>

In RUP, the Software Development Plan (SDP) aggregates all information required to manage the project and may encompass a number of separate items developed during the Inception phase. It must be maintained and updated throughout the project.

The SDP defines the project schedule (including Project Plan and Iteration Plan) and resource needs (Resources and Tools), and is used to track progress against the schedule. It also guides planning for other process components: Project Organization, Requirements Management Plan, Configuration Management Plan, Problem



Resolution Plan, QA Plan, Test Plan, Test Cases, Evaluation Plan, and Product Acceptance Plan.

In a simple project, statements for these plans may consist of only one or two sentences. A Configuration Management Plan, for example, may simply state: "At the end of each day, the contents of the project directory structure will be zipped, copied onto a dated, labeled zip disk, marked with a version number, and placed in the central filing cabinet."

The format of the Software Development Plan is not as important as the activity and thought that go into producing it. As Dwight D. Eisenhower said, "The plan is nothing; the planning is everything."

"Manage to the Plan" -- together with essentials #3, #4, #5, and #8 in our list above -- captures the essence of the *Project Management Workflow* in RUP, which involves conceiving the project, evaluating scope and risk, monitoring and controlling the project, and planning for and evaluating each iteration and phase.

### 3. Identify and Mitigate Risks



An essential precept of RUP is to identify and attack the highest risk items early in the project. Each risk the project team identifies should have a corresponding mitigation plan. The risk list should serve both as a planning tool for project activities and as the basis for specifying iterations.

### 4. Assign and Track Issues

Continuous analysis of objective data derived directly from ongoing activities and evolving product configurations is important in any project. In RUP, regular status assessments provide the mechanism for addressing, communicating, and resolving management issues, technical issues, and project risks. Once the appropriate team has identified the hurdles, they assign all of these issues a due date and a person with responsibility for resolving them. Progress should be tracked regularly, and updates should be issued as necessary.



These project "snapshots" highlight issues requiring management attention. While the period may vary, regular assessment enables managers to capture project history and remove any roadblocks or bottlenecks that restrict progress.

### 5. Examine the Business Case

The Business Case provides the necessary information, from a business standpoint, to determine whether the project is a worthwhile investment. The Business Case also helps develop an economic plan for realizing the project Vision. It provides justification for the project and establishes economic constraints. As the project proceeds, analysts use the Business Case to accurately assess return on investment (ROI).



Rather than delve deeply into the specifics of a problem, the Business Case should create a brief but compelling justification for the product that all project team members can easily understand and remember. At critical milestones, managers should return to the Business Case to measure actual costs and returns against projections and decide whether to continue the project.

## 6. Design a Component Architecture

In the Rational Unified Process, a software system's architecture (at a given point in time) is defined as the organization or structure of the system's significant components interacting, through interfaces, with components composed of successively smaller components and interfaces. What are the main pieces? And how do they fit together?



RUP provides a methodical, systematic way to design, develop, and validate such an architecture. The steps involved in the *Analysis and Design Workflow* include defining a candidate architecture, refining the architecture, analyzing behavior, and designing components of the system.

To speak and reason about software architecture, you must first create an architectural representation that describes important aspects of the architecture. In RUP, this is captured in the Software Architecture Document, which presents multiple views of the architecture. Each view addresses a set of concerns specific to a set of stakeholders in the development process: end users, designers, managers, system engineers, system administrators, and so on. The document enables system architects and other project team members to communicate effectively about architecturally significant project decisions.

## 7. Incrementally Build and Test the Product

The essence of the *Implementation* and *Test* workflows in RUP is to incrementally code, build, and test system components throughout the project lifecycle, producing executable releases at the end of each iteration after inception.



At the end of the elaboration phase, an architectural prototype is

available for evaluation; this might also include a user-interface prototype, if necessary. Then throughout each iteration of the construction phase, components are integrated into executable, tested builds that evolve into the final product. Also key to this essential process element are ongoing Configuration Management and review activities.

## 8. Verify and Evaluate Results



As the name implies, the Iteration Assessment in RUP captures the results of an iteration. It determines to what degree the iteration met the evaluation criteria, including lessons learned and process changes to be implemented.

Depending on the scope and risk of the project and the nature of the iteration, the assessment ranges from a simple record of a demonstration and its outcomes to a complete, formal test review record.

The key here is to focus on process problems as well as product problems. The sooner you fall behind, the more time you will have to catch up.

## 9. Manage and Control Changes

The "essence" of RUP's *Configuration and Change Management Workflow* is to manage and control the scope of the project as changes occur throughout the project lifecycle. The goal is to consider all stakeholder needs and meet them to whatever extent possible, while still delivering a quality product, on time.

As soon as users get the first prototype of a product (and often even before that!), they *will* request changes. It is important that these changes be proposed and managed through a consistent process.

In RUP, Change Requests are used to document and track defects, enhancement requests, and any other type of request for a change to the product. They provide an instrument for assessing the impact of a potential change as well as a record of decisions made about that change. They also help ensure that all project team members understand the potential impact of a proposed change.



