

PUMA

Proposition pour l'Unification des Méthodes Agiles

Proposal for unifying agile methods

TABLE OF CONTENTS

1.1	Predictability Vs. Adaptability	2
1.2	Complementary methods	2
1.3	Techniques shared by the methods	3
1.4	Techniques specific to a method	3
1.5	Towards optimal agility	5
1.6	Managing stakes and risks	5
1.7	Relationship between Owner-User and CIO	6
1.8	Outsourcing choices	6
1.9	Agile planning strategy	7
1.10	Agile QAP	7
1.11	Modus operandi for emergency halts	7
1.12	Adapting the methodological level	8
1.13	Model structure	8
1.14	Resource commitment	9
1.15	Motivating A Project Team	9
1.16	The ten commandments of RAD	10
1.17	Reminders and conclusions	10

Jean-Pierre Vickoff



Although Jean-Pierre is a team leader and an Integrated Systems Architect, he always involves himself directly in the actual conception and development of his projects (as SWAT requires). He specializes in directing key strategic projects with high priority time constraints.

He is an active participant in state of the art evolution and has authored several communications and books dealing with methods. His publications focus on iterative incremental approaches such as RAD. He occasionally gives conferences and directs training programs for private companies, universities and France's "Grandes Ecoles".

A development process that associates both performance and quality naturally requires an iterative incremental project management style, coupled to a simple phasing. James Martin's¹ RAD method gave rise to these pragmatic principles. More recently UML revealed the procedural need for a strong pairing between modeling outline and development by prototyping. To this day a handful of methods meet the Agile criteria.

The principal methods are Adaptative Software Development (ASD), Feature Driven Development (FDD), Crystal Clear, Dynamic Software Development Method (**DSDM**), Rapid Application Development (**RAD**), Scrum, Xtreme Programming (**XP**) and Rational Unified Process (**RUP**). Since these methods all exhibit a varying market potential, it is reasonable to cite RUP and take its influence into consideration, even though it is a proprietary method.

¹ Late 80's for RAD and mid-90's for UML.

Thanks to the small number of such methods, they tend to be globally comparable so that most of the standards and techniques they put forward are common to all of them. A study of the proposed standards reveals a set of practices stemming from the roots of RAD and shared by all. Only a few techniques that complement each other or that are adapted to specific typologies and project sizes distinguish the methods.

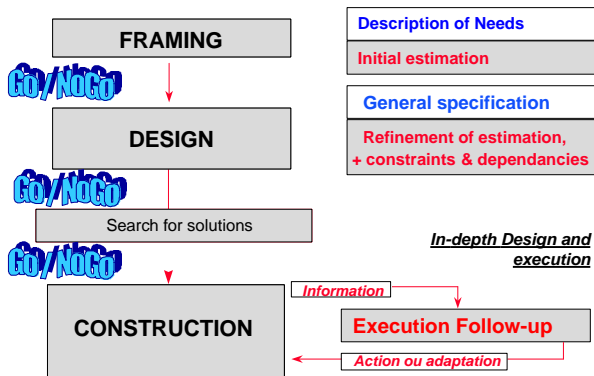
Summed up below are the four main ideas of these Agile approaches :

- 1 – **Communication and the resulting interaction** over respect of contract specifications.
- 2 – **Ability and resource involvement** over strict observance of formal processes and a “toolbox” vision of development.
- 3 – **Effective delivery of features** over production of a cumbersome documentation.
- 4 – **Acceptance of change** and modification of **priorities** over loyalty to a static plan.

1.1 Predictability Vs. Adaptability

The paradigm of traditional methods is **predictability**. The paradigm of Agile methods is **adaptability**. It should be clear that no approach is entirely reducible to one of these paradigms. All the methods try to deal with conflicting goals of flexibility and rigidity with more or less finesse and with varying degrees of success depending on the context:

- Predictive methods attempt to curb incertitude using very precise and detailed planning at the very start of the project. This risk removal procedure implies that the application requirements be set in stone.
- Agile methods favor adaptation to context changes, starting from an initial plan outline that will be regularly reevaluated. Reevaluation will serve as input to a GO or NO GO type decision (see Fig. 1) each time a significant change is applied to the initial outline.



“To effectively control the allowable error margin.”

Figure 1 – Continuous Evaluation / Decision process.

Each method can be positioned on a scale ranging from the most “predictive” to the most “adaptable”. If one wants to get a clearer view of the field of application of the different methods, it is necessary to list them as a function of these criterions.

