A whitepaper from IBM's Rational software division
July 2005

Rational. software

Managing Subcontractors with Rational Unified Process

Moacyr Cardoso de Mello Filho
review 2.0

Contents

Abstract

In many commercial projects, we must combine subcontracting with normal software development to provide a complete management process, acceptable to all stakeholders involved in the project. This paper explores the subcontracting strategies available to organizations adopting the IBM®, Rational Unified Process®, as part of their software development strategy.

1. Introduction

We can only propely solve the demand for service contracting solutions when we introduce a well-defined development process. In this paper we will approach subcontracting in a well-defined project context, particularly one in which RUP is applied.

1.1 Purpose

This article serves to describe service subcontracting strategies in software projects using RUP.

1.2 Scope

We will limit our approach to subcontracting in the ambit of a well-defined process. However, we do not assume both parties are necessarily using RUP. We accept that the subcontracted party may use some other process, different from RUP, or even RUP itself, but with a unique customization, characterizing the need for mapping.

1.3 Acronyms and Abbreviations

PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
SA-CMM	Software Acquisition Capability Maturity Model
SEI/CMU	Software Engineering Institute/Carnegie Mellon University
UML	Unified Modeling Language

1.4 References

- *PMBOK – Project Management Body of Knowledge,* Project Management Institute
- *The Rational Unified Process An Introduction*, Philippe Kruchten, Addison-Wesley Longman, 1999
- *Software Project Management – A Unified Approach,* Walker Royce, Addison-Wesley Longman, 1999
- *Rational Unified Process, 2003,* Rational Software, 2003
- *Reaching CMM Levels 2 and 3 with the Rational Unified Process,* Rational Software Whitepaper, Rational Unified Process Whitepaper Page, 2001

1.5 Overview

This paper begins with a general discussion regarding subcontracting; it analyzes the subcontracting structure that Rational's process resources have in order to deal with the problem, and defines two important concepts: equivalent artifact and subcontracting scenario. It then deals with the processes, contract types, and management models necessary for subcontracting. Finally, it presents the development cases with subcontracting scenarios and a simplified application example.

2. Subcontracting in a Software Project

When a software project involves contracting a service, a dilemma always comes to mind: "Do it in-house or purchase it?" As "everything is software…," it would seem that the option of doing it ourselves would be more manageable and cause less trouble – it is, afterall, in the house! This illusion is reinforced by closed project contracting experiences which, by and large, were not successful and over which we had little control.

However, we know perfectly well that a project, even when small, benefits greatly from the capacity of purchasing market services. We frequently cannot, due to the lack of knowledge or interest in a certain technology, carry out a project without contracting external services. Thus, we must be trained to hire outsourcers. In this text, we will use the word "subcontracting" to define the cases in which we seek services in the market to complement our project team's capacities and complete, or build, the best product for our clients.

When we analyze the reasons for failures in subcontracting, we soon discover that the lack of a well-defined process to guide our actions is the main reason for so many disappointments. Purchasing services, despite being a rather routine task in a project manager's life, is a high-risk endeavor and, usually, an empirical activity. Nonetheless, we purchase many things during a project. When we purchase hardware, or some material, we are performing a search procedure for certain characteristics we will evaluate during the purchase. We know how to do that! Further, most organizations have a well-defined procedure to make these acquisitions. This procedure, or acquisition process, is known in classical engineering as the procurement process. (Note: Procurement Management, one of the nine PMI Knowledge Areas, see the reference). This process' activities, when executed, provide all necessary supplies (equipment, materials, services) to deploy a project.

Our approach will be to treat subcontracting as a special supply case in which the purchased good is basically a service. We say "basically" because, strictly, we may have a set of services and products — for example, software component libraries — but the service characteristic will always be dominant. Therefore:

Subcontracting	*Process in charge of obtaining, within the defined term, cost, and quality parameters, the services necessary to deploy a software project.*

We must deal with subcontracting, since it adds complexity due to third-party relationships, as an open system. It must have certain "plasticity" and flexibility; that is, *adapting* as the project goes on. Rigidity and immutability prevent us from capturing the dynamics of several organizations involved in a project. We must consider the cause and effect relationships with feedback in open systems, in which control depends on information and works via retro-feeding, seeking balance among the project's parts and objectives. In fact, imagining such control is by no means new, but how often do we catch ourselves thinking otherwise, imagining third-parties will strictly comply with and vendors our plans?

Obviously, the management model must be adequate for these process characteristics, and this is one main cause for the lack of success when dealing with third-party contracting during a project.

Due to the cost of administering new subcontracting for each new project, it might be advantageous to definitively delegate certain tasks or product manufacturing parts to third parties. In this case, you will confront the difficulties of selecting the vendor only once, and the contracting will be a semi-permanent arrangement. These ideas came about some time ago and seemed to be a "new trend" aiming at reducing costs and allegedly adjusting the company to its specific business. This trend was called *outsourcing*. However, one of the biggest gains in the value chain is the capacity to use the market's power and knowledge to our benefit. The "On-Demand Era" we are entering does just that: it redraws the value chain, assimilating several companies' specializations, knowledge, and productive capacities to a certain business. This leads us to a more extreme aspect of the problem, i.e., permanently managing someone else's service. We can define:

Outsourcing	*Planned subcontracting process characterized when a company activity is permanently transferred for another company to develop.*
	(Note: This outsourcing definition is valid both for the project period and for the system's later operation, "ongoing operations" in PMI's nomenclature; however, in this paper, our focus is on software development projects).

In the current software development market stage, we notice a lack of planning, with focus placed merely on cost reductions and an absolutely deficient management. This potentializes the risks that are inherent to the process, such as excessive dependency on the vendor, distancing from other vendors, technological limitations, etc.

Just as normal project subcontracting benefits from a well-defined process, outsourcing only stands to gain if we establish a process to manage the relationship with vendors. Additionally, changing to a well-defined context will allow us to focus on efficiency and competitiveness, promoting vendor development and making a truly On Demand operation viable.

3. Subcontracting in the RUP Scope

The Rational Unified Process allows subcontracting as a normal project development activity. Since the development organization is supposedly using the Rational process, we can assume that the subcontracter may use all of the concepts and the several procedures available in RUP. The process, however, doesn't elaborate how we do this yet.in addition, only it recommends that we define an artifact called *Subcontractor Management Plan*, which should guide the management actions, presenting the subcontracting strategy. This plan must be coherent with the organization's business objectives and with the objectives of the project itself; thus, both parties must agree with the project's Business Case and we must mention the plan in this last artifact.

When the project works under contract, RUP also foresees, in addition to the Subcontractor Management Plan, another three documents: the Request for Proposal (RFP)*, the vendor's response, and the contract itself. However, there is no special consideration for subcontracting issues, nor are there templates for the documents in question.

** We know RFPs that require time and cost for systems that have not been conceived, specified, and do not exist in level of reasonable abstraction. The problem is that an RFP requires a true requirement specification before being launched to the market. The doubt of whether to make the requirement specification before or after the RFP is false, i.e., as if the entire specification could be contracted as part of the project. On the other hand, it is certain we can contract some specification or detailing for this; any contracting must specify the services up to the level of the pricing and quality and scope definition; otherwise, it becomes a blind purchase.*

The problem of defining the specific subcontracting form is less dependent on the choice of artifacts; rather, it relies much more on the decision regarding development management. Different organizations may adopt incompatible lifecycle models, causing huge management and control difficulties. This is the case of the waterfall and iterative models, which have totally diverse planning and management philosophies. Thus, how the contracting organization would behave toward the contracted ones that didn't use the same process remains open.

RUP has an iterative and incremental lifecycle. Both parties must respect this characteristic in order for all to achieve the planning and managerial control assertiveness benefits. Our challenge is to identify subcontracting scenarios that allow organizations with different processes to be able to contract among each other as long as at least one of them uses RUP.

To undertake this challenge, we need to create two simple concepts: equivalent artifact and subcontracting scenario.

3.1 The Equivalent Artifact Concept

All contracted organizations somehow execute a development process that, no matter how informal, must have project documents for diverse functions, such as planning, evaluation, supervision, documentation, design, release notes, manuals, etc.

Each company, contracting or contracted, always has a set of project documentation artifacts, in addition to an endless amount of internal control, registration, and information transmission documents. The first difficulty to overcome is reducing communication confusion, defining a coherent set of well-known artifacts that have an adequate production process for the contracting.

To achieve homogeneity among the contracting and contracted parties' processes, we need to impose that the needed information, from the RUP point of view, exists, and tha we can generate, update, and communicate it. The material form this information takes on is negotiable and less relevant than its existence. Therefore, to impose the need for the information contained in RUP artifacts, we must simply determine which contracted party artifacts satisfy the content. If one is missing, or is incomplete, the contracting party must request, via an agreement, the need for the information to be complemented. Thus:

Equivalent Artifact	*One or more contracted company artifacts that fulfill the content requirements for one or more of the RUP artifacts that the contracting company uses*

To operationalize equivalent artifacts we must simply, during the vendor selection activity, map the contracted party's documents and models and compare them to the contracting party's artifacts. For example:

Discipline	**Contracting Party**	**Contracted Party**
Requirements	Vision	Vision Statement
	Use Case Model	Product Specification
	Supplementary Specifications	
	Software Requirement Specification	
Project Management	Iteration Assessment	Post Mortem

3.2 The Subcontracting Scenario Concept

We can define typical scenarios to understand the multiple subcontracting strategies based on the Rational process. Each scenario has a remarkable characteristic that predominates in the set, and many other secondary characteristics that result from the first one. A real project case can use more than one scenario to represent its problem.

