The Ten Essentials of RUP—the Essence of an Effective Development Process

Leslee Probasco

Rational Software White Paper

TP177, 9/00

Rational®
the software development company

Table of Contents

Abstract **1**

A Plea for Help **1**

Overwhelmed with the Details **1**

Focus on the Essentials **1**

Start with a "Short List" **2**

What Should be on Your List? "Well it depends…" **2**

Build a Framework First **2**

The Ten Essentials of RUP **3**

Vision — Develop a Vision 4
Plan — Manage to the Plan 4
Risks — Identify and Mitigate Risks 5
Issues — Assign and Track Issues 5
Business Case — Examine the Business Case 5
Architecture — Design a Component Architecture 5
Product — Incrementally Build and Test the Product 5
Evaluation — Regularly Assess Results 6
Change Requests — Manage and Control Changes 6
User Support — Provide Assistance to the User 6

What about Requirements? **6**

What about Test? **7**

Summary: Applying the Ten Essentials **7**

For Very Small Projects 7
For Growing Projects 7
For Mature Project Teams 7

Abstract

To effectively apply a software development process such as the Rational Unified Process[1] (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' "real" needs.

A Plea for Help

The other evening, my neighbor Randy came over to ask for some help: He was preparing for an weekend camping and hiking trip with his church group and was trying to determine what gear[1] to pack for the outing. He already had some clothing and equipment, and was undoubtedly going to have to buy some more. Could he please borrow my gear list?

My gear list? Yes. It appears that he was over visiting one day when I was preparing for a backpacking and climbing trip. He knows that I am quite experienced in leading and participating in wilderness trips and was very impressed that I was able to quickly and efficiently determine the items to fit into my limited packing space by referring to a list I have of much of my inventoried equipment and clothing. Could he please borrow that list?

Well, sure, he was welcome to borrow my gear list — but I was afraid it wouldn't be much help. Why not? Didn't I think that everything he needed would be on that list?

Actually, I was quite sure that everything he needed would at least be *represented* by something on my list, but more than likely, the exact items he needed for this particular trip might not be on the list at all. For example, not only is his shoe size *much* bigger than mine, but he no doubt prefers different food (and quantities) than I might choose to take along.

Overwhelmed with the Details

The fact is that I have literally *hundreds* of items on my outdoor gear list, covering many different outing types from backpacking and climbing, to skiing, snow-shoeing, ice-climbing and kayaking – and covering trip lengths from simple day trips to multi-day expeditions. And I have different items I would take along if I were leading a trip for an organization (such as trip sign-up sheet and waiver forms), than if I were just heading out into the wilderness with some of my buddies. My main concern was that he would not be able to wade through the multitude of items on the list and figure out what *he* really needed for his relatively simple outing.

Randy threw up his hands in exasperation! He'd never be ready in time to be properly prepared for his hike. How would he know what to pack? He had started packing last week — he had done extensive research on the Internet, bought fancy new boots, jackets and clothing — his pack was already full!

And he hadn't even packed any food yet! Oh, yeah, and didn't he need to take water, too? And what if something went wrong? What if they got lost? Or someone got hurt?

Focus on the Essentials

As I sorted through the items Randy had already put into his pack, I could easily see that he did not have a balanced view of the essential items needed on a wilderness outing.

Do you have the "ten essentials"? I asked. Ten essentials? What are those?

Here, Randy, I have just the list you need. I pulled out a blank sheet of paper and wrote at the top: "The Ten Essentials". On it, I wrote 10 items:

1. Map
2. Compass
3. Sunglasses and sunscreen

[1] "Gear", a slang term used by mountaineers and other outdoor enthusiasts to denote any manner of equipment, tools, clothing, footwear or other artifacts used in their various and sundry sports. If you think RUP has a lot of artifacts, see: www.mgear.com.

4. Extra clothing
5. Extra food and water
6. Headlamp
7. First-aid kit
8. Fire-starter
9. Matches
10. Knife[2]

Start with a "Short List"

That's it? Yup! That's it! If you start with these 10 elements, making sure you've got each area covered, then the other things will fall into place as needed. Of course, each of these "essentials" will scale 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.

