ORDISOFTWARE ENGINEERING

AGILE CREATION OF OBJECT-ORIENTED APPLICATIONS

SOFTWARE DEVELOPMENT GUIDELINES

VERSION 0.1

OCTOBER 2016

OLIVIER ROGIER

WWW.ORDISOFTWARE.COM

GITHUB.COM/ORDISOFTWARE/GUIDELINES

TABLE OF CONTENTS

LICENSE

FOREWORD

This document presents some development guidelines to produce libre, personal, private, commercial and military software.

They are a description of how the author tries to work currently. They are considerations coming from the practice of manufacturing own and business applications. They are generally basic and obvious. They are not absolutes and not something imposed as are arithmetic and geometry. They are malleable and improvable like lots of things in this wise human world.

Each computer practitioner as everyone has its own rules forged teacher after teacher, talk after talk, book after book, line after line, launch after launch, pixel after pixel, click after click, error after error, reboot after reboot and update after update.

Everyone mostly thinks having the best system, since it comes from learnings those work. Everyone think to have the best for self and for doing some things, each time this fact is thanked. Everyone sometimes just wants most of the time do tomorrow a better work than yesterday.

A programming system does not escape to the difficulty to work with others that have different means to do some things while improving each without making war to impose one while saying everyone is free to justify the denial of the existence of numbers and letters that are the sole cause of the reality created by the chromosomic intelligence of this area whose the first rule of any legal activity is democratically applicable for each to not willingly harm anybody.

DISCLAIMER

The author is not very advanced in the way of writing in English.

He was not able to learn to speak and write English properly, and not so much better French. But he knows well things like `start-stop`, `if-then-else` and `call-return`.

He uses a lot Google's online search engine and translator with English⇔French articles of Wikipedia and Wiktionary, as well as Word's linguistics tools.

He hopes that the reader will not hold against him for his way to express, for the tone he uses and for his mistakes.

WHO THIS DOCUMENT IS FOR

This document is for anyone who wants to know how the author uses computing technologies and tools to fabricate programs.

It mainly refers to Agile thoughts and C#.NET but it can be used with most of systems. It covers mechanisms related to structuring items of a project and elements of an application. It does not concern the function of a special methodology.

It may be enhanced as it allows well creating legal and allowed software that works properly.

HOW THIS DOCUMENT IS ORGANIZED

This document is divided in twelve parts:

- « *License* » specifies the terms of use for this document.
- « *Foreword* » presents this document and the author.
- « *Project tools* » indicates the means used by the author.
- « *Project documents* » indicates the types of notes produced for a project.
- « *Folder structure* » indicates how are organized the file elements of a project.
- « *Naming convention* » specifies the standards used to write the source code.
- « *Code formatting* » specifies the standards used to render the source code.
- « *Comments usage* » specifies the standards used to describe the source code.
- « *UI design* » indicates some user interface practices currently used by the author.
- « *Using Git* » indicates some rules currently used by the author.
- « *Using GitHub* » indicates some parameters currently used by the author.
- « *Bibliography* » indicates some books related to computers.

CONVENTIONS USED IN THIS DOCUMENT

Phrases use mainly the « French double angle quotes ».

The "Typewriter identical double quotes" is used to distinguish a technical thing.

A section that is intended to be written in a future release indicates:

This section is undescribed yet.

The mention of a computing artifact looks like:

```
Menu / Submenu / Action
Filename.ext
www.domain.tld
```

RGB colors are noted as: #000000

Source code and lists looks like:

```
class Sample
{
}
- Value 1
```

```
- Value 2
- Value n
```

ABOUT THE AUTHOR

Olivier Rogier is a software craftsman mainly skilled in C#.NET and Delphi.

Such was the destiny of his abilities, of his will, of lived experience and of opportunities.

Despite constant unjustified and illegitimate oppressions and aggressions, he worked and works day and night every day when that is possible since his childhood for becoming and being a computer programmer, regardless of his results that were sometimes good and sometimes bad.

To learn more about him:

- Twitter: `twitter.com/ordisoftware`
- Facebook: `www.facebook.com/ordisoftware`
- LinkedIn: `www.linkedin.com/in/ordisoftware`
- Contact: `www.ordisoftware.com/contact`
- Profile: `www.ordisoftware.com/about/author`
- Thoughts: `www.ordisoftware.com/about/libre-software`
- Projects: `www.ordisoftware.com/projects`
- Blog: `www.ordisoftware.com/blog`
- Skills: `www.ordisoftware.com/business/skills`
- Achievements: `www.ordisoftware.com/business/history`
- Bibliography: `www.ordisoftware.com/business/bibliography`
- Service offer: `www.ordisoftware.com/services`

PROJECT TOOLS

The author currently uses the following tools to work on an assembled midrange PC.

OPERATING SYSTEM

Windows @ `www.microsoft.com/windows`

FILE MANAGER

Total Commander @ `www.ghisler.com`

BACKUP MANAGER

O&O DiskImage @ `www.oo-software.com`

Macrium Reflect @ `www.macrium.com`

FreeFileSync @ `www.freefilesync.org`

AutoVer @ `autover.codeplex.com`

SOURCE CONTROL

Git @ `git-scm.com`

GitHub @ `github.com`

Git Extensions @ `gitextensions.github.io`

TEXT EDITOR

Notepad2-mod @ `xhmikosr.io/notepad2-mod`

WORD PROCESSOR

Word @ `products.office.com/word`

PDF files are generated with the following options:

- *ISO 19005-1 compliant (PDF/A).*
- *Document structure tags for accessibility.*