The main characteristic of a scenario is to define what point, in the development process, the contracting organization wants to reach. In other words, which software models the contracting party wants to produce internally. This subset of models can give it internal communication security, control over the problem, and is probably already part of the developer and manage rexperience. The immediate implication in terms of RUP is to ask which artifacts the contracting party will it execute and which ones will purchase from third parties. This can be said in a different manner: based on which model (Artifact) does the contracting party want to contract service execution?

But the scenario isn't characterized solely by the choice of artifacts; it also implies a structure for the subcontracting project. You need to decide if external personnel will have access to the project's stakeholders, whether the development and test environment will be internal, if homologation will involve internal clients, and so on and so forth. There are implications resulting from the type of model you choose and want to contract with; the transferred risks can return in the form of other short- or long-term risks. All of these considerations constitute the subcontracting strategy that is implicit in selecting a scenario. Therefore:

Subcontracting Scenario	*The artifact and project structure set that composes a certain subcontracting or outsourcing strategy.*

In the scenarios we will describe, the basic assumption is that the contracting organization adopts RUP to its fullest extent. That it applies, internally, the iterative lifecycle model planning and management but wants to subcontract parts of its projects from organizations that don't necessarily adopt RUP, or perhaps adopt it but customize it for their own needs. The contracted organization must have a reasonably well-defined process, whatever it may be; however; it should be flexible enough to attend to the contracting, execution, auditing, and documentation requirements the contracting party will impose.

There are five or six scenarios of true interest when analyzing subcontracting:

3.3 Scenario 1: Subcontracting Based on a Use Case Model

The contracting party defines the Use Case model and contracts development from there.

3.4 Scenario 2: Subcontracting Based on an Analysis Model

The contracting party elaborates the Use Case and Analysis models (Static Class Diagram) and contracts development from there.

3.5 Scenario 3: Subcontracting Based on a Design Model

The contracting party elaborates the Use Case, Analysis (optionally), and Design (Static Class Diagram) models, checking the architecture and validating the main project risks and architectural issues. It contracts development from there.

3.6 Scenario 4: Programming Subcontracting

The contracting party elaborates the Use Case, Analysis (optionally), and Design models. It remains in charge of developing these models. However, the programming part is outsourced in the "software factor" molds. In this scenario, the code the contracted party generates is integrated and the contracting party performs the homologation tests.

3.7 Scenario 5: Test Subcontracting

The contracting party elaborates the Use Case, Analysis (optionally), and Design models.It implements the code, but the tests are outsourced in the "test bureau" molds. In this scenario, the bureau is in charge of planning, administrating, and performing the tests.

3.8 Scenario 6: Subcontracting Programming and Tests

The contracting party elaborates the Use Case, Analysis (optionally), Design, and Test models. However, the contracted party is in charge of program implementation and test application. (This is a hybrid mode mixing Scenarios 4 and 5, however, with the difference that the contracting party retains the domain and the knowledge in tests, and only outsources its application).

3.9 Hybrid, Trivial, and Canonic Scenarios

The actual subcontracting cases may be considered variations of these six scenarios, hybridized or not, composing a specific subcontracting case. This specific case must be described in the subcontracted party management plan and referenced in the project's Business Case and Development Case.

There are special, secondary interest cases that may satisfy highly particular conditions. For example, there is the scenario in which a contracting party, because it doesn't have the necessary knowledge about a certain business domain, orders a Business Analysis model for the contracted party, specialized in the business area. In this case, the contracted party may request the application design model itself from as an initial development for the product in question (product concept). The interest here is nearly academic and does not correspond to most common commercial developments.

The trivial scenario, in which the contracting party requests the entire development from the contracted party, as if it were a client, characterizes precisely the client/stakeholder roles RUP itself deals with, and it is summarized in the client-vendor relationship in which the contracting party is the client. Obviously, this scenario will not be dealt with here.

The basic hypothesis, implicit in the six scenarios, is that the process the contracted party uses may not be RUP, at least in its totality, thus, the need for using the equivalent artifact concept. But would it not be possible to understand the six scenarios in which both sides, contracted and contracting, use the exact same RUP configuration?

Yes. There may certainly be a case in which both organizations use the same RUP configuration, exactly out-of-the-box, with no customization. This situation is an exception, as RUP itself teaches and stimulates us to customize the configuration in a process implementation. Even for two organizations that use RUP, the artifact mapping idea is necessary, although in many cases it is facilitated.

When both organizations used the same process framework, the six scenarios that were presented would be treated with the conceptual arsenal that exists in RUP; the management forms, the roles, and exchanged artifact content would be well-defined and the process dynamics would follow the known iterative cycle norms. Nonetheless, we would define, in the subcontracted management plan, each part's artifacts and responsibilities based on one or more subcontracting scenarios (referred to in the Business Case and Development Case). When this subcontracting situation occurs, with any scenario, we can call it canonic; the process itself is defined, and we only need to solve artifact details.

For references to the canonic scenarios, we recommend the Rational whitepaper about the CMM model mapping to the Rational best practices that briefly deals with subcontracting (*Reaching CMM Levels 2 and 3 with Rational Unified Process*, Rational Software Whitepaper, Rational Unified Process Whitepaper Page, 2001).

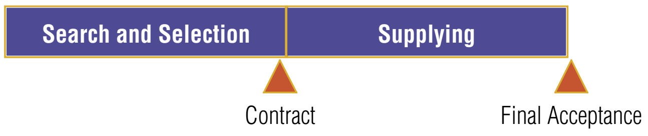
For references regarding the trivial scenario, we recommend reading RUP itself (*Rational Unified Process 2003*, Rational Software, 2003).

4. Processes Connected to Subcontracting

You may or may not need subcontracting in a certain project. When you do, specific activities will compose a workflow that is characteristic of this process. Such a workflow will generate information about registration and handling artifacts. Before listing them, we will identify subcontracting macro processes:

- *Subcontracting strategy*
- *Subcontracting content specification*
- *Vendor evaluation and development*
- *Selection and contracting*
- *Contract and delivery management*

There are clearly two distinct phases: one in which you define the subcontracting and its vendors, and another, in which the vendor performs and follows up on the service. We can define, following a classical supply model, the following phases:



Signing the contract signifies the end of the first phase, when a new moment opens for the project. In the following period, we work the supply phase with the subcontracted party; this phase only ends with final service or product acceptance.

4.1 Artifact Definition

For each of the identified processes, we will define one or more information registration forms, and later the activities, the roles, and those responsible for generating them.

Subcontracting strategy	- Subcontractor Management Plan - Business Case
Subcontracting content specification	- Request for Proposal (RFP) Requirement workflow and Analysis & Design artifacts, according to the subcontracting scenario.
Vendor evaluation and development	- Decision Criteria Legal, fiscal, economical and financial, and technical qualification proof documents.
Selection and contracting	- Contract - Response of Request for Proposal
Contract and delivery management	- Review Record - Status Assessment Other Project Manager workflow artifacts according to the subcontracting scenario

Defining these artifacts does not exclude the need for other documents that are required or may facilitate the subcontracting process in specific situations.

5. Contracts and Contractual Mode

One of the most relevant issues in subcontracting is the type of contract you select that will govern the relationship with the subcontracted party. Our scope does not include discussing the contract itself, but we will cover the contractual mode that characterizes how services are purchased. Insofar as the contract is concerned, we recommend the project manager always check the document structure, analyzing it if the contract's object and scope are well defined, and if they attend to the project's expectations. All contracts always have a similar structure, that may include some or all of the following:

- *The parties*
- *Object and scope*
- *Contractual mode*
- *Term*
- *Value and form of payment*
- *Price recalculation*
- *Required guarantees*
- *Responsibilities*
- *Rights and obligations*
- *Object changes*
- *Schedule changes*
- *Project errors*
- *Fines and sanctions*
- *Rescission*
- *Jurisdiction*

You should check whether you properly formulated the objectives and needs based on the specification and whether there is an appropriate detailing level; the adequate items for this are the object and the scope.

Object	*The formal description of what you intend to achieve with the project. This is normally measurable and definable through numerical parameters (time, cost, and performance).*
Scope	*Project reach description. This defines what the project will and will not do. It represents the limit between the project and the rest of the organization, identifying what will be significantly changed by the venture.*

We need to make a distinction between project scope and system scope. We implement a system is implemented in a project, but their scopes are not the same thing. The system's scope. which we should find in the Software Requirements Specification (SRS) RUP artifact. There, we find what the system must be and/or the resources it must have.

