XP versus UP

A methodology comparison study work

Darius Damalakas
November 5, 2004

Roskilde Business College, 2004
3rd semester, Advanced Computer Science

1. Introduction

The purpose of this document is to analyze the two methodologies XP and UP using the framework proposed in “Information Systems development” [9]. The analysis is carried out for the studying purposes and thus the reader should be aware that the analysis will not be of the same depth in all aspects and may contain errors or inconsistencies. The author would be very thankful for comments and advices (e-mail: darius.damalakas@gmail.com).

The comparison will be made based on information from official XP web site [1] and practice gain from following the approaches described in Craig’s Larman book [10] “Applying UML and patterns" and ‘The New Methodology” [11] from Martin Flower.

Table of Contents

2. Philosophy

These sections present the philosophy the methodologies expose. In this context, 'philosophy' is regarded the same as in the book "Information Systems development" [9] it is a principle or a set of principles that underlie the methodology.

2.1 Paradigm

Two paradigms are considered relevant to this are the science [6] and the systems [7] paradigm.

It is difficult to strictly say to what paradigm methodologies should be put. We identify UP as being science paradigm while XP being systems paradigm. This partitioning is on the biggest part influenced by the overall perception of systems development for particular methodology. XP tries to build the system gradually. On the other hand, we believe, that UP is based on science paradigm because it tries more or less to build the system which is based on elaborate architecture (because UP is architecture-centric). Of course UP (as also XP) is based on an iterative approach.
The important issue is not the exact paradigm we assign, but the discussion our ideas might generate.

2.2 Objectives

This section distinguishes the objectives of the methodologies, or more strictly – it tries to answer the question 'could the use of methodology lead to the implementation of a purely organizational or non-IT solution?'

The objective of the XP methodology is not to come up with the non-IT or purely organizational solution, but quite the opposite - XP is designed to build IT-systems and do not address the analysis of the organization strategy and functions. UP is identified also as capable of producing only IT solution. This is based on the fact that both methodologies do not incorporate business analysis steps.

2.3 Domain

This section describes methodologies as being of the planning, organization, and strategy type or problem solving methodologies. This is related to the objectives of the methodology, but focuses on what aspects or domain the methodology seeks to address.

The primary focus is to solve a particular problem (to create a system which helps achieve some specific goals). That means the domain of interest is rather narrow and solely limits to creating an IT-solution and non-IT solution is not in the area of interest. XP does not address the general planning, organizational, and strategy of information and systems in the organization analysis. UP is also described as problem-solving methodology for the same reasons.

2.4 Target

This section is concerned with the target system to be developed, or more particularly whether the methodology is aimed at particular types of application, types of domain, size of system, environment and so on.

The XP is suited, as stated in XP web site [1], for small groups of people, usually varying between 2 and 12, though larger groups of 30 have also report success. It is not possible to use XP with huge staff.
XP enables to embrace change, so projects with highly unstable project requirements fits perfectly.

UP is considered to be general purpose methodology, although it is suggested they are not really helpful for simple, limited systems.

3. Model

This sections concerns with the model or models that the methodologies use. This can be described in terms of the type of the model, the levels of abstraction of the model and the orientation or focus of the model.

XP uses OO modeling techniques. It is stated that there is some written documentation for the software, more commonly interpreting requirements than documenting design.

In UP the basic models are also of object-orientation approach. However, the models are oriented not only for software design, but also business processes.

4. Techniques and tools

This section describes techniques and tools used by the methodologies

4.1 UP techniques

UP utilizes the following techniques in developing IT-solution

- Entity modeling
- Gannt charts

- OO techniques and UML (class diagram, use case diagram, interaction diagram, sequence and state chart diagrams, activity diagram)

4.2 UP tools

UP does not specifically address the tools to be used together with methodology.
Some individual tools that might be used with UP

- For diagramming:
  Microsoft Visio, MagicDraw.
- For project management
  Microsoft project
- Requirements:
  Microsoft office, Open Office, any XML technology supporting program

- Application packages (supports whole process)
  Rational Rose

4.3 XP techniques

XP does not specify techniques to be used except that it is object oriented methodology and appropriate tools and techniques should be used. Based on this we can state that it uses

- OO techniques and UML (class diagram, use case diagram, interaction diagram, sequence and state chart diagrams, activity diagram)
- Requirements are modeled with the use of CRC cards. CRC stands for Class, Responsibilities, and Collaboration. The cards are rarely filled up in great detail and in most cases only a class name is written at the top of the card.
- User stories [3]

4.4 XP tools

XP does not specify any specific tools to use except the very important area – testing.