SPREADSHEET

Excel @ `products.office.com/excel`

IMAGE PROCESSOR

XnView @ www.xnview.com

Axialis Icon Workshop @ www.axialis.com/iconworkshop

GIMP @ www.gimp.org

DIAGRAM DESIGNER

Software Ideas Modeler @ www.softwareideas.net

AGILE STORYBOARD

ZenHub @ www.zenhub.com

TIME TRACKING

AllNetic Working Time Tracker @ www.allnetic.com

INTEGRATED DEVELOPMENT ENVIRONMENT

Visual Studio @ www.visualstudio.com

NuSphere PhpED @ www.nusphere.com/products/phped.htm

VISUAL STUDIO EXTENSIONS

GitHub Extension for Visual Studio @ visualstudio.github.com

GitExtensions VSIX @ gitextensions.github.io

Codinion @ www.codinion.com

CodeMaid @ www.codemaid.net

Power Commands @ github.com/Microsoft/VS-PPT

Editor Guidelines @ github.com/pharring/EditorGuidelines

Solution Error Filter @ github.com/Microsoft/VS-PPT

File Icons @ github.com/madskristensen/FileIcons

Markdown Editor @ github.com/madskristensen/MarkdownEditor

Disable Solution Dynamic Nodes @ github.com/madskristensen/ToggleFeatures

Editor ToolTips @ github.com/Oceanware/TameVisualStudioEditorToolTips

Hide Suggestion @ marketplace.visualstudio.com/items?itemName=...

DATABASE

SQL Server @ `www.microsoft.com/sql-server`

SQLite @ `www.sqlite.org`

SQLite.NET @ `system.data.sqlite.org`

SQLite Expert @ `www.sqliteexpert.com`

DbSchema @ `www.dbschema.com`

COMMENTS GENERATOR

Atomineer Pro Documentation @ `www.atomineerutils.com`

DOCUMENTATION GENERATOR

Sandcastle Help File Builder @ `github.com/EWSoftware/SHFB`

SETUP PACKAGER

Inno Setup Installer @ `www.jrsoftware.org/isinfo.php`

PROJECT DOCUMENTS

GUIDELINES

Software guidelines are the rules that define how to create applications.

This one is the « Software Development Guidelines ».

It indicates technical and structural means used for the production.

The « Software Methodology Guidelines » indicates executive and functional processes.

Some others can be made like one for user interfaces or robots.

They both take part in the « Software Manufacturing Guidelines » package.

They are used to establish documents related to a specific project.

The development guidelines should be used with consistency within an organization.

The methodology guidelines may vary depending on the needs.

The nomenclature set forth below is currently used by the author.

GLOBAL SPECIFICATION

This is the document for the project goals with legal-contract and links:

- The « Project Charter » describes for who, why and how exists the product.
  - Who are the project owners and users?
  - What is a simple and sketchy description of domains, problems and goals?
  - What is the overall direction of the project?
  - What are the papers to produce?
  - What is the first estimation of means and timings?

OVERALL REALIZATION

This is the documents set for the project implementation:

- The « Application Reference » describes how is constructed and deployed the program to achieve the « Project charter ».
- The « Designer Diagram Reference » describes the organization of models and packages.
- The « Developer Data Processing Reference » describes physical schemas of classes with development help files and database tables with generation scripts as well as code algorithms and procedures-triggers.
- The « UI Reference » describes how are managed the interactions between users and computers by using keyboard, mouse, phone, etc. and screens, windows, controls, etc.

HIGH-LEVEL DESIGN FOR FUNCTIONS

This is the documents set for the project elaboration with analysis and conceptual modeling:

- The « Use cases Reference » describes stories and diagrams that describe actors and scenarios acting on activities of the domain.
- The « Communication Reference » describes how actors exist as scenarios.
- The « Activity Reference » describes the dynamic view of use cases.
- The « Sequence Reference » describes how activities exist as scenarios.

LOW-LEVEL DESIGN FOR STRUCTURES

This is the documents set for the project construction with technical and physical modeling:

- The « Deployment Reference » describes how to install the product.
- The « Component Reference » describes the combination of components.
- The « Class Reference » describes abstraction of things from the domain.
- The « Object Reference » describes how class instances exist as living entities.
- The « Collaboration Reference » in describes how objects interact.
- The « State Reference » describes the comportment of objects according to scenario.
- The « DB Reference » describes tables and schemas if necessary.

USER DOCUMENTATION

This is the documents set for the users:

- The « User Manual » is the traditional installation and usage guide.
- The « Quick Start Guide » is the conventional summary of the user manual.
- The « Troubleshooting Reference » indicates what to do if the program does not do the user want. It includes correcting the flow of operations in case of mistakes and actions to take in case of error message or even system crash as well as how to repairing data and executable files.

TIME TRACKING STAGES

Any methodology acts on height main scopes over any dichotomy and nomenclature:

- The « Management » is the time to supervise the project.
- The « Training » is the time to learn things like skills and domains.
- The « Data » is the time to study and defining things like with a database.
- The « Processing » is the time to handle things like those in a database and UI.
- The « Manual » is the time spent to give instructions to users like with a guide.
- The « Setup » is the time to deliver the application to users like with an executable.
- The « Publicity » is the time to advertise potential users like with a public message.
- The « Support » is the time to help users in difficulty like with assistance or recycling.

FOLDER STRUCTURE

This section is undescribed yet.

NAMING CONVENTION

FILES

This section is undescribed yet.

NAMESPACES

This section is undescribed yet.

TYPES

This section is undescribed yet.

ENUM

This section is undescribed yet.

CLASS

This section is undescribed yet.

INTERFACE

This section is undescribed yet.

VARIABLES

This section is undescribed yet.

INSTANCE

This section is