RAD Agile

How to adapt RAD to Agile Quality Process

- Matthieu GIROUX - www.liberlog.fr
- Member of www.caplibre.org

- Self programmer
- Installing and Customizing Web Sites
- Creating Management Softwares
- Created a Management framework

RAD Agile

Table of Contents

1.1) Defining Rapid Applications Development

Rapid Applications Development ou RAD

=

Creating software visualy to create faster

With RAD you gain time so you gain money.

The Very Rapid Applications Development permits to create your personal management software erasing a stage of software programming.

1.2) Bad exemples : API Toolkits

An API is a toolbox to program

The Management APIs need :

- To create some sources hardly manipulable
- To program and create on the same way

An engineer can transform The Management API to create the software management interface from some VRAD library files like LEONARDI.

1.3) How to create a library ?

Only one rule : Lesser copy-paste

Creating a framework steps :

- At the begining we create some functions units
- So we use and inherit some components
- We create some components packages
- We automate some components in a library
- So we open our library to other APIs
- The library is not used with any code

1.4) Rapid Applications Development (RAD)

- Creating visually a software
- To gain some time creating
- To create a customised software

Most of RAD tools not automate enought Enterprise Management Software.

The Very Rapid Application Development makes RAD better for Management Servers and other defined interfaces.

2.1) Very Rapid Applications Development

A software is composed by :

- What the user wants
- Computing technics

With or without RAD tools we :

- Mixed technical and user part
- Remade a software entirely

The user part must be kept.

2.2) Very Rapid Applications Development

VRAD permits to delete the stage of software programming from analysis.

So it permits to create the interface from passive files.

The VRAD is the next step of RAD and components programing.

2.3) Passive files vs classic RAD

With passive files we :

- Think funtionnalities and user part
- Define some themes easily
- Define what is made faster
- Define what is not on models
- Create some plugins for what is not made
- Know where we go

The RAD or VRAD permit to work on the front of the project to anticipate the future components.

2.4) Interesting of Very Rapid Applications Development

Le Very Rapid Development permits :

- Less errors be produced
- To create only the software analysis at the end
- To gain some time when creating
- To be self-sufficient of any framework
- That the programmer thing functionnalities
- To make better the quality of softwares

The RAD and VRAD are integrated and agility quality.

2.5) Creating an interface with passive files : VRAD

It is now possible to create a management interface from simple passive files.

A passive file containing user part is read and creates the interface from VRAD engine.

- GLADE GTK permits to create an interface not linked to date from passive files.
- LEONARDI permits to create a management interface from passive files.
- LIBERLOG owns a RAD framework adapting to VRAD.

2.6) VRAD Quality

With a VRAD engine :

- We gain in time and be more agile
- We facilitate the making of future softwares
- Test only the engine, not the created interface
- The analysis is the software
- The maintenance is centralised
- The programer goes to the fundamental

3.1) Iterative components programing ?

Programming funtionalities permits to :

- Keep that has been done
- Create a framework and share it if wanted
- Use really engineering and participate
- User create the software with analyst
- User be respected
- Make better
- Gain some time, some listening

3.2) Iterative components programing ?

The component permits to :

- Make one source for some technics
- Adapt an human micro-technic
- Gain some time in RAD
- Facilitate programmers working on RAD
- Anticipate on customer demands
- Delete the copy-paste
- Make better software

3.3) Iterative components programing – The team

Programming funtionalities needs :

- Two engineers unless for components
- Some analysts to exchange with customer
- Some weekly or monthly revues with customer
- A adaptive, comprehensive, listening management team
- A customer understood who know how is made his software to know how to make it better how he wants

3.4) Iterative components programing – Quality

Programming funtionalities needs :

- Some simple, modularly and generic concepts
- Some unit tests on components
- The will to make better components
- To communicate with models and interface
- A feedback to anticipate on components
- Some components created with two engineers

3.5) Iterative components programing – The software

Programmers or Software Analysts :

- Demand an daily exchange with engineers and customer
- Need to create software easily
- Made lesser on no programming
- Do not need unit tests

The software is visible with an interface, inspected, adapted. The begining interface is the future software.

3.6) Iterative components programing – The component

A component :

- Needs a logical structure
- Must be readable, organised, simple, revued
- Has lesser objectives
- Inherits of technics that must be done
- Needs to organise looking on future
- Needs some searching
- Needs to have the herited sources

3.7) Iterative components programing – The components

The components :

- Are created before software
- Are grouped on themas
- Are modeled easily
- Modelise some human micro-technics
- Permit to create the interface easily
- Permet to gain lot of time on RAD

3.8) Iterative components programing – Programmers

The software or components sources :

- Own to every programmers of the team
- Own at the begining the participating workers
- Need the names when modifying
- Need some revues for the good and bad points, growing and ungrowing
- Need to know where we go

4.1) Rapid Development vs Command Line

Example : Creating a simple form

A centralised code used with some copy-paste

- 3 days and it is not always finished

Same making with a RAD tool

- Analysing ½ day and ½ day of creating
- The component automatee some creating
- The form is usable with less tests

4.2) Passive files of VRAD vs RAD

Passive files :

- Permit to define the Job Part in their mind
- Can be created from analytic or data
- Can be slef-sufficient from all used framework
- Are defined a must be evolutive
- Permit to creating some more interfaces
- Permit to think functionality

The analysis of ½ day qui creates the software.

The analysis is always same as created software.

4.3) Publishing

It is time to ask ourselves some questions. To become a publisher and :

- Publish his self-made at a lot of customers permits to satisfy future customers
- Utiliser l'agilité renforce la confiance du client
- Le RAD et VRAD autorisent une élite avec des analystes orientés communication