Methods vs. The 4 Agile Criterions

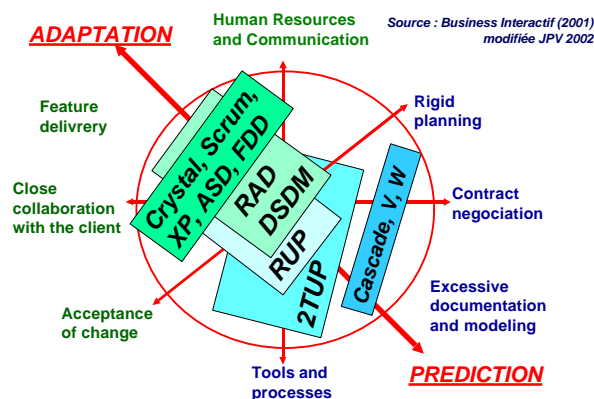


Figure 2 - Adaptability / Predictability scale.

The differences emphasized in Fig. 3 justify a rigorous analysis of the project environment and application category in order to determine which aspects of the method will allow for risk limitation and what level of methodological service and application quality will be needed.

1.2 Complementary methods

Closer examination of agile methods reveals that they rest on the same foundations. They can only be differentiated by a few techniques that are either mutually complementary or better adapted to specific typologies and specific project sizes. On the other hand each method offers a varying coverage of a team leader’s typical concerns:

- 1 - **Respect of urbanization** (position of the project in the information system).
- 2 - **Direction** (resource management, planning, monitoring, quality, reporting, visibility).
- 3 – **Application engineering** (specification management, conception & development, approval of deliverable products).
- 4 – **Change management** (organizational impact and deployment).

Scope / Activity - Application - Project -

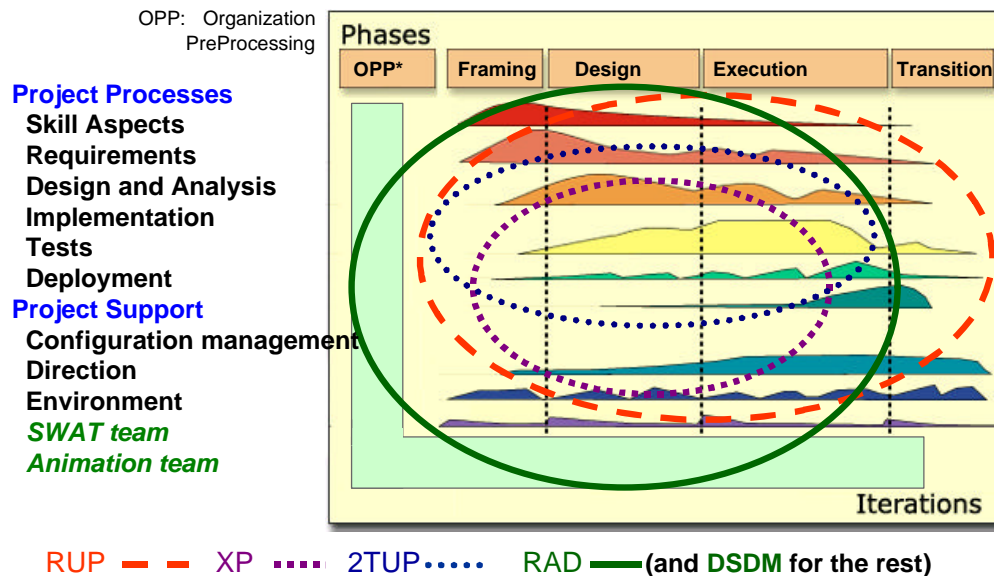


Figure 3 - Each method deals in different ways with application engineering and project direction.

1.3 Techniques shared by the methods

- Specification and **continuous assessment** of *Requirements*.
- Participation of **end user** in workgroups.
- **Empowerment** of negotiation partner.
- **Autonomy** and centralized organization of the team (motivation).
- **Variable methodological QOS** depending on the stakes of the project.
- Direction focused on **stakes and risks**.
- Pursuit of excellence in **conception technique**.
- Graphical view of **necessary/sufficient modeling**.
- Vision of **necessary/sufficient integrated documentation**.
- Strategic global planning based on **quick iterations**.
- Execution through **reference points** by active, iterative and incremental **prototyping**.
- Reasonable norms and techniques for **code quality** (metric).
- **Component based** architecture, **change management**.
- Continuous pursuit of optimization and upgrade of techniques.

1.4 Techniques specific to a method

Let's take a look at the most significant techniques that distinguish the different methods.

The DSDM method characterizes itself by insisting on **specialization of the protagonists**. A DSDM meeting will usually bring together executive sponsors, ambassador users, visionaries, advising users, reporters and last but not least the animator-facilitator.

The Group for Animation and Report (GAR) is a substantial contribution from RAD to project communication and to formalization of application requirements.

The SCRUM method sets itself apart by calling for **daily meetings**. The different objectives of these sessions include boosting morale, synchronizing tasks, unlocking difficult situations and generally increasing the quantity of knowledge sharing.

FDD's distinguishing idea is **Mission Focus**, which calls for a noticeable orientation towards an immediate and measurable goal. This tends to reinforce the global ambition of a given iteration.