On the other hand, the project scope delimits the result of the work that will be delivered (the deliverables) and it is found in the Software Development Plan (SDP). A simple example would be calculator software that performed the four operations. The system scope would be - a four-operation calculator. The project scope could be - the calculator software itself, online help, a printed manual, training material, two user groups trained, and the installation at four client sites. You must define the two scopes for contracting to be adequate.

From the management point of view, however, the form of the legal contract is not as important as the process to achieve the contractual relationship between the buyer and the vendor, i.e., how we purchse the service. The item to highlight is the contractual mode, which is fundamentally important for proper project performance. We will deal with this below.

5.1 Contractual Mode

All projects are subject to risks and uncertainties: depending on how we organize the subcontracting we may or may not improve our projects' risk profiles. The different contractual modes can help us create a safer project, one that is more adequate for the established objective. Contractual mode is how the subcontracted parties are hired and paid for the services they render. They include:

5.1.1 Lump Sum

In this mode, the contracted party stipulates a single, fixed, global price for the entire service. This is defined by the scope that mandates you determine the nature and the amount of work to be done, usually through project specification artifacts.

Although it is fixed, there may be readjustments due to economic factors, and payment may be divided into installments, as the stages are carried out or periodically; this does not decharacterize the global price contract.

5.1.2 Unit Price

In this mode, you hire with a payment that is pre-defined per unit for each project element. You define a list of elements to be worked on, and attribute a fixed price to each of them (e.g.: screens, stored procedures, code lines). It is important to determine all of the elements to be contracted and to estimate the amounts.

You calculate payment periodically or when a certain volume is reached.

5.1.3 Hourly Fee

In this mode, you paythe contracted party through a cost table established per hour based on the type of professional who is hired. You make payment by multiplying the number of hours by the hourly fee. The nature of the work has secondary importance. You use this when you are uncertain about the project's cost elements and the number of hours for the work.

In this mode, the contracted party has no administration obligations and control, or such obligations and control are minimal, and the market controls fee rates. This is a special variation of the unit price in which the unit is the hour.

5.1.4 Administration – Cost Plus Fee

In this mode, the contracted party performs the items the project scope defines and its costs are reimbursed, in addition to receiving a fixed percentage rate for expenses. In this case, the contracted party's payment increases when the service cost goes up, which may promote inefficiency.

In this mode, there is a natural increase of managerial control elements by the contracting party. However, this is the most common mode when the service nature and amounts are unknown.

5.1.5 Cost Plus Fixed Fee

This mode is a variation of the Administration mode; however, there is no fixed percentage fee, rather a fixed amount. This mode does not promote inefficiency.

5.1.6 Cost Plus Incentive Fee

This mode is a variation of the Administration mode, however, with an incentive instrument: when the contracted party performs the service with an above-expected performance (with savings), the percentage rate is increased. When the contracted party performs the service with a below-expected performance (spending more or delaying delivery), the percentage rate is reduced.

5.1.7 Guaranteed Maximum Price

This mode is a variation of the Administration mode, whether the percentage rate or fixed payment, to which you ad a maximum expense limit. The contracted party has the obligation of not surpassing the maximum limit. Above such limit, the contracted party must bear all expenses.

5.1.8 Target-Price

This mode is a variation of the Administration mode, more specifically a variation of the Guaranteed Maximum Price mode, in which you establish a target price and both parties agree that after the contracted party completes the work, if the cost is lower (i.e. there are savings) than expected, both parties share difference between the actual cost and the target-price. If, however, the contracted party surpasses the target price, the contracting and contracted parties will also share the overhead cost.

5.1.9 Turn-Key

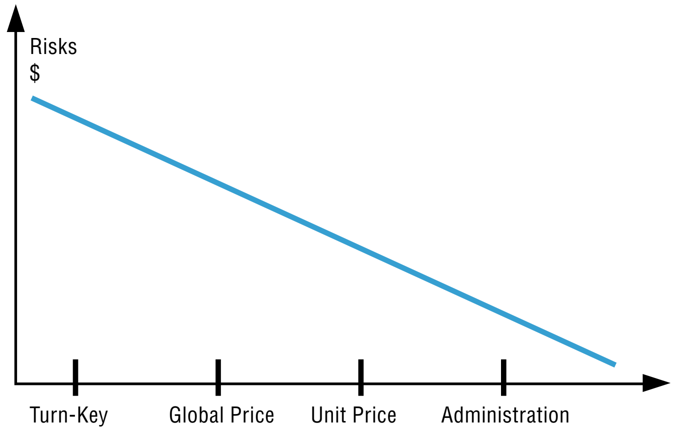
In this mode, the contracted party commits to perform the work for the entire project scope, delivering it completed and in operational conditions. The contracting party, as it does not have the project's technology or know-how, gives the contracted party the entire responsibility for performing the development cycle, from survey, project, and implementation. The contracting party, however, has performance information and functional needs with which it will evaluate the contracted party's work.

This contract mode transfers all of the responsibility to the contracted party and, in general, is paid for at lump-sum prices. The contracted party incurs the economic and execution risks. In special cases, you may use the cost system for this mode. In this case, the contracting party incurs the economic risks, while the contracted party the execution risks.

To select a mode, we must first consider the contracting party's organization strategy. Based on service specifications, we then consider, whether or not we know it well, the nature and amount of work we want to hire. We check if a certain activity is well defined and whether we have a good estimation of its duration or result. In sum:

The nature and amount of all work is well known.	We use Lump Sum
The nature of the work is known, but not its amount.	We use Unit Price or Hourly Fee
Neither the nature nor the amount of work can be characterized well, but the project has been defined.	We use Administration
The contracted party has know-how and can take the project on through implementation, delivering it completed.	We use Turn-Key (paid, usually, in a lump sum)

To assist in mode selection, we examine the risk and expense profile the project will confront. The vendor is aware of the higher risk/higher price ratio. This is due to the margins established for uncertainties and for amount, labor, and performance variations. By and large, the following graph is true:

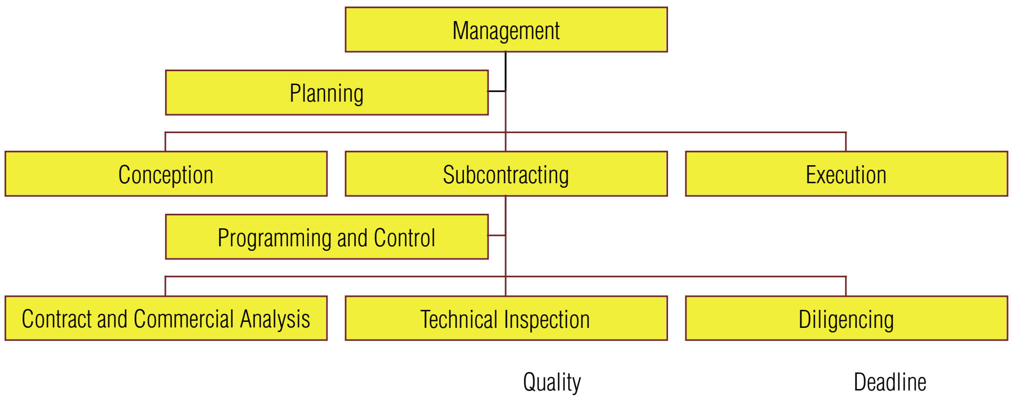


Therefore, the higher the risk, the higher the price, and the modes that transfer all risk to the contracted party tend to cost more. Obviously, in a real case, the best solution is to seek a hybrid solution, elaborated specifically for the project at hand.

6. Typical Management Model

A management structure manages the entire software project. No matter how small the project, even if limited to a single project manager, there is a managerial function division that can be defined as planning, conception, execution, and supplies; the latter, in a software project, understood as service subcontracting.

Based on the subcontracting function description, we can elaborate an organizational structure that is typical of this sector, as the following figure shows.



Subcontracting, however, must be seen as a management function, subordinate to planning and to supply nature.

Under the subcontracting managerial function, we identify the programming and control functions, in charge of elaborating a schedule and synchronizing it according to the rest of the project; the contracting and commercial analysis function, in charge of checking the contract and of analyzing the commercial proposals that are received (don't confuse this with the procurement and legal departments that are usually structures separate from project management and which may assist in the contract and commercial analysis manager's work).

Finally, we identify the technical inspection and diligence functions, which we will deal with later on, the first concerned with quality, while the latter with deadlines. Naturally, depending on the project's level of formality, depending on system size, and on subcontracting diversity, this structure may have more or fewer people individually in charge of each managerial function.

Note that the graph shows managerial functions, not necessarily organizational areas, and especially not individual. However, in a major project, we can have this entire organizational structure with several people in each position. In most cases, however, project management is performed by one person sharing functions with one or two other people. It is only in extreme cases that a single project manager performs – or tries to perform –all functions presented.