## 10. Provide User Support



In RUP, the "essence" of the *Deployment Workflow* is to wrap up and deliver the product, along with whatever materials are necessary to assist the end-user in learning, using, and maintaining the product.

At a minimum, a project should supply users with a User's Guide -- perhaps implemented through online Help -- and possibly an Installation Guide and Release Notes.

Depending on the complexity of the product, users may also need training materials. Finally, a Bill of Materials clearly documents what should be shipped with the product.

## **What about Requirements?**

Some of you may look at my list of essentials and vehemently disagree with my choices. You may ask, for example, where "requirements" fit into this picture. Aren't they essential? I'll tell you why I have not included them on my list. Sometimes I'll ask a project team (especially a team for an internal project), "What are your requirements?" and receive the response, "We don't really have any requirements."

This amazed me at first (since I come from a military-aerospace development background). How could they not have any requirements? As I talked to these teams further, I found out that to them, "requirements" meant a set of externally imposed "shall" statements about what they must have or the project will be rejected -- and they really don't have any of those! Especially if a team is involved in research and development, the product requirements may evolve throughout the project.

So for these projects, I follow up their response with another question: "Okay, then what is the vision for the product?" Then their eyes light up. We move easily through each of the questions listed as bullet points under RUP essential #1 above ("Develop a Vision"), and the requirements just flow naturally.

For teams working on contracts with specified requirements, it may be useful to have "Meet Requirements" on their essentials list. Remember, my list is meant only as a starting point for further discussion.

## **Summary: Applying the Ten Essentials**

So, how can discovery of the "Ten Essentials of RUP" make a difference in my life? Here are a few ways that these recommendations can help me work with projects of varying sizes.

### **For Very Small Projects**

First of all, if someone asks me how they might use RUP and the Rational development tools for building a simple product with a very small, inexperienced team that is just learning about process, I can share my "Ten Essentials" list and avoid overwhelming the project team with all the details in RUP and the full power of the Rational Suites of tools.

In fact, these ten essentials can be implemented without *any* automated tool support! A project notebook with one section devoted to each of the ten essentials is actually a very good starting point for managing a small project. (And I have found Post-It Notes invaluable for managing, prioritizing, and tracking change requests on small projects!)

### **For Growing Projects**

Of course, as a project's size and complexity grow, these simple means of applying the ten essentials soon become unmanageable, and the need for automated tools will become more obvious. Nevertheless, I would still encourage team leaders to start with the "Ten Essentials" and "Best Practices" of RUP and incrementally add tool support as needed, rather than immediately attempt to fully utilize the complete set of tools in the Rational Suites.

### **For Mature Project Teams**

For more mature project teams that may already be applying a software process and using development tools, the "Ten Essentials" can help provide a quick method for assessing the balance of key process elements and identify and prioritize areas for improvement.

### **For All Projects**

Of course, each project is different, and it may seem that some projects don't really need all of these "essentials." In these cases, it is also important to consider what will happen if your team ignores any of these essentials. For example:

- No vision? You may lose track of where you are going and wind up taking unproductive detours.
- No plan? You will not be able to track progress.
- No risk list? Your project is in danger of focusing on the wrong issues now and may get blown up by an undetected "land mine" five months from now.
- No issues list? Without timely analysis and problem solving, small issues often evolve into major roadblocks.
- No business case? You risk losing time and money on the project. Eventually it may run out of funds and be cancelled.
- No architecture? You may be unable to handle communication, synchronization, and data access issues as they arise. You may also have problems with scaling and performance.
- No product (prototype)? You won't be able to test effectively and will also lose credibility with customers.
- No evaluation? You'll have no way of knowing how close you really are to meeting your deadlines, project goals, and budget.
- No change requests? You'll have no way to assess the potential impact of changes, prioritize competing demands, and update the

whole team when changes are implemented.

- No user support? Users will not be able to use the product to best advantage, and tech support may be overwhelmed with requests for help.

So, there you have it -- it's very risky to live without knowledge of the "Ten Essentials." I encourage you to use these as a starting point for your project group. Decide what you want to add, change, or take away. Then, decide what else is really "essential" for your project -- no matter what size it may be -- to deliver a product on time, within budget, that meets your stakeholders' *real* needs!



## Other Essentials

Other organizations have published similar lists of software project essentials. IEEE Software Magazine, March/April 1997, included an article by Steve McConnell, "Software's Ten Essentials." The Software Project Manager's Network includes a listing of "**16 Critical Software Practices**," available at [www.spmn.com](http://www.spmn.com). And the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) contains Key Process Areas (KPAs) which might also be considered "essentials" (see [www.sei.cmu.edu](http://www.sei.cmu.edu)).

---

<sup>1</sup>For a complete analysis of this "Ten Essentials" list, see pages 35-41 of *Mountaineering: The Freedom of the Hills*, 6th edition, published by The Mountaineers of Seattle, WA in 1997.

<sup>2</sup>From the *Johnson Space Center Shuttle Software Group*. Quoted in "They Write the Right Stuff," by Charles Fishman, *Fast Company*, Issue 6, p. 95, December 1996.

---

**For more information on the products or services discussed in this article, please click [here](#) and follow the instructions provided.**  
**Thank you!**