This concept also appears in RAD under the name of Focus Objectives and in SCRUM as *Sprint*.

As its name suggests, FDD recommends **Feature Driven Development**. This technique is defined by two concepts, *Feature and Feature Set*. Priority is given to high value features.

Related ideas are put forward by RAD: delivery of a reduced Feature Set and theme based delivery.

XProgramming : Fundamental Ideas

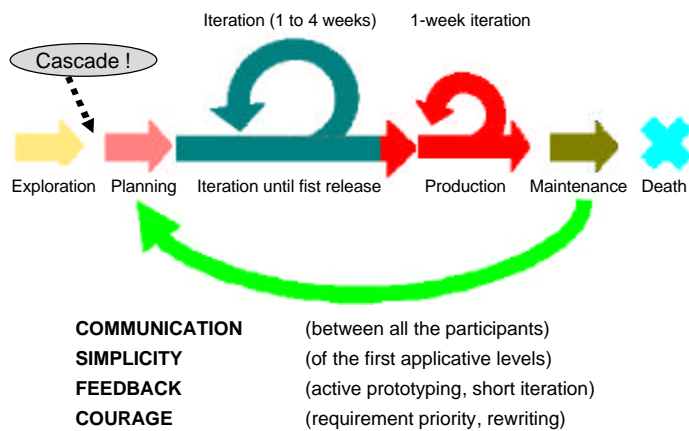


Figure 4 – XP’s project cycle (the other Agile methods follow an analogous cycle).

XP concentrates on Application Building. Its main originality resides in its approach to preparation, materialized by a **planning game** simultaneously involving users and programmers. **Binomial coding, collective appropriation of source code, refactoring and continuous integration** all deal with code writing and are also worth considering.

In a similar way, RAD advises individual code checking sessions as a first step followed by a collective session. On the other hand, RAD restricts binomial coding to strategic or complex code sections.

may seem very similar to the cascade cycle but it does induce a certain form of internal iterativeness in some tasks. This cycle is not for sure an Agile approach, nevertheless 2TUP introduces some interesting quality and performance enhancements such as reusable services and generic conception (**Framework and Design Pattern**), both of these concepts being very close to RUP’s architectural mechanisms.

RUP³ is characterized by its global “**4+1 perspectives**” angle of attack (Fig. 6).

RUP also introduces, as RAD does, a formal **Guiding Process** that serves as an adaptable activity guide. Unfortunately in RUP’s case, this process is proprietary and tool-oriented.

RAD calls for a Group for Animation and Report (GAR) and a self-reliant **SWAT** team. This team is essentially made up of designer-coders corresponding to the same profile and trained in complementary technical skills. The SWAT team interacts with the end users in a specially equipped room designed to function as a communications stage (**RAD room**).

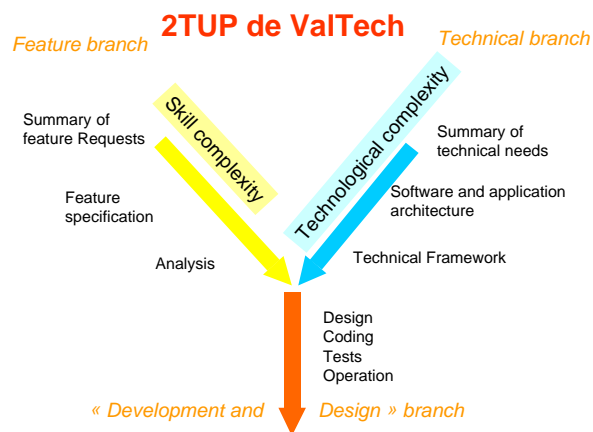


Figure 5 – 2TUP’s Y-shaped cycle.

2TUP² calls for a **Y-shaped life cycle** that separates resolution of technical issues from resolution of feature issues. The 2TUP life cycle

² A combination of V-shaped cycle and USDP according to Miguel Moquillon. A combination of project direction and modeling according to the author.

³ One of the foundations of the OOP method USDP (Unified Software Development Process), this method was created by the designers of UML to specify the design and analysis model. Its main implementations are 2TUP (Valtech) and RUP (Rational).

Object Modeling: (4+1) Perspective

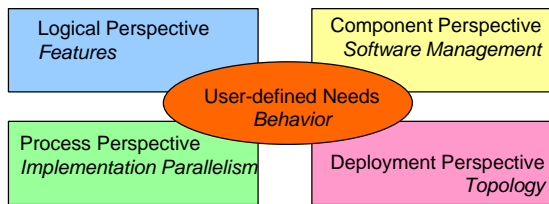


Figure 6 – The 4+1 perspective of RUP.

RAD also recommends **adjusting the size and experience⁴ of the workgroups** for each different phase of the project in order to optimize resource involvement and preserve morale by handing out work that suits the workforce's different interests. Meetings are organized on the basis of three different steps (**3-step meetings**) and techniques enforcing **continuous review** are put into effect during these meetings.