We define the technical inspection and diligencing managerial functions better below:

Technical Inspection	*This function is in charge of checking purchased artifact or service quality. It is a managerial, however specialized, activity that uses revisers who are available in the development organization. In a few organizations, this takes on a permanent character under acronyms, such as SQA, QA, etc. (In these cases, other internal responsibilities are added, which will not be discussed herein; its location in the managerial function hierarchy may vary according to the adopted doctrine).*

Diligencing	*This function is in charge of controlling procurement process progress so it will attend to the project's deadline and requirement contractual conditions. It performs together with the contracted party, particularly synchronizing the project's schedule stages with its iterative cycles.*

RUP provides us with several forms of technical inspection deploying, the most common of which is the interpair revision. However, for quality management of a contracted supply, we need to establish balance in the inspection procedure. If the manager intends to check all artifacts in their entirety, in all techniques, he/she will expend a huge volume of man-hours in a task, and this may become a project bottleneck.

A reasonable procedure is to clearly define the contracted party as the one in charge of service quality and perform more selectively. Always check if the artifacts comply with contractual terms and the definitions established in specification artifacts or guidelines. Apply a sampling criterion for technical artifacts using RUP checklists. Separate critical artifacts, such as architecture definitions, for example, and examine them more carefully.

This sampling approach may be possible depending the risk, on the project's formalism, and the artifact's nature (e.g., it is convenient to extensively examine a design model, but not the code itself, which can be examined through sampling).

Any of the presented subcontracting scenarios requires some type of diligencing. This managerial function has frequently been overlooked in software projects and is the cause of the most disagreeable surprises. Particularly when we hear that a contract was signed with the expression "project closed," the immediate psychological feeling is one of relief and of work load transference, but this is pure illusion. We are usually just postponing problems, which will appear at the most inconvenient time, i.e., during integration.

Traditional engineering, trained in supply and subcontracting problems, created the figure of the forwarding agent; that is, the manager or assistant manager who will check not service content, but schedule execution by the contracted party. The purpose is to predict problems and, if possible, anticipate deliveries — and make sure project iteration synchronism is respected. He or she must establish his function clearly in a contract, as it assumes the forwarding agent will visit the vendor for in loco verifications.

6.1 Role Definition

Two roles must be part of the managerial structure: the inspector and the forwarding agent. The first is concerned with the quality of what is being produced and purchased and, for this, the inspector establishes an inspection schedule that uses specialized technical revisers, according to the object to be checked. RUP is mandatory to provide elements for each technical revision artifact or activity.

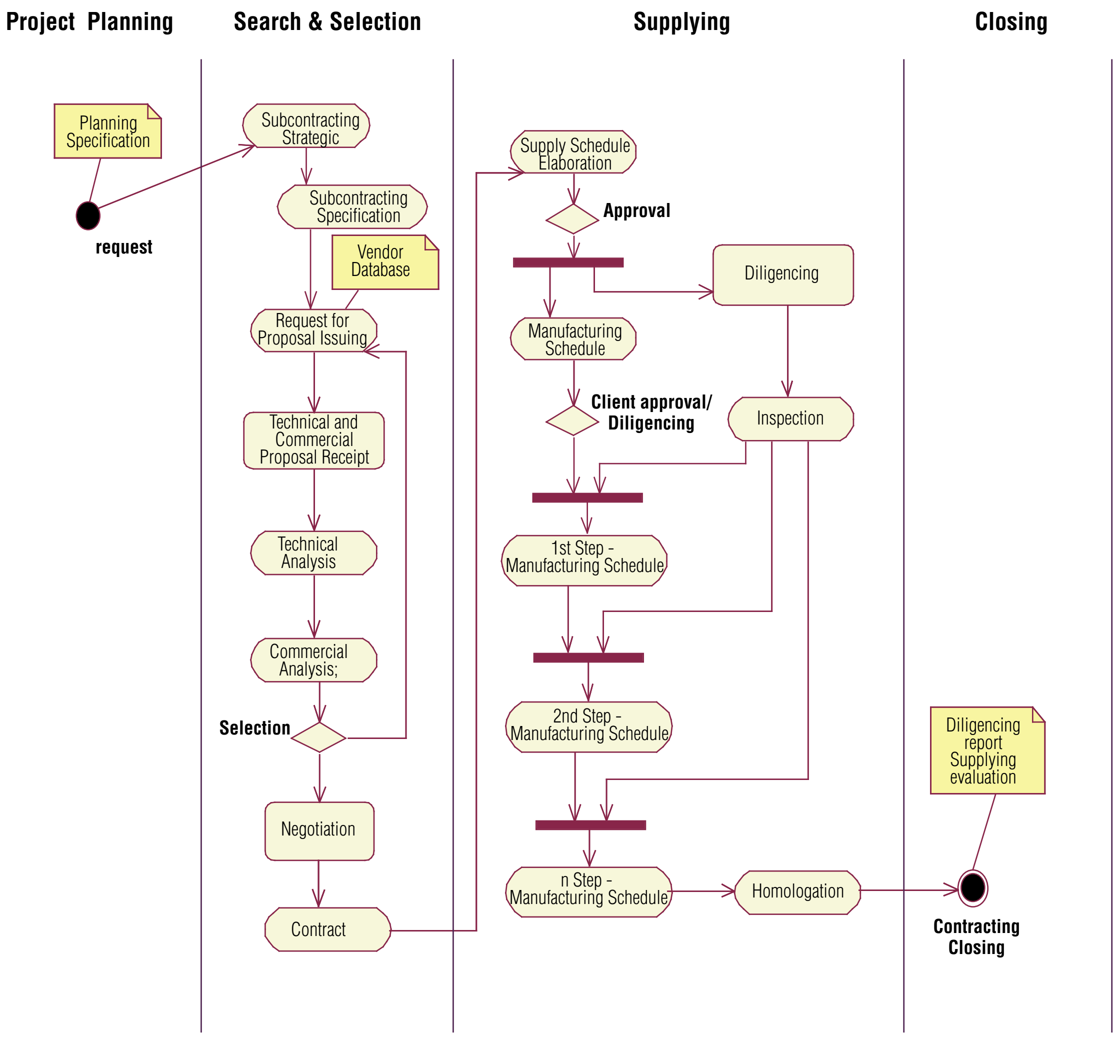
The second role, the forwarding agent, is concerned with supply deadlines and goals. The forwarding agent checks the manufacturing process and the progress of whatever the contracting party ordered. This person works at the contracted party's site, visiting it frequently or infrequently, and using subcontracting documents and contracts. The forwarding agent will seek maximum efficiency and promote better synchronization between contracting and contracted party schedules.

Subcontracting	- Project Manager
Programming and Control	- Project Manager
Contract and Commercial Analysis	- Purchaser Procurement and Legal Department
Technical Inspection	- Inspector SQA and specialized technical revisers
Diligence	- Forwarding agent