As you gain experience, you'll evolve your own "long list" which will make sense to you as time goes on. At that point, it may be helpful for you to look at my list and others' to see how you might properly expand your own. But no two person's lists will ever look exactly the same (unless they've cheated!).

I memorized this same 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 intend to be gone. If you ask any experienced mountaineer, "Do you have the 'ten essentials'?", most of them will know exactly what it is you're asking about. Actually, different groups or individuals may have a different list of "ten essentials", but in essence, they are the same. And for each trip, the actual items I pack may be entirely different, but I can still say that I have the "ten essentials".

What Should be on Your List? "Well it depends…"

So, how does this relate to RUP? Often, as I help project teams sort through the many elements in RUP (at last count, I found 4 phases, 9 core workflows, 31 workers, 103 artifacts, 136 activities, plus more guidelines, checklists and tool mentors than I would care to count!), I hear questions such as: "How do I sort through all of these items and determine which I need for my project?" "Do I need this one?" "Isn't RUP is only for big projects?"

Most often the answer is (one of our favorites): "Well, it depends…"

What I really want, I've decided, is a list of "Ten Essentials of RUP" that I can give to folks searching for process guidance—like I did for my friend Randy. This list could serve as a reasonable starting point for determining the right elements to include for any project and could apply if they were going for a short day hike (a very small project); an overnight backpack, ski or kayak trip (a medium-sized project in different domains); or a major Everest-type expedition (a very large, critical project). The idea is to focus on what I call "the essence" of RUP—or really of any effective software process. This also ties back to our concepts of "best practices"[3] in software development.

Build a Framework First

A common problem I see in many projects, is that they will often focus heavily in one particular area to the extent that they get bogged down with the details of that particular area before making sure that they have a good idea of the "key" elements involved in the whole process lifecycle of producing a quality product.

Then, when the project falls behind they say, "See, I told you that requirements management would just slow us down!". You can replace "requirements management" in the previous statement with "use cases", "collecting project metrics", "using

---

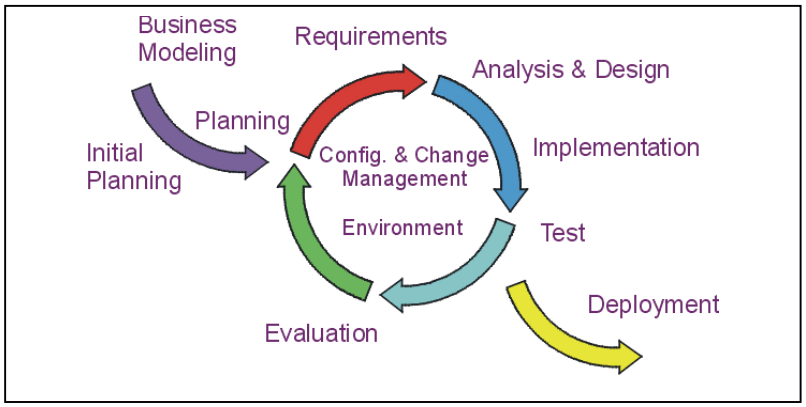
[2] For a complete analysis of this list of "Ten Essentials", I refer you to the excellent book, "Mountaineering – The Freedom of the Hills", 6th Edition, The Mountaineers, Seattle, WA, 1997. pp. 35-41.

[3] The "best practices" that Rational has been touting over the last few years include: (1) Develop Iteratively, (2) Manage Requirements, (3) Use Component Architectures, (4) Model Visually, (5) Verify Quality, and (6) Control Changes.

configuration management", "using a defect tracking tool", "having status meetings" or any of a number of phrases that I'm sure you're familiar with.

I believe it is much more effective to take a more systematic and holistic approach, making sure that the key elements of a process are in place (an architecture, so to speak) before determining to focus on any one particular problem area.