RAD : Rapid Application Development

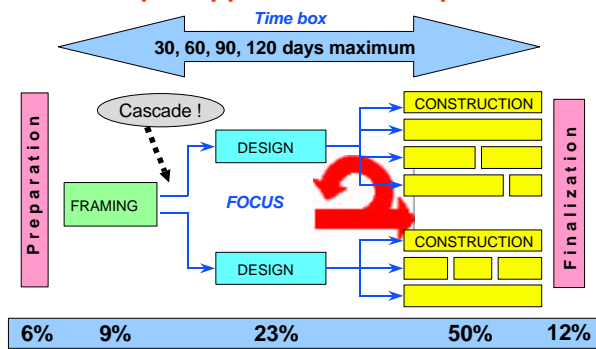


Figure 7 – RAD cycle: **parallelisation** and **serialization**.

1.5 Towards optimal agility

Although the task of merging the different Agile methods into a single one could prove very daunting, it should nevertheless be possible to create a unified method regrouping all the sensible practices of the different systems.

Once the techniques used by all have been separated from those specific to one method, it is easy to imagine an optimal method that could adapt itself to each particular type of project (Fig. 8). This unified Agile method would incorporate all or part of the set of shared techniques and some of the method-specific techniques most suitable to the context. All of these would of course be adapted to a different methodological quality of service.

⁴ The **animator-facilitator** is responsible for implementing these variations.

Adapting the method to the project's size

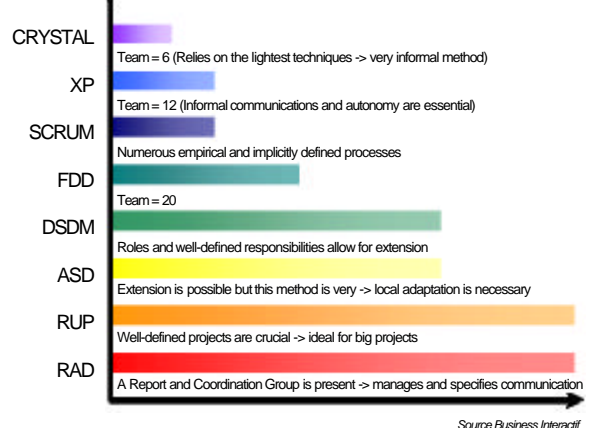


Figure 8 – Project size / Appropriateness of the method.

1.6 Managing stakes and risks

The leader of a development project is usually confronted to 3 different and concurring processes:

- Managing a **profitable** project.
- Building a **customized** application.
- Organize **expected** change.

Each one of the previous aspects brings about a dialogue between stakes and risks. A secure leadership process will therefore cover all these aspects (Fig. 9). The pathos specific to the ailing project sheds light on the overlapping causes of failure. On the other hand, the effects are usually limited to four different types of deficiencies:

- Feature adequacy.
- Technical quality.
- Economic context of execution.
- Readiness to change.

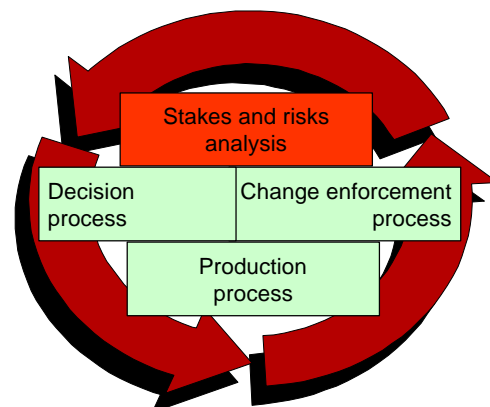


Figure 9 – Directing a project by stakes and risks.

Main sources of conflict in a IT project

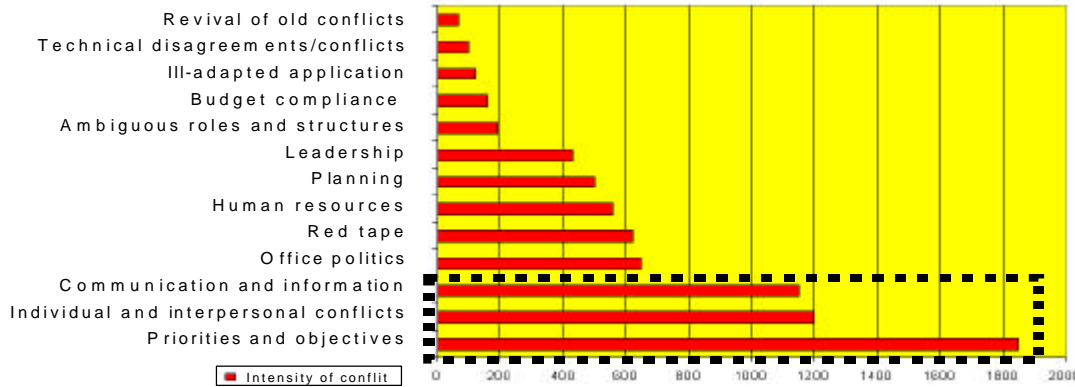


Figure 10 – Risk in projects (source Hervé Courtot).