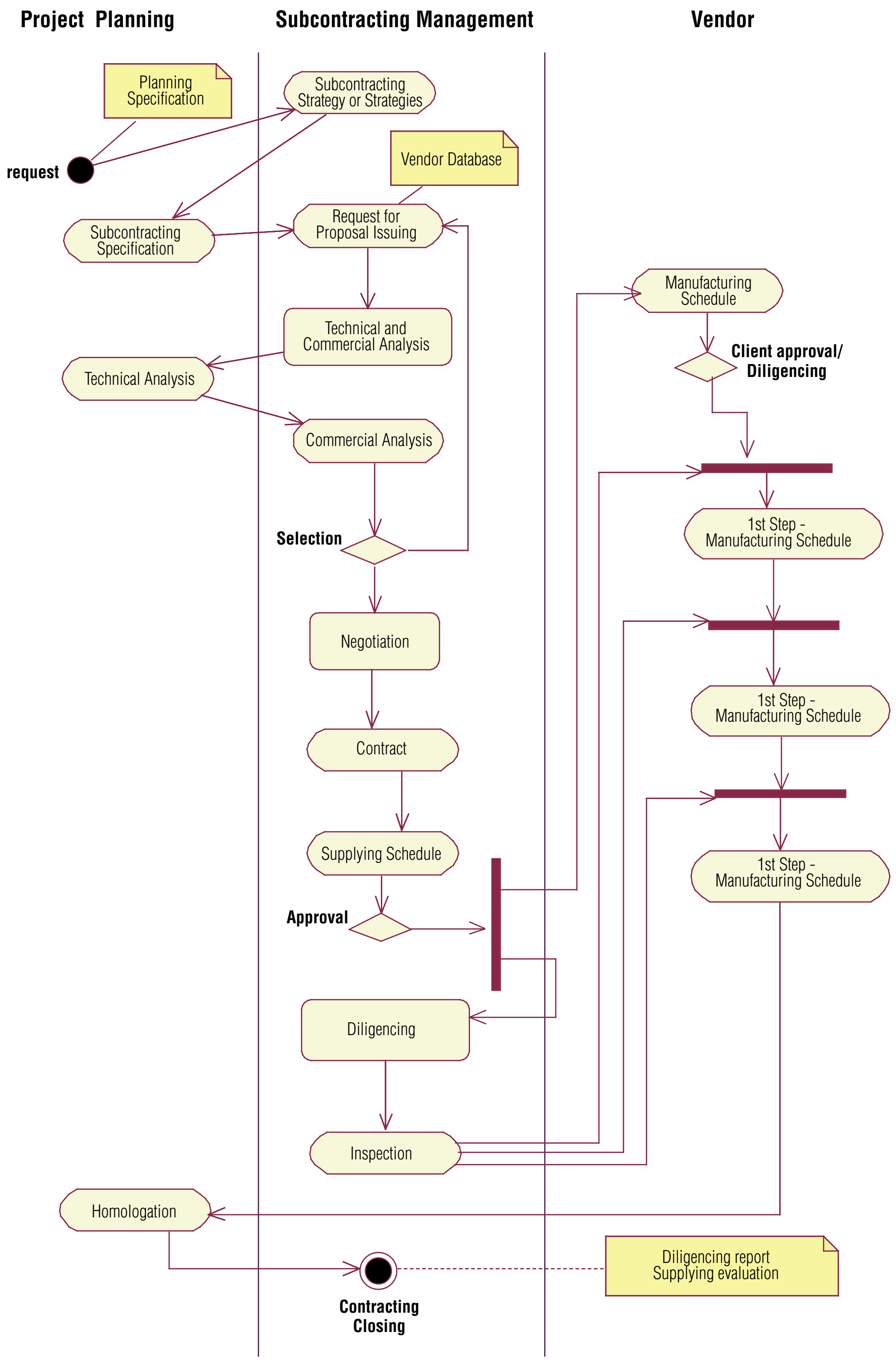
7. The Subcontracting Cycle

The subcontracting cycle refers to the activities performed during the project to obtain service subcontracting and to receive and accept the supply. We can observe this cycle from two different points of view: in time, viewing the Search & Selection and Supplying stages; or per responsibilities, viewing the departments or areas involved in the process.

The following figure shows the temporal view, from planning to closure.


Project Planning
Search & Selection
Supplying
Closing
Planning Specification
request
Subcontracting Strategic
Subcontracting Specification
Vendor Database
Request for Proposal Issuing
Technical and Commercial Proposal Receipt
Technical Analysis
Commercial Analysis;
Selection
Negotiation
Contract
Supply Schedule Elaboration
Approval
Diligencing
Manufacturing Schedule
Client approval/ Diligencing
Inspection
1st Step - Manufacturing Schedule
2nd Step - Manufacturing Schedule
n Step - Manufacturing Schedule
Homologation
Diligencing report Supplying evaluation
Contracting Closing

This video shows the organizational view, project team, subcontracting management, and vendor responsibilities.


Project Planning
Subcontracting Management
Vendor
Planning Specification
request
Subcontracting Strategy or Strategies
Vendor Database
Subcontracting Specification
Request for Proposal Issuing
Manufacturing Schedule
Client approval/ Diligencing
Technical and Commercial Analysis
Technical Analysis
Commercial Analysis
Selection
1st Step - Manufacturing Schedule
Negotiation
1st Step - Manufacturing Schedule
Contract
Supplying Schedule
1st Step - Manufacturing Schedule
Approval
Diligencing
Inspection
Homologation
Contracting Closing
Diligencing report Supplying evaluation

The supply, in the temporal view, or in the case of the vendor's scope, in the organizational view, can be performed once or repeated as often as necessary. For example, when ordering several software components. Homologation would also be repeated in this case. This depends enormously on the situation. The diagrams illustrate the overall subcontracting cycle format.

7.1 Activity Definition

The subcontracting cycle activities are added to the activities RUP defines to specify Requirements, Analysis & Design, and Testing. Therefore, for now, we won't worry about specification genesis; rather, admitting they exist (as well as project plans, development case, etc.), we will look specifically at what interests us: subcontracting activities. They include:

- *Subcontracting strategy*
- *Subcontracting specification*
- *Request for Proposal issuing*
- *Proposal receipt*
- *Technical analysis*
- *Commercial analysis*
- *Selection*
- *Negotiation*
- *Contract elaboration*
- *Supply schedule elaboration*
- *Diligencing*
- *Inspection*
- *Homologation*
- *Contract closing*

As we stated, based on normal project specifications, management defines the subcontracting scenario by elaborating on a subcontract management plan. Then, the several project requests will begin the subcontracting cycle for each specific element. This first activity is defining a subcontracting strategy.

Then, management elaborates on the specification of whatever will be contracted, together with budget evaluations, a basic schedule, and decision criteria. Requests for proposals are then sent to the contracted party. Thus far, we've established the following important elements:

- *Contracting object*
- *Contracting scope*
- *contractual mode*
- *eEstimated budget*
- *Basic schedule*
- *Decision criteria*

No matter how these elements may change, defining them before asking for proposals is crucial from the project, refinement point of view. No process is born perfect; it is refined with successive iterations.

So far, the artifacts we have produced are: Subcontractor Management Plan, Business Case, Request for Proposal, Decision Criteria, and any other elements that may be necessary for the process of elucidating the invitation or presenting the subcontracting proposal (slides, etc.).

At a second moment, proposals are received and qualified; then the technical and commercial activities are performed and, finally, the choice is made. We emphasize the project team must be in charge of technical analysis and that the commercial issue must not be limited to economic or financial values. Qualifications must also be made in terms of:

- *Legal authorization*
- *Fiscal authorization*
- *Economic and financial capacity*

- *Market opinion on financial health*
- *Realistic behavior and long-term partnership*

Many other qualifications may and must be made to improve our technical and commercial analyses. Our intention is not to go deeper into this subject, but something must be said about the grade system for proposal selection.

Grades are commonly used for several technical or commercial items to evaluate vendors. However, this is only meaningful for homogeneous proposals. By and large, proposals are different and we must find terms of comparison. We can list a series of characteristics for a proposal: content, domain, etc., and define weights according to our decision criteria and, thus, calculate a technical grade. You can also do this for the commercial grade. Additionally, we align proposal values and normalize them in such a manner that the price also influences the commercial grade according to the weight we want. We then get a price grade. Again, we insist: this procedure only makes sense for similar things.

The final grade is composed of two grades contemplated according to the contracting strategy: quality or cost. It is important not to be deceived by low costs, but this is a matter of business strategy.

Depending on the selected subcontracting scenario, natural preponderance will be given to the technical grade or to the price grade. For example, subcontracting design, where expertise, competence, and quality are important and, knowing, further, that the overall project design corresponds to only a small part of the costs, we recommend the 70:30 ratio in favor of the technical grade. When we contract programming, and if we have a good quality assurance procedure, we can use a 40:60 ratio.

Negotiation and contract elaboration are not included in our scope.

We must supply schedule elaboration carefully with the vendor. RUP imposes a process characteristic here, and we must respect it: the development cycle is iterative, therefore, we must plan the iterations. The consequence is that, as the main project, we must divide the supply into iterations, and synchronize such iterations. The same rules and heuristics are valid both for one and for the other; but be careful, if the vendor is not familiar with RUP, the contracting party must impose project synchronism clearly.

We dealt with diligencing and inspection in the "Typical Management Model" chapter.

Homologation is the activity performed to ensure acceptance by the main project team. As for inspections, you must have well-defined acceptance criteria. The test plan must foresee homologation and define which test sets will be applied to validate receipt.

If everything goes as expected — which is rare — the contract closing activity must be the experience storage and registration vehicle. We evaluate not only the supply results regarding the project, but our own subcontracting process. This is the collective learning instrument, and it must be followed strictly.

8. Main Subcontracting Scenario Development Cases

Based on the scenarios identified above, we describe the set of activities and artifacts that compose the scenario.

By indicating the contracted party's scope, in the phase and iteration diagram, we automatically delimit the activities that correspond to this scope and assume that the contracted party will be responsibile for all these activities, while the contracting party will handle the remaining ones.

Each scenario establishes specific conditions for the business that must be reflected in the project's Business Case. For example, we must describe the contractual mode, which defines the form of payment and is directly connected to implementing the subcontracting strategy, in the Business Case with estimates and objectives.

To implement the subcontracting process with RUP, we need to select one (or more) scenarios, which we must describe in the project's Development Case. It must contain, in addition to the table that defines how to use the artifacts, the description of how the selected contractual mode will operate, defining acceptance and payment criteria. The roles, and the main lifecycle points where they will perform, must appear. The elaboration of a subcontract management plan materializes the Development Case information in the form of a plan and schedule.