- Unit Test Framework

Unit test framework is a development tool to facilitate creating and managing tests. “Your unit test framework can help you formalize requirements, clarify architecture, write code, debug code, integrate code, release, optimize, and of course test.”

5. Scope

Both methodologies are valued against the scope of software development life cycle stages they cover. This kind of comparison has major flaw – not all methodologies do follow a life cycle and follow other approaches. As is with our case, both methodologies use iterative approach.
The values for evaluating stage coverage ranges from 0 (means methodology does not cover the stage at all) to 3 (covers in detail).
The comparison is based on our quick investigation so the table should not be understood as concise and in most cases correct .

Phase/ Methodology	XP	UP
Strategy	0	1
Feasibility	0	1
Analysis	1	3
Logical Design	2	3
Physical Design	1	3
Programming	3	1
Testing	3	2
Implementation	1	3
Evaluation	0	2
Maintenance	0	0

The comparison clearly shows that the UP covers most of the areas rather in detail while XP focuses on programming and testing. This is because XP is a lightweight methodology [2]. XP focuses on delivering the software when it is needed and embraces changes gently through short-time boxed iterations (early feedback) and simplicity.

UP starts with inception phase looking at whether the project is feasible, and proceeds further with the project requirement gathering and analysis. XP does not have an implicit analysis phase – customer is available all the time and writes user stories [3]. User stories serve as a replacement of a big requirements document.

XP explicitly does not state logical or physical design stages – though spike solutions are created to help resolve various technical or system design problems. UP, on the contrary, has a design discipline, where design and data models are created with the Software Architecture document.

XP covers programming stage thoroughly. It has many, though simple rules and practices to guide the programmer of achieving simple code with extendable design. The coding rules are as follows: the customer is always available, code must be written to agreed standards, code the unit test first, all code is pair programmed, only one pair integrates code at a time, integrate often, use collective code ownership, leave optimization till last, no overtime working (instead change the project scope or timing). UP explicitly assumes that the OO programming language will be used (as is the methodology itself OO).

The last stage XP covers thoroughly is testing. Testing is one of the core roots of XP. It is even said “We would like to have more lines of test than we do of actual code” [5]. Both acceptance tests and unit tests are used. Unit test should be in most cases 100% automated. UP pays a great deal of attention to testing also, though not so “aggressively” as XP. In a testing discipline a test model is created with lots of test cases, and, presumably, automated tests.

6. Outputs

6.1 Introduction

The next sections will be investigations of what is actually produced in terms of deliverables at the end of each stage of methodology.

6.2 XP Outputs

- User Stories [3]
- CRC cards
- Release plans
- Iteration plan
- Unit tests
- Acceptance tests

6.3 UP Outputs

The outputs for UP will be categorized by the UP disciplines. Only the major artifacts (deliverables) will be named here and many simple non major artifacts will be skipped.

- Business modeling
  - Domain Model
- Requirements
  - Use-Case model
  - Vision
  - Supplementary Specification
  - Glossary
- Design
  - Design Model (Use Case Realizations)
  - Software architecture document
  - Data Model
- Implementation
  - Implementation model
- Project Management
  - Software Development plan (includes Project plan, iteration plans, etc.)
- Testing
  - Test Model
- Environment

- Development Case

7. Practice

7.1 Background

The background of the methodology broadly identifies its origins in terms of academic or commercial.

The UP has mainly an academic background, while XP is a commercial methodology. The new ways of developing software developed by Kent Beck were first tried in DaimlerChrysler and resulted as an XP methodology.

7.2 User Base

This topic describes how wide the methodology is used.

Not discussed in this document due to the lack of information

7.3 Participants

In this topic we discuss the participants involved, in some contexts these are referred to as 'actors' or 'stakeholders'. The questions answered here will be 'Who is supposed to use the methodology', 'What roles do they perform'. Skill levels are also addressed.

7.4 UP participants

A specialist team of professional systems analysts and designers perform the analysis and design aspects and professional programmers design the programs and write the code. The system is implemented by the analysts.
In order to learn UP, significant training and experience is needed.

7.5 XP participants

Participants are "ordinary programmers", who do not need a Ph.D. to use XP. Part of the team are also managers and customers as well. Through XP being an lightweight methodology [2], no intensive and expensive training is needed to wield the methodology.

8. Product

This section describes what is supplied when purchasing a methodology and at what cost.

8.1 UP products

The author of this document was not been able to find a freely available documentation.
Software to support the methodology (in activities such as configuration and change control, design and analysis) is advisable, though suitable open source software might be downloaded freely from web.