When one or more of these deficiencies shows up, the project is in dire straits. Efficient intervention and redirection of an ailing project call for efficient management of these new risks. Hervé Courtot, an expert in this field, has determined that the three most significant risky areas are communication, human relations and arbitration of priorities (Fig. 10).

Any unified agile method must therefore focus on taming these recurrent risks. **Neutral animation-facilitation** is RAD's most significant contribution to prevention of these three eventualities.

1.7 Relationship between Owner-User and CIO

In modern applications, the **skills** adapt themselves to the compression of time. The solutions are bound to **technology** and evolve along with the project. The complexity of modern solutions imposes an between Owner-User with the ability to project its immediate needs into the future and suggest an applicative scheme. The strategic (and if possible permanent) participation of the end-users becomes crucial.

Updated Owner-User / CIO relation

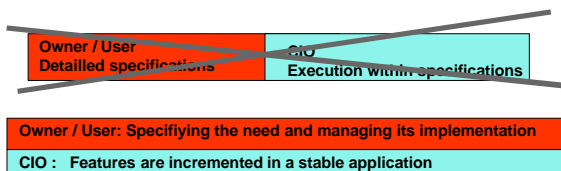


Figure 11 - The updated owner-user / CIO relation.

CIO (Chief Information Officer and his technical team) must be looked upon as an entity providing technological support. Once this has been established it follows that no matter what some people may say, the relationship between Owner-User and **CIO** is not null and void! In fact the concept that has been rendered useless is that of sequential intervention of both parties (Fig. 11). One can even go so far as to

state that the connection between both entities has never been so important. Especially if one takes into account the need to outsource parts of the solution (the different roles must be officialized in a contract).

This model fits into a universal⁵ logic and is highly recommended. Definition of responsibility must always be clear in agile methods, as well as in traditional methods.

The following materialization should help us get a better grip of the subtleties introduced by these concepts:

- Between Owner-User deals with the evolution of organizational skills. An between Owner-User thinks in terms of information systems, processes and applications. An between Owner-User directs the projects it finances.
- CIO** is in charge of delivering a solution. An **CIO** thinks in terms of computer systems, design and implementation. A **CIO** manages the project and engineers the solution.

1.8 Outsourcing choices

Given the increasing complexity of applications in general, outsourcing the solution is now a vital necessity. The model that RAD suggests to the project leader has seven outsourceable layers that can be customized to fit the project environment:

1. Development of the GENERIC component.
2. Development of the SPECIFIC component.
3. MAINTENANCE strategy.
4. DEPLOYMENT solution.
5. User ASSISTANCE strategy.
6. OPERATIONS solution.
7. LOGISTICS solution.

⁵ In the US: Owner/Chief Information Officer (CIO).

1.9 Agile planning strategy

There are four basic types of constraints in a project. RAD recommends management techniques adapted to each one of these constraints (Fig. 12, right column). These four constraints are unfortunately contradictory and anyone trying to exert optimal control over them will be facing a serious combinatorics problem. The strategic planning assistant included in the Evaluator software tool can help you bring these problems under control. Evaluator is freely available on www.RAD.fr.

RAD : planning Strategy

	Strategic (deadlines)	Time Boxing
	Safe (resources)	SWAT
	Reliable (visibility, quality)	Focus
	Cheap (budget)	Target Costing

"Four contradictory constraints and a serious optimization problem"

Figure 12 - Four different types of constraints.

1.10 Agile QAP

Agility and quality control are not conflicting concepts in the RAD approach. Quality must be profitable and so the QAP (Agile Quality Assurance Plan) will be minimalist (but CMM-normalized). The principle underlying its construction must take into account the following thought: "As useful as any undertaking may seem, if it is not directly related to production then it is a superfluous maneuver." The following points describe a generic QAP that can be formalized during the creation of the project-specific QAP :

- Quality control and quality of service.
- Organization, communication and direction.
- Management of software quality.
- Software development method.
- Management of requirements.
- Management of the testing schedule.
- Management of disagreements.
- Quantitative management of costs.
- Management of planning.
- Management of documentation.
- Pragmatic management of risk.
- Management of configurations and software versions.

Using this list, the supervisor will choose which approaches yield the best risk-prevention probabilities. In short, this QAP contains the minimum number of elements required to conduct a safe development project. If an extremely simple *Agile* QAP is called for, the following four points should be taken into consideration:

1. Communication and commitment plan.
2. Management of requirements, testing schedule and disagreements.
3. Application referential and project memory.
4. Planning, appraisal and follow-up of the project.