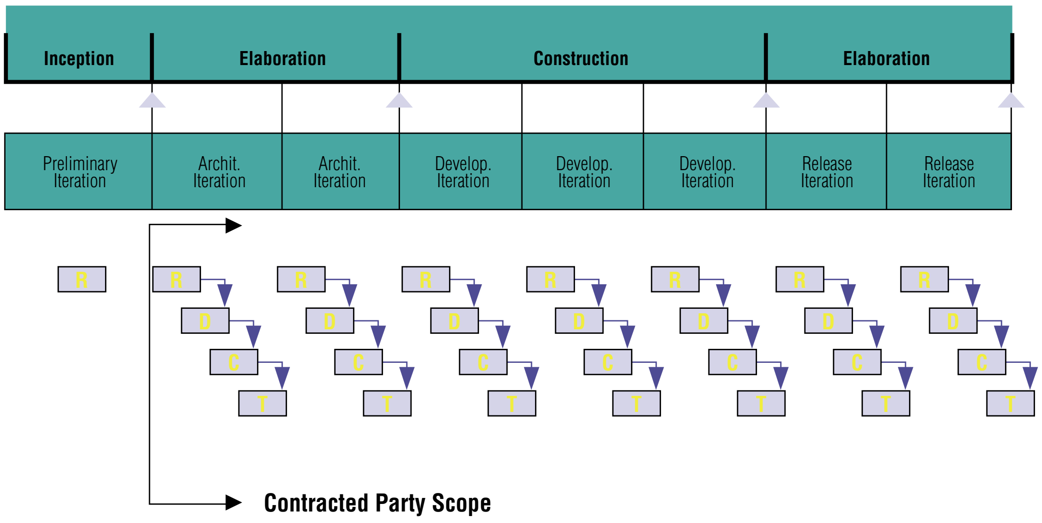
We will present the essential subcontracting artifacts in a table, with the following meanings:

Discipline	**Contracting Party**	**Contracted Party**
RUP Discipline Name (workflows)	Name of RUP artifacts for which the contracting party is responsible	Name of RUP artifacts for which the contracted party is responsible
	[Note: It is not expected that the contracted party's artifacts be exactly RUP artifacts, rather that their content and meaning be equivalent. See the Equivalent Artifact concept. Therefore, after mapping, this column must also reflect the name of the artifacts that are equivalent to RUP that the contracted party uses, if there is any discrepancy.]	

8.1 Scenario 1: Subcontracting Based on a Use Case Model

In this scenario, the contracting party executes RUP until defining the first Use Case model stage, which ends in the Inception phase. At this moment, the contracting party's project has already been sufficiently defined in order for the organization's directors to give the "go ahead." We can then ask why we should build the system in-house. Do we have enough data to generate a subcontracting specification?

The main characteristic of an organization that would answer yes to both issues is that it prioritizes the results in detriment to the technical domain on the system to be built. The contracted party is responsible for all iterations that will create the system, including contact with project stakeholders, to complete the Use Case model through its final stage. In RUP, the cutoff line will be based on the Elaboration phase, as in the following illustration:



Implications:

- *The contracted party must have access to the stakeholders to finish requirement specification.*
- *The contracting party only acquires skills in requirement modeling.*
- *The contracting party transfers most of the development risks.*
- *The contracted party will seek a margin to protect itself from the higher development risks.*

Artifacts:

Discipline	**Contracting Party**	**Contracted Party**
Requirements	Vision Use Case Model Supplementary Specifications Glossary Requirement Management Plan Requirement Attribute Domain Model (optional)	
Analysis & Design		Software Architecture Document Design Model Data Model Deployment Model Analysis Model (optional)
Implementation		Implementation Model Integration Build Plan
Test	Test Plan	Test Suite Test Results Test Case
Deployment		Bill of Materials Releases Notes Installation Artifacts Training Materials End-User Support Material
Configuration & Change	Config Management Plan Change Request	Config Management Plan
Project Management	Business Case Risk List Product Acceptance Plan Software Development Plan Iteration Plan Iteration Assessment Status Assessment Deployment Plan	Iteration Plan Deployment Plan
Environment	Development Case Use Case Guidelines Design Guidelines Program Guidelines Test Guidelines User Interface Guidelines	

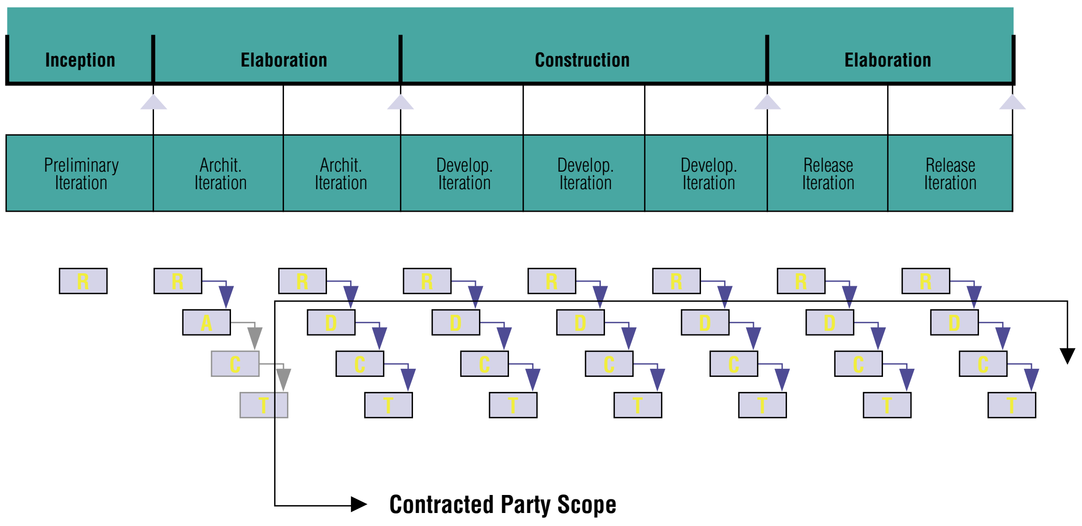
8.2 Scenario 2: Subcontracting Based on an Analysis Model

In this scenario, the contracting party executes RUP through Analysis model elaboration, normally optional. This means that the contracting party executes the first phase, Inception, creates the Use Case model through its first stage, confirmes the project's executability, and advances to the second phase, Elaboration.

In Elaboration, in the first iteration, the contracting party executes RUP to deepen the Use Case model and to perform the Analysis model, arriving at the Static Class Diagram, in the usual manner. At this point, there are sufficient elements to make a subcontracter build the system. Note the contracting party continues to be in charge of the Use Case model and will build it to the end.

Note: It is optional for the contracting party to conclude the iteration (in which it elaborated the Analysis model), arrive at an executable, and test it. The issue here is that the contracting party is not in charge of the Design model and, therefore, it is not concerned with the definitive architecture; on the contrary, it will outsource the work of elaborating the definitive architecture.

The main characteristic of an organization that acts this way would be the interest in dominating the system functionally and logically, without going into deploying or modeling details. In RUP, the cutoff line follows the requirement workflow from the first Elaboration iteration:



Implications:

- *The contracting party executes the iterations, however, with a smaller scope*
- *The contracting party dominates the functional model*
- *The contracting party dominates the logical model*
- *The contracted party doesn't need to have access to the stakeholders*

Artifacts:

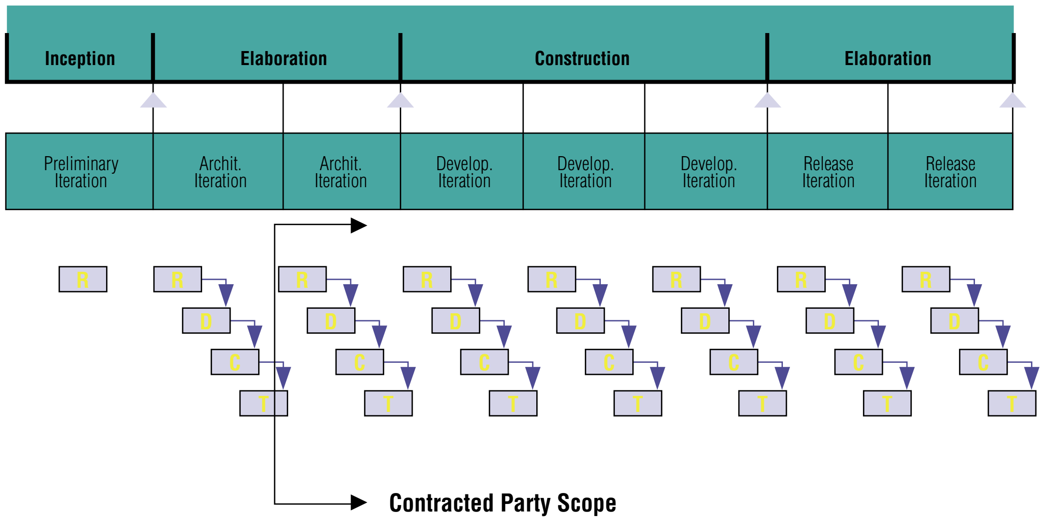
Discipline	Contracting Party	Contracted Party
Requirements	Vision Use Case Model Supplementary Specifications Glossary Requirement Management Plan Requirement Attribute Domain Model (optional)	
Analysis & Design	Analysis Model	Software Architecture Document Design Model Data Model Deployment Model User-Interface/Navigation Map
Implementation		Implementation Model Integration Build Plan
Test	Test Plan	Test Suite Test Results Test Case
Deployment		Bill of Materials Releases Notes Installation Artifacts Training Materials End-User Support Material
Configuration & Change	Config Management Plan Change Request	Config Management Plan
Project Management	Business Case Risk List Product Acceptance Plan Software Development Plan Iteration Plan Iteration Assessment Status Assessment Deployment Plan	Iteration Plan Deployment Plan
Environment	Development Case Use Case Guidelines Design Guidelines Program Guidelines Test Guidelines User Interface Guidelines	

8.3 Scenario 3: Subcontracting Based on a Design Model

In this scenario, the contracting party executes RUP through the elaboration of the first executable release, where it checks the main architecture problems. Therefore, it follows RUP through the Design model, arriving at the Static Class Diagram in the traditional manner.