8.2 XP products

The descriptive documentation of the methodology is available online and is presented to the public with no charge. XP does not require any additional software to use it.

9. Conclusion

The two compared methodologies XP and UP are quite different from their background.

XP is designed to be a lightweight [2] and agile methodology. It is intended to be not used in projects with huge staff (suggested team size varies from 2 to 12 people). XP focuses on following practices which makes software development go faster and remove practices which make it move slower – it means documentation is made only when it is needed.

UP is quite the opposite – it is designed for big projects and with huge staff. A thorough process must be followed in order use UP methodology.

Both methodologies embrace change, though there are some differences in the two: UP embraces change through the use iterations and planning. XP uses also an iterative approach though with much smaller iterations (one of the reasons is because of the huge staff and the communication taking place there) thus enabling considerably fast feedbacks.

It is clear that the differences are primarily implied by the size of the staff. This allows XP to reduce documentation to the minimum and still keep the communication active and efficient.

Another underlying difference between these two methodologies is the way they treat system design. UP is based on an idea that as soon as the design is stabilized, coding planning and programming can take place. While this is a very good idea in civil engineering (where the cost of designing a bridge is 10% of the whole cost of the project), McConnell (

http://www.amazon.com/exec/obidos/ASIN/1556159005) suggests that for a large project, only 15% of the project is code and unit test, an almost perfect reversal of the bridge building ratios. He states that the design part takes not less than 50% of the work. XP addresses this (and all agile methodologies [8]) by focusing on coding thus limiting design (and documentation) activities.

UP is

- architecture centric
- Use-case driven
- Iterative
- Deals high risk issues in first place
- Object oriented

XP emphasizes:

- Team work
- Customer satisfaction by delivering software which is needed when it is needed
- Improvement of software project in four essential ways: communication, simplicity, feedback and courage
- Writing tests before actual code to be tested
- Implementing things you need right now, not the things you will need.
- Do documentation whenever it is needed
- The quality of the source code
- Small iterations
- Customer involvement

Whilst there are many differences, the authors personal opinion is that these two methodologies do not compete with each other – both stresses different parts of IS development and thus provide different approaches and practices.

10. References

10.1 [1] XP website

www.extremeprogramming.org

10.2 [2] Lightweight methodologies

An excerpt from XP website [1]: “A software methodology is a set of rules and practices used to create computer programs. A heavyweight methodology has many rules, practices, and documents. It requires discipline and time to follow correctly. A **lightweight methodology** has only a few rules and practices or ones which are easy to follow”.

10.3 [3] User stories

An excerpt from XP website [1]: “User stories serve the same purpose as use cases but are not the same. They are used to create time estimates <…>. They are also used instead of a large requirements document. User stories are written by the customers as things that the system needs to do for them. <…> They are in the format of about three sentences of text written by the customer in the customers terminology without techno-syntax”.

10.4 [4] Spike solutions

An excerpt from XP website [1]: “Create spike solutions to figure out answers to tough technical or design problems<…> Most spikes are not good enough to keep, so expect to throw it away. The goal is reducing the risk of a technical problem or increase the reliability of a user story's estimate”.

10.5 [5] The four project values

http://www.xprogramming.com/Practices/PracValues.html

10.6 [6] Science paradigm

Checkland (1981) describes science paradigm as consisting of reductionism, repeatability, and refutation: “we may reduce the complexity of the variety of the real world in experiments whose results are validated by their repeatability, and we may build knowledge by the refutation of hypothesis”.

10.7 [7] Systems paradigm

Checkland (1981) argues that human activity systems are systems which do not display characteristics of breaking down a problem into smaller parts does not break the whole system. Human activity systems has quite the opposite characteristics – emergent properties (i.e. the whole is greater than the sum of the parts) and perform differently as a whole. The systems paradigm concerns itself for the whole picture, the emergent properties, and interrelationships between parts of the whole.

10.8 [8] Agile methodology

Some time ago lightweight methodologies [2] were the term to be used instead of agile, whilst now these terms are used interchangeably. Agile methods “attempt a useful compromise between no process and too much process, providing just enough process to gain a reasonable payoff”, an excerpt from Martin Flower document “The new Methodology” [11].
“Agile methods are adaptive rather than predictive, Agile methods are people-oriented rather than process-oriented”.

10.9 [9] “Information Systems Development”

10.10 [10] “Applying UML and patterns”.

An introduction to Object-Oriented Analysis and Design and the Unified Process.
Craig Larman, 2nd edition © 2002

10.11 [11] The New Methodology

Martin Fowler. http://www.martinfowler.com