These 4 points may be summed up in the following way: (1) The appropriate negotiating partners (end-users and owner-user) give themselves the means (2) to express their needs and approve a reusable execution scheme (3). The project leader and the CIO (technicians) can then certify the production environment (4).

1.11 Modus operandi for emergency halts

Any project management method should include a modus operandi for emergency halts. But close examination of the oldest predictive methods puts us in an optimistic frame of mind: their authors have never experienced failure and the pupils are encouraged to follow the lead.

RAD's main strength in this area is the presence of the Animator-facilitator. This external protagonist should have a neutral relation to all the actors involved in the project and should answer to a higher authority. When the context changes or when the project environment takes a turn for the worse, the animator reports back to the executive who empowered him. The executive can then make a prompt decision based on objective input.

In a traditional environment, the participants are actively involved in the different problems. Therefore they tend to hide them in order to correct them later. The ROI, the plans, the budgets are then doctored. This situation makes delivery of objective information to higher levels of management more difficult and usually creates a split in the command and decision chain. When reality is subjected to so much alteration any refocusing attempt sheds light on organizational discrepancies. The notion of *Courage* (one of the four fundamental values put forward in *Extreme Programming*) takes all its meaning and some start regretting that *Feedback* was not taken into consideration earlier in the project's life.

1.12 Adapting the methodological level

A methodological solution succeeding in a given environment may not be adapted to a project of different size or complexity. The tailoring of method and production quality to application lifecycle and economic context has become a necessary evil. The following paragraph lists the different methodological quality of service components taken into account by RAD and the *Agiles* methods :

Service level “project management”

- Planning and follow-up of progress
- Planning and follow-up of risks
- Planning and follow-up of security
- Formalization of direction activities
- Formalization of requirement and changes
- Formalization of configuration management
- Formalization of test management

Service level “application engineering”

- Technical quality of the application
- Quality of the application’s features
- Quality of the technical documentation
- Quality of the user guide

Notice the separation between direction of the project and engineering of the application.

Example: an application destined to be used by a small number of permanent executives should not be documented in the same way as an application tailored for hundreds of high-turnover users.

1.13 Model structure

The kind⁶ of model used to reproduce the project and the level of detail exhibited in this model must be chosen in accordance with the project’s context. It’s fairly obvious that Object Oriented Conception and UML heavily influence most of the recent models. On the other hand it is not clear whether a comprehensive⁷ and uncompromising UML model is absolutely necessary.

Agile Modeling is a simplifying school of thought that considers this excessive theorizing to be ill adapted to projects with strong time or money constraints.

The pragmatic project leader must always keep in mind that **the final objective is an application** and not a model. Judging by my own experience,

I predict that abuses similar to those brought forward by Merise a few years ago will surface with UML. The most worrying phenomenon is the fanaticism of Object Oriented Design supporters. No criticism of the practices their sect imposes is tolerated.

Object Oriented Design has suggested in the last few years a number of techniques that are not related to any particular kind of modeling scheme, the most useful being concealment, modularity, abstraction, encapsulation, cohesion, coupling, hierarchical structure, polymorphism, basic algorithmics and data & processing structuring.

The ultimate goal of any efficient design is to facilitate development and maintenance but also to position the project with regard to essential factor: return on investment. The principal of “design in light of further modifications” was invented for this reason. It is essentially base on “information concealment”. It is necessary to follow the steps listed below in order to apply this technique:

- Define the features precisely.
- Identify the unstable management rules.
- Encapsulate the changeable sections.
- Formally define the call interfaces.
- Apply modular programming.

Development big shots that did not immediately grasp the benefits of this approach were eventually forced to make amends. Fred Brooks “My criticism of this technique is one the few mistakes present in the first edition of my book, *The Mythical Man-Month*”. Boehm “This is one of the rare theoretical methods that has proved useful in the real world”. Systematic use of sensible design guidelines would already be a major step forward, regardless of the model used. According to Microsoft’s statistics (*54 règles pour un grand logiciel*) 97.5% of data processors do not know or do not use these basic design guidelines.

⁶ *The concern and abstraction of Merise, the Flows associated to the entity/relation, OOC...*

⁷ *No less than 8 different diagrams.*

1.14 Resource commitment

The changing nature of applications requires the expression of a need whose complexity is continually increasing. At the same time the technologies responsible for providing solutions are also liable to evolve. On top of this the main restriction is often linked to time-compression. **The Project Mode is the answer to these constraints.**

In this organizational scheme, all the participants involved in a given mission are regrouped at the same time in the same place.