Once this framework (or architecture) for a quality software process is in place, then I believe a project can effectively focus on a particular area which is identified as being a major contributor to their problems (and usually, I'll admit, requirements management is often right there at the top of the list[4]). Keep in mind, though, that the RUP approach is to base this selection on identifying and prioritizing risks for the project, and determining early mitigation strategies for those identified risks.



The Ten Essentials of RUP

So, what are the "Ten Essentials of RUP"? The following list describes what I believe to be the minimal set of items a project will have in place if they are truly following the "essence" of the Rational Unified Process:

1. Vision
2. Plan
3. Risks
4. Issues
5. Business Case
6. Architecture
7. Product
8. Evaluation
9. Change Requests
10. User Support

Let's look at each of these items individually, see where they fit in RUP and try to understand why each made my "short list" of essential items.

[4]Since Requirements Management is one of the initial "Key Process Areas" of the Capability Maturity Model (CMM), this is not at all an unlikely scenario for those determining to improve their software development process.

Vision — Develop a Vision

Having a clear Vision is key to developing a product that meets your stakeholders' *real* needs".[5]

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. The Vision captures very high-level requirements and design constraints, to give the reader an understanding of the system to be developed. It provides input to the project-approval process, and is therefore intimately related to the Business Case. It communicates the fundamental "why's and what's" related to the project and is a gauge against which all future decisions should be validated.

The contents of the Vision should answer the following questions, which might be broken out to separate, more detailed, artifacts, as needed:

- 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 needs?
- What are the product features?
- What are the functional requirements? (Use Cases)
- What are the non-functional requirements?
- What are the design constraints?

Plan — Manage to the Plan

"The product is only as good as the plan for the product."[6]

In RUP, the Software Development Plan (SDP) gathers all information required to manage the project. It may enclose a number of separate artifacts developed during the Inception phase and is maintained throughout the project.

The SDP is used to plan the project schedule and resource needs, and to track progress against the schedule. It addresses such areas as: Project Organization, Schedule (Project Plan, Iteration Plan, Resources, Tools), 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, these may include only one or two sentences each. For example a CM Plan 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 itself is not as important as the activity and thought that go into producing it. So, I don't even really care what it looks like – or what tools you use to build it. As Dwight D. Eisenhower said, "The plan is nothing; the planning is everything."

Essential items numbers 2, 3, 4, 5 and 8 capture the "essence" of the *Project Management workflow* in RUP: conceiving a new project; evaluating scope and risk; monitoring and controlling the project; planning for and evaluating each iteration and phase.

---

[5] "The goal is to develop a quality product, on time and on budget, that meets the stakeholders' real needs." — *Managing Software Requirements,* Dean Leffingwell & Don Widrig, Addison-Wesley Longman, January, 2000.

[6] Johnson Space Center Shuttle Software Group, "They Write the Right Stuff", Charles Fishman, *Fastcompany,* Issue 6, p. 95, December, 1996.

Risks — Identify and Mitigate Risks

An essential precept of RUP is to identify and attack the highest risk items early in the project. The risk list is intended to capture the perceived risks to the success of the project. It identifies, in decreasing order of priority, the events which could lead to a significant negative outcome.

Along with each risk, should be a plan for mitigating that risk. This serves as a focal point for planning project activities, and is the basis around which iterations are organized.

Issues — Assign and Track Issues

Continuous open communication with objective data derived directly from ongoing activities, and the evolving product configurations are important in any project. In RUP, this is done through regular status assessments, which provide the mechanism for addressing, communicating, and resolving management issues, technical issues, and project risks. In addition to identifying the issues, each should be assigned a due date, with a responsible person who is accountable for the resolution. This should be regularly tracked and updated as necessary.

These project snapshots provide the heartbeat for management attention. While the period may vary, the forcing function needs to capture the project history and resolve to remove any roadblocks or bottlenecks that restrict progress.

Business Case — Examine the Business Case

The Business Case provides the necessary information, from a business standpoint, to determine whether or not this project is worth investing in.