Note: The number of contracted party iterations is not strict; in fact, it will perform as many iterations as it deems necessary to validate its architecture suppositions.

The main characteristic of an organization that selected this path would be reducing subcontracting risks.



Implications:

- *The contracting party only executes a few iterations in Elaboration.*
- *The contracting party must have the necessary ability to solve architecture issues.*
- *The contracted party must have access to the stakeholders to finish requirement survey.*
- *The prototype can be provided as an auxiliary specification.*

Artifacts:

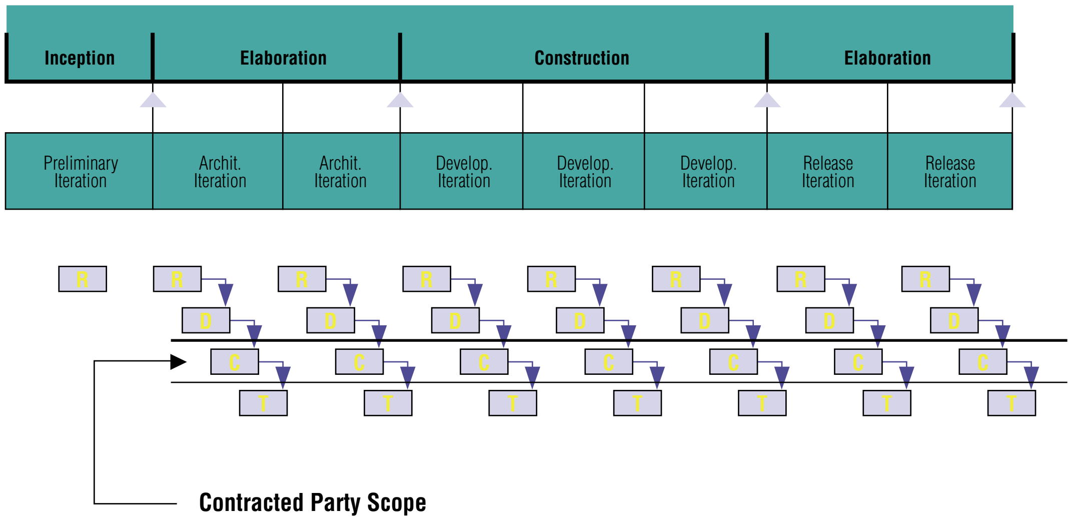
Discipline	Contracting Party	Contracted Party
Requirements	Vision Use Case Model Supplementary Specifications Glossary Requirement Management Plan Requirement Attribute Domain Model (optional)	
Analysis & Design	Software Architecture Document Analysis Model (optional) Design Model Prototype (optional)	Data Model Deployment Model User-Interface/Navigation Map
Implementation		Implementation Model Integration Build Plan
Test	Test Plan	Test Suite Test Results Test Case
Deployment		Bill of Materials Releases Notes Installation Artifacts Training Materials End-User Support Material
Configuration & Change	Config Management Plan Change Request	Config Management Plan
Project Management	Business Case Risk List Product Acceptance Plan Software Development Plan Iteration Plan Iteration Assessment Status Assessment Deployment Plan	Iteration Plan Deployment Plan
Environment	Development Case Use Case Guidelines Design Guidelines Program Guidelines Test Guidelines User Interface Guidelines	

8.4 Scenario 4: Programming Subcontracting

In this scenario, the contracting party executes RUP normally through the time when code implementation must begin; it then hires programming as specified in the Outsourced Software Factory model. Once the code is received, it applies the tests as prescribed in the process.

Note: In this scenario, the code the contracted party generates is integrated and the contracting party performs the homologation tests.

The main characteristic of this type of development organization is outsourcing whatever doesn't create control or dominance over the business system.



Implications:

- *The contracting party executes all RUP iterations.*
- *The contracting party must have the necessary ability to solve architecture issues.*
- *The contracted party doesn't need to have access to the stakeholders.*

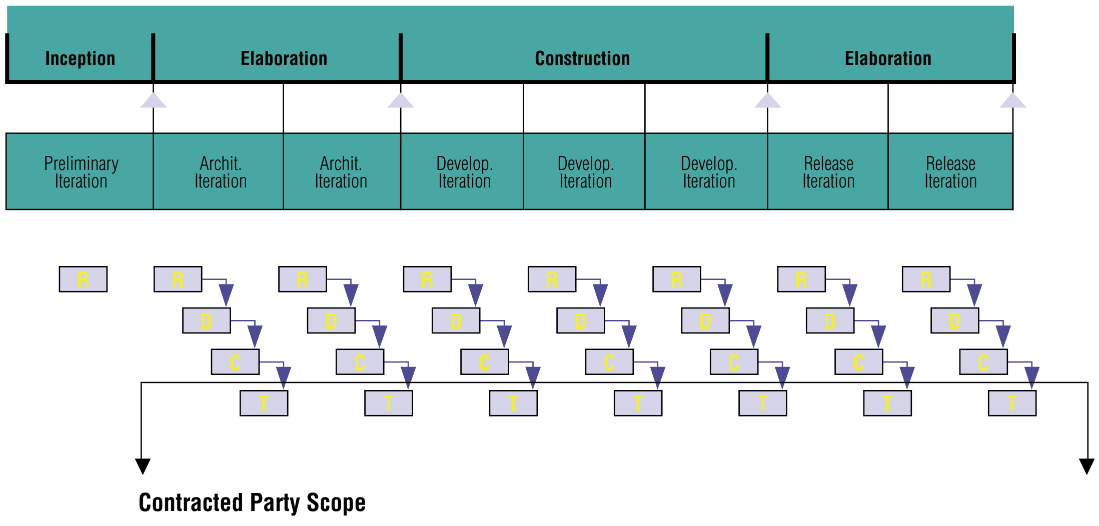
Artifacts:

Discipline	Contracting Party	Contracted Party
Requirements	Vision Use Case Model Supplementary Specifications Glossary Requirement Management Plan Requirement Attribute Domain Model (optional)	
Analysis & Design	Software Architecture Document Design Model Data Model Deployment Model User-Interface/Navigation Map	
Implementation		Implementation Model Integration Build Plan
Test	Test Plan Test Suite Test Results Test Case	
Deployment	Bill of Materials Releases Notes Installation Artifacts Training Materials End-User Support Material	
Configuration & Change	Config Management Plan Change Request	Config Management Plan
Project Management	Business Case Risk List Product Acceptance Plan Software Development Plan Iteration Plan Iteration Assessment Status Assessment Deployment Plan	Iteration Plan
Environment	Development Case Use Case Guidelines Design Guidelines Program Guidelines Test Guidelines User Interface Guidelines	

8.5 Scenario 5: Tests Subcontracting

In this scenario, the contracting party executes RUP in its entirety. It only doesn't execute the tests at each iteration's conclusion. Tests are outsourced in the "test bureau" mold. In this scenario, the bureau is in charge of planning, administrating, and performing the tests. The contracted party only has to perform initial project planning, which will be used as input for the contracted party to elaborate the test plans.

The main characteristic of this type of development organization is the lack of interest in acquiring test knowledge and skills.



Implications:

- *The contracting party executes all RUP iterations.*
- *The contracting party must have the necessary ability to solve architecture issues.*
- *The contracted party doesn't need to have access to the stakeholders.*

Artifacts:

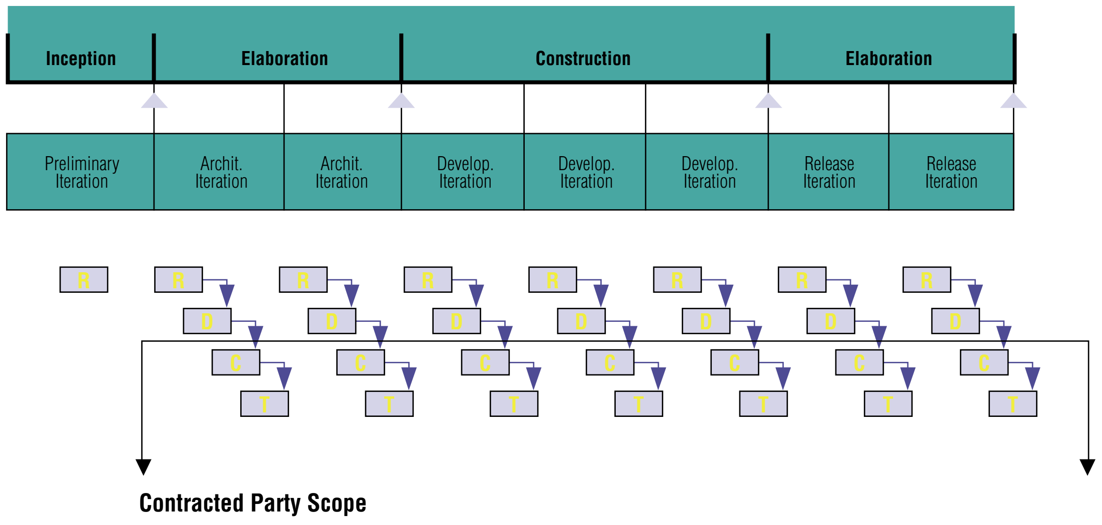
Discipline	Contracting Party	Contracted Party
Requirements	Vision Use Case Model Supplementary Specifications Glossary Requirement Management Plan Requirement Attribute Domain Model (optional)	
Analysis & Design	Software Architecture Document Design Model Data Model Deployment Model User-Interface/Navigation Map	
Implementation	Implementation Model Integration Build Plan	
Test		Test Plan Test Suite Test Results Test Case
Deployment	Bill of Materials Releases Notes Installation Artifacts Training Materials End-User Support Material	
Configuration & Change	Config Management Plan Change Request	Config Management Plan
Project Management	Business Case Risk List Product Acceptance Plan Software Development Plan Iteration Plan Status Assessment Deployment Plan	Iteration Plan Iteration Assessment
Environment	Development Case Use Case Guidelines Design Guidelines Program Guidelines Test Guidelines User Interface Guidelines	

8.6 Scenario 6: Programming and Test Subcontracting

In this scenario, the contracting party executes RUP normally through code implementation, when it subcontracts programming and tests. The contracted party is in charge of implementing the programs and applying the tests, but the test plans have already been established by the project.