Projects managed with this approach are successful. Knowing this, one could be surprised with the endurance of matrix-type resource management methods. Supporters of this approach (who are in general genuine managers) hope to optimize resource commitment over several simultaneous projects. This theory might be efficient when applied to small maintenance tasks but it is ineffective when used in development projects: **squandering does not pay!**

The culprit is human nature: loss of recognition and responsibility. During my career, I have seen individuals cease any form of participation even when they were booked for only 5% of their available time! On other occasions it was the plethora of committed resources that led to the same results: "When it seems that everybody is involved in everything, most of the time nobody is involved in anything".

The headcount in RAD team is reasonable (four is ideal, 8 is the upper limit) but the team as whole is involved a single segment from the outset of the project (Framing).

1.15 Motivating A Project Team

Agile methods try to deal with the specificities of the human resources instead of going against the flow. The best guarantee for a project is the enthusiastic involvement of a motivated team. The ability to work as a team and interact with the end-user is essential.

RAD recommends a special kind of team: the SWAT team (Skill With Advanced Tools, derived from *Special Weapons And Tactics* in law enforcement jargon).

The elements of this team have a distinctive⁸ designer-coder profile but are well versed in a specialty and are trained in RAD discussion

techniques. They focus on interaction with the end-user while complementing and supporting each other.

To give a team exceptional motivation, RAD recommends:

- Precisely defining of the challenge.
- Formalizing the reward (depending on whether or not the set reference points were reached).
- Voluntary enrollment in the team.
- Organizing the team's self-reliance in terms of means and work organization.
- Quantifying the time spent by team members helping less competent colleagues.
- Personalizing the tasks to fit desires and resource availability.
- Publishing the results regularly and honestly (no diplomatic language).
- Celebrating the achievement of important reference points using a dedicated budget (restaurant diners, field trips).
- Encouraging the surfacing of a team group identity (this point is not very well understood in France).

The return on investment of these American-style practices is obvious.

From a technical point of view, the structure of a SWAT team recommends that you favor the establishment of generalist-experts⁹. Make a list of these proficiencies and post them on a drawing board that employees can refer to when they have a narrowly targeted question.

The time spent exchanging knowledge and favors must be quantified so that the most competent team members are not penalized by the large sum of assistance they are likely to give out. This will save a lot of time when searching for technical solutions.

It will also prevent someone seeking advice from being afraid to ask questions because the expert is not compensated for time invested in knowledge transfer.

This last item being understood, **reward must focus on team achievements and not on individual ones**, since a SWAT team must remain a synergy and a set of balancing specialists involved in a team achievement.

⁸ The introduction of a feature in a SWAT team is well suited to demanding modern applications.

⁹ Generalist in design and development and specialized in a complementary field.

1.16 The ten commandments of RAD

As in most human activities, luck may play its part in the success of a project. But following judicious guidelines ensures that this luck capital is not carelessly squandered.

CONSTRUCTION : stages for a perfect execution

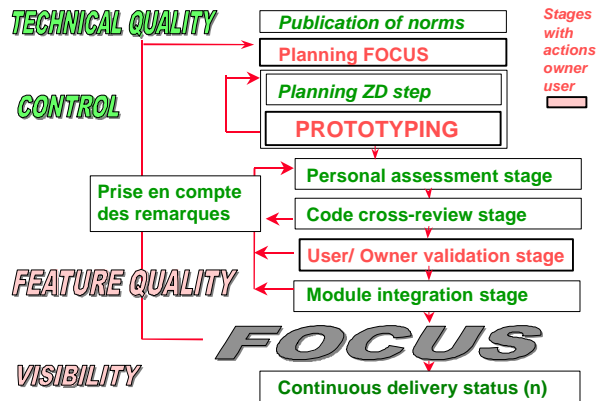


Figure 13 - Techniques for controlling execution.

Here are RAD's Ten Commandments:

1. Demand an efficient organizational and technical environment.
2. Practice continuous assessment with the end-user.
3. Plan realistically at the very beginning and then regularly re-evaluate (GO/NOGO).
4. Respect the phasing and the continuation prerequisites.
5. Respect the time limit (optimum 90 days).
6. Immediately assess operational viability (2TUP's Y-shaped lifecycle).
7. Organize a team possessing the technical and functional skills to meet the challenge.
8. Apply the fundamental concepts of OOC.
9. Apply the techniques for execution control: Zero-Defect steps and code assessment (Fig. 13).
10. Make sure that each visibility Focus enforces continuous validation.

1.17 Reminders and conclusions

In most projects, technique is rarely the sole cause of failure (*appendix A*). Human relations and economic constraints contain many potential pitfalls.

New economy projects, innovating projects, cross-field projects and even cross-organization projects underline this observation. A successful project must associate two contradictory perceptions: **creativity**, a prerequisite for innovation, and **rigidity** of the strategic and economic constraints. Managing **stakes, risks and constraints** is therefore essential.

One thing is obvious: this new kind of project is pushing the envelopes of IT development teams, project leaders and owner-users. They need to think up a new formal development structure based on acts and responsibilities.