The main purpose of the Business Case is to develop an economic plan for realizing the project Vision. Once developed, the Business Case is used to make an accurate assessment of the return on investment (ROI) provided by the project. It provides the justification for the project and establishes its economic constraints. It provides information to the economic decision makers on the project's worth, and is used to determine whether the project should move ahead.

The description should not delve deeply into the specifics of the problem, but rather it should create a compelling argument why the product is needed. It must be brief, however, so that it is easy enough for all project team members to understand and remember. At critical milestones, the Business Case is re-examined to see if estimates of expected return and cost are still accurate, and whether the project should be continued

Architecture — Design a Component Architecture

In the Rational Unified Process, the architecture of a software system (at a given point) is 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 a software architecture. This is the “essence” of the *Analysis and Design workflow* in RUP: 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 define an architectural representation, a way of describing important aspects of an architecture. In the RUP, this description is captured in the Software Architecture Document, which presents the architecture in multiple views.

Each architectural view addresses some specific set of concerns, specific to stakeholders in the development process: end users, designers, managers, system engineers, maintainers, and so on. This serves as a communication medium between the architect and other project team members regarding architecturally significant decisions which have been made on the project.

Product — Incrementally Build and Test the Product

The “essence” of the *Implementation and Test workflows* in RUP is to incrementally code, build and test the components of the system, with 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. Throughout each iteration of the construction phase, components are integrated into executable, tested builds that evolve into the final product.

Key to this essential element is an integrated set of test activities that accompany the building of the product – as well as ongoing Configuration Management and review activities.

Evaluation — Regularly Assess Results

The Iteration Assessment captures the results of an iteration, the degree to which the evaluation criteria were met, the lessons learned and process changes to be implemented.

The Iteration Assessment is an essential artifact of the iterative approach. Depending on the scope and risk of the project and the nature of the iteration, it may range from a simple record of demonstration and 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."

Change Requests — Manage and Control Changes

The "essence" of the *Configuration and Change Management workflow* is to manage and control the scope of the project, as changes occur throughout the project lifecycle, while maintaining the goal of considering all stakeholder needs and meeting those, to whatever extent possible.

As soon as the first prototype is put before the users (and often even before that), changes *will* be requested. (One of those certainties of life!) In order to control those changes and effectively manage the scope of the project and expectations of the stakeholders, it is important that all changes to any development artifacts be proposed through Change Requests and managed with a consistent process.

Change Requests are used to document and track defects, enhancement requests and any other type of request for a change to the product. The benefit of Change Requests is that they provide a record of decisions, and, due to their assessment process, ensure that impacts of the potential change are understood by all project team members. The Change Requests are essential for managing the scope of the project, as well as assessing the impact of proposed changes.

User Support — Provide Assistance to the User

At a minimum, this should include a User's Guide, perhaps implemented through online help, and may also include an Installation Guide and Release Notes. Depending on the complexity of the product, training materials may also be needed, as well as a bill of materials along with any product packaging.

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, operating, and maintaining the product.

What about Requirements?

Some of you may look at my list of essentials and vehemently disagree with my choices. For example, where do "Requirements" fit into this picture? Wouldn't that be a better choice instead of "Vision"? Requirements are certainly essential, right? Good questions. I would say that's basically what the first essential is – and for those of you who prefer to have "Requirements" on the list, you can put it at this point on your own list! (Remember, each project team should come up with their own list. My list of ten essentials is only meant as a starting point for further discussion.)

In my experience, though, I sometimes will ask a project team (especially for an internal project), "What are your requirements?" and receive the response, "We don't really have any requirements." At first this surprised me (coming from a mil-aerospace background)! How could they not have any requirements? Then I found out that what they have in mind when they hear the word "requirements" is a set of externally imposed "shall" statements that they "must" conform to or have the project rejected – and they *really* don't have any of those! They may be involved in research and development where the requirements of the product evolve throughout the project.