Note: This is a hybrid mode mixing Scenarios 4 and 5, however, with the difference that the contracting party retains the domain and the knowledge in tests, and only outsources their application.

The main characteristic of an organization that would follow this route is that it would outsource whatever doesn't create control or dominance over the business system.



Implications:

- *The contracting party executes all RUP iterations*
- *The contracting party must have the necessary ability to solve architecture issues*
- *The contracted party doesn't need to have access to the stakeholders*

Discipline	Contracting Party	Contracted Party
Requirements	Vision Use Case Model Supplementary Specifications Glossary Requirement Management Plan Requirement Attribute Domain Model (optional)	
Analysis & Design	Software Architecture Document Design Model Data Model Deployment Model User-Interface/Navigation Map	
Implementation		Implementation Model Integration Build Plan
Test	Test Plan	Test Suite Test Results Test Case
Deployment	Bill of Materials Releases Notes Installation Artifacts Training Materials End-User Support Material	
Configuration & Change	Config Management Plan Change Request	Config Management Plan
Project Management	Business Case Risk List Product Acceptance Plan Software Development Plan Iteration Plan Iteration Assessment Status Assessment Deployment Plan	Iteration Plan Iteration Assessment
Environment	Development Case Use Case Guidelines Design Guidelines Program Guidelines Test Guidelines User Interface Guidelines	

9. Application Example for a Software Factory

In this didactic example, we select the programming subcontracting scenario, a software factory, which, in spite of the diverse concepts this term includes, allows you to see this article's scenario idea and contractual mode being applied.

We define a *software factory* as a programming vendor and, in this example, focus on the period involved in application development. The maintenance stage has peculiarities that go beyond this article's scope, basically requiring the definition of limits for which the project is considered and what can (or should) be treated in a functional team.

9.1 Specific Products in a Project

You must define the following results when applying subcontracting:

1. Scenario: Programming Subcontracting (Scenario 4, Chapter 8.4).

Description: "Our company, concerned with the exaggerated increase of extremely specialized functions that are foreign to our main business, decides to study outsourcing strategies with all of our current vendors with the purpose of, progressively, concentrating only the management, knowledge, and control of our business on our employees. Vendor X, an old and reliable partner based on its history with us, can provide this project's software programming. For the software development strategy, we want to maintain control over application design and integration because we recognize that generating componentized software is strategic for the business domain in which we perform. As a path to be followed, we indicate fomenting a strong componentization for the application architecture as a competitive differential."

2. Development Case: instantiation of Corporate Case Development.

During the Inception phase, we undertake the subcontracting Search & per future use Selection process phase activities.

During the remaining RUP phases, the Provision phase activities (see Chapter 4).

You may complemente Development Case with subcontracting cycle activities (Chapter 7) or generate a RUP configuration, adding the subcontracting activities.

Notes:

- *Formal-external are artifacts we use to hire the vendor.*
- *Formal-internal are artifacts we use to document our project.*
- *Informal are the artifacts that will not be perennial and don't document the project.*

It is very important, in an actual case, to define the "tool" column, as the tools characterize the documentation and formalization mechanisms for project participants.

Discipline	**Contracting Party**	**Contracted Party**	**Use**	**Tools**	**Notes**
Requirements	Vision		Formal-external		
	Use Case Model		Formal-external		Model complete with all artifacts the RUP suggests.
	Supplementary Specifications		Formal-external		
	Glossary		Formal-external		Optionally controlled by RequisitePro.
	Requirement Management Plan		Formal-internal		
	Requirement Attributes		Formal-internal		
	Domain Model (optional)		Formal-internal		Optional artifact that can substitute or complement the Glossary.

Analysis & Design	Software Architecture Document		Formal-internal		Elaborating this artifact well, observing not only the project but the corporation as a whole, is one of the main reasons to select this factory scenario. It must be complete and corporative.
	Design Model		Formal-external		Model complete with all artifacts RUP suggests.
	Data Model		Formal-external		
	Deployment Model		Formal-external		
	User-Interface/Nav. Map		Formal-external		
Implementation		Implementation Model	Formal-external		Model complete with all artifacts the RUP suggests. This is the vendor's main deliverable.
		Integration Build Plan	Formal-external		This document must enter the vendor's deliverables list.
		Developer Test	Formal-external		Corresponds to the unit or bench test.
Test	Test Plan		Formal-external		
	Test Suite		Formal-internal		Complete model, with test script, workload, and data organized for regression test application.
	Test Results		Formal-external		
	Test Case		Formal-external		
Deployment	Bill of Materials		Formal-internal		
	Releases Notes		Formal-internal		
	Installation Artifacts		Formal-internal		Optionally, its programming can be outsourced. In this case, the procedure is analogous to the main system artifacts.
	Training Materials		Formal-internal		
	End-User Support Material		Formal-internal		
Configuration & Change Management	Config Management Plan	Config Management Plan	Formal-external		This artifact is produced in common agreement between the parts. It must be in the vendor's deliverables list and includes the plan itself and the project's repository model.
	Change Request		Formal-external		In the form of a workflow application between contracting and contracted parties.

Project Management	Business Case		Formal-internal		
	Risk List		Formal-internal		
	Product Acceptance Plan		Formal-internal		
	Software Development Plan		Formal-external		
	Iteration Plan	Iteration Plan	Formal-external		This artifact is produced in common agreement between the parts. It must enter the vendor's deliverables list.
	Iteration Assessment		Formal-external		
	Status Assessment		Formal-external		Forwarding agent report or vendor deliverable, with metrics and checkpoints defined via a contract.
	Deployment Plan		Formal-internal		
Environment	Development Case		Formal-internal		
	Use Case Guidelines		Formal-internal		
	Design Guidelines		Formal-internal		
	Program Guidelines		Formal-external		
	Test Guidelines		Formal-internal		
	User Interface Guidelines		Formal-external		

3. Business Case: Elaborated for the entire project, with highlight on subcontracting. Here, the contractual mode must be defined; assume that the trend is to select unit price, perhaps because there is interest in paying for a use case implemented for the vendors who used to be paid via hourly fees (common payment mode in the market). The idea is that the nature of the work is known (effort to implement a use case), but the use case amount is unknown.

Contractual mode: Use case unit price classified as:

- *Simple: up to three use case scenarios [Note:* Use Case scenario *is a technical term, and has nothing to do with the subcontracting scenario used in this article.]*
- *Medium: three to seven use case scenarios*
- *Complex: more than seven use case scenarios*

You must stipulate a value using market information, internal historical series, etc., as a base. This value will work as a parameter for vendor selection, but it will be subject to the actual project contracting conditions. ROI and break-even analyses will be performed on the estimated value.

4. Subcontract Party Management Plan: Developed as a specialized plan in the Software Development Plan (SDP).

We define the Forwarding Agent and Inspector manager roles in this plan. It is necessary to define profiles and responsibilities, for example:

Forwarding Agent: Role the Project Manager performs at the supply control points established in advance. Purpose: Check the schedule.

Inspector: Role the corporate SQA performs on the deliverables established in advance, during the supply control points. Purpose: Guarantee, through technical revisions, the same quality level achieved in internal projects. It applies specialized resources available in the team and/or corporation.

5. RFPs and Contracts for Subcontracting

When the project's documentation is satisfactory, you can elaborate the external documents that will guide the subcontracting business, basically a Request for Proposal (RFP) and the service contract. Both reflect project artifact content and must be part of the subcontracting baseline.

10. Conclusion

Although this document doesn't aim at exhausting the possible subcontracting modes using RUP, we believe a composition among the five or six most important scenarios is likely to suffice to help us assemble the subcontracting strategy.

We attempted to show the implications of making a selection among the several subcontracting options and alerted to that, which we believe to be the main issue in the process: the adequate selection of the contractual mode and of the scenario.

We remained within RUP scope and always imagined that the contracting organization has RUP as a process that is deployed in its organizational culture.

G517-2182-00