Next comes the need for a **semi-iterative development method** whose security is enforced by a reliable and adaptable process.

In order to insure economic viability, all of these concepts must be supported by a **variable methodological QOS and a variable expected quality QOS**.

What's more, when innovation reaches a certain degree, it will ostensibly affect the core of the trade. In this case the project reveals the need for a business process engineering, an action that OOP naturally imposes. This observation often forces the organization as a whole to mutate. Such a huge change of production habits (one might even use the term cultural habits) is still exceptional and is considered by management to be "the time of all dangers". Thus, the obstacles standing in the way of evolution are still very powerful.

These obstacles must be taken into account when a method is applied in its own enlarged communications context.

Beyond project management, the critical evolution brought on by an iterative and incremental method based on user input is the positive rhythm of change it induces in the organization.

Jean-Pierre Vickoff

APPENDIX A – Main success and failure factors in recent projects.

Aspect	Success Factors	Failure Factors
Political	The project leader and/or the coordination group leader answer to a higher authority (main management in the case of a key project).	The project leader is not sufficiently empowered. He cannot settle arguments or restrain office politics.
Direction	The project is <i>dynamically</i> directed, with management of stakes and risks being the central idea.	The project is <i>administratively</i> directed, with management of budgets and resources being the main focus.
Organization	Use of a project mode that centralizes all the different protagonists on the same stage. Formalization of vertical and horizontal responsibility delegation in short missions.	Use of part-time resources and/or use of resources that do not directly depend of the project management. Partial knowledge of any of the protagonists' electronic schedule. Failure to specify organizational position of any of the participants.
Financial	Justification of the project by a global yet simple investment plan made of realistic and accepted elements.	Creation and perpetuation of a financial model whose level of detail prohibits any future criticism and whose complexity will eventually lead to loss of validity .
Planning	Sub-projects and constraints are integrated into a realistic plan at the beginning of the project. The plan is monitored step-by-step using a lightweight professional tool.	Rigorous inspection of subcontracted or outsourced parts is neglected.
Innovation	Use of <i>emerging technologies</i> whose stable version will be available at release date in order to obtain maximum strategic efficiency.	Use of technologies or computing power <i>available at the start of the project</i> for administrative or contractual reasons.
Communication	The animation and report team uses modern means to dynamize communication, specify information and centralize information.	No distinction is made between workgroups and validation groups . Organization of <i>brainstorming</i> sessions is unsupervised and chaotic.
Method	Iterative incremental lifecycle: partial results are delivered in the shape of several validation Focuses, follow-up of a final prototype and finally a pilot site.	Traditional cascade lifecycle: a totally finalized system (supported by the hypothesis of total and perfect deployment) is aimed for before the technical and organizational risks are removed.

APPENDIX B: Cartesian rationalism (predictive) / Pragmatic empiricism (adaptive).

The radically different epistemological postulates on which predictive and adaptable methods base themselves are emphasized in this table.

Methods	Traditional or “Heavy”	New or “Agile”
Original Paradigm	Predictability	Adaptability
Foundation	Cartesian analysis	Pragmatic empiricism
Project Cycle	Cascade (no retroaction)	Incremental and Iterative (adaptable)
Removal of Risk	Description and documentation	Investigate - act – experiment
Reasoning	Discursive (premises-conclusions)	Systemic and heuristic
Underlying Vision	Isolate in order to structure a rigid universe	Execute to understand the interaction mechanisms
Frame of Thought	Reductionism and mechanistic hypotheses	Holistic vision of the various phenomenon (human resources, communication, environment)
Analytical Philosophy	Examines the nature of interactions	Examines the effects of interactions
Method Structure	Based on isolated stages and rigid specifications	Based on a simple and flexible phasing that takes project constraints into account
Research Axis	Structural analysis	Outcome analysis
Restrictions and Possibilities	Reduction of simple systems by analysis	Understanding of complex systems by their finalities
Leads to Systems with	Strong entropy	Strong retroactivity (cybernetic)
Outcome	Aims for an comprehensive solution	Aims for a “ satisfying output ”
Action Philosophy	Leads to a totally detailed and programmed schedule	Leads to an objective-based and flexible schedule
In Reality, leads to	Reproduction of the existing structure	Improvement of the systems
Verification	Theoretical comparison using tests at the end of the run	Continuous confrontation with reality (prototype)

Bibliography

Livre Blanc: Méthode Agiles état des lieux, www.BusinessInterActif.fr

[Paulk], *The Capability Model: Guidelines*, SEI, 1995

[P. Roques et F. Vallée], *UML en action*, Edition Eyrolles

[Philippe Kruchten], *Rational Unified Process*, Edition Eyrolles

[Vickoff (J-P)], *Piloter les projets informatiques de la nouvelle économie*, Editions d’Organisation, 2000