Now, if I follow up that response with: Okay then, what is the vision for the product? Then, their eyes light up – we easily move through each of the questions in A through G above with no problem – and the requirements follow along thereafter. This is most often true in a collaborative environment, rather than a contractual project, where the requirements are "discovered" more than imposed by edict.

What about Test?

Undoubtedly, some of you keen sorts have also noticed that I didn't include "Test" as one of the "essentials" of RUP. Why is that? Unlike Business Modeling, which I truly believe is optional to an effective software development process, I in no way believe that testing is optional. I view test as an integrated set of activities that accompany the designing and building of the product – much the same way as ongoing Configuration Management and review activities – and, coincidentally, just as the IEEE 1074 software process standard does.

And, if you were very keen, then you noticed that testing is indeed included along with the building of the product (in essential number seven), with the testing results critically important for Verification & Evaluation (essential number eight). The essence of the Rational Unified Process' iterative approach to software development, is to continually build, test and evaluate executable versions of the product in order to flush out the problems and resolve the risks and issues as early as possible.

Summary: Applying the Ten Essentials

So, how can discovery of the "Ten Essentials of RUP" make a difference in my life?

For Very Small Projects

Well, first of all, if someone like my friend, Randy, had in fact come by to ask for advice on a how he could use RUP for building a simple product by himself or with a very small inexperienced team, just learning about process, I now have my own list of "ten essentials" that I can share with him, rather than overwhelm him with the full power of RUP and the Rational Suites of tools!

In fact, these ten essentials can be accomplished with no specialized tool support! A project notebook with one section devoted to each of the ten essentials, is actually a very good starting point for managing the project. And I have found Post-It Notes invaluable when initially managing, prioritizing and tracking change requests for my own personal application of the RUP!

For Growing Projects

Of course, these simple means of applying the ten essentials soon become unmanageable as a project's size and complexity grows, so those of you looking for potential customers for Rational's tools will find ripe prospects in those applying the "Ten Essentials of RUP".

And, of course, they'll eventually want the full RUP, too, as well as support of the Rational tools (if you do your job right!), once they decide to go deeper than just the surface level in each area. I would encourage you, though, to lead with the "Ten Essentials of RUP", alongside our "Best Practices", rather than leading with the tools.

For Mature Project Teams

For more mature project teams, the "ten essentials" can help provide you with a quick method for assessing the balance of the key elements of their process and identify areas where improvement is most beneficial.

In all projects, it is important to explore what will happen if any of these essentials is ignored – what will fail if you do not have it. For example:

- No vision? You may lose track of where you are going and may be easily distracted on detours.
- No plan? You will not be able to track progress.
- No risk list? You may be focusing on the wrong issues now and will explode on an unsuspecting mine 5 months from now.
- No issues list? Without accountability, these are the roadblocks to success.
- No business case? You risk losing time and money on the project; it may be cancelled or go bankrupt.

- No architecture? You may be unable to handle communication, synchronization, and data access issues as they arise; problems with scaling and performance.
- No product (prototype)? Get started writing code; just accumulating paperwork won't get you very far; get something working in front of the customers.
- No evaluation? Don't keep your head in the sand. It is important to face the truth. How close are you really to your deadline? To your goals in quality or budget?
- No change requests? How do you keep track of these?
- No user support? What happens when a user has a question or can't figure out how to use the product? How easy is it to get help?

So, there you have it!

Why "ten" essentials? No particular reason, except I didn't want *more* than ten (can't count that high!). I would have been happy to have "nine Essentials" or some smaller number, but we didn't seem to be able to cut any of them out. Guess that qualifies them as essential, right? At least to me! I encourage you to use this as a starting point in your project group: What are your "10 essentials"?

If you're like me, you may want to come up with a little memory hook or acronym to help you remember your list of "Ten Essentials". For example, to get it down to 4 syllables, I use V-PRI-BAPE-CU—and, of course, since I kept the number of essentials to 10, I can slyly use my fingers for counting them without having to refer to the list on my PalmPilot! (Now you know my secret!)

Rational®
the software development company

Dual Headquarters:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com
International Locations: www.rational.com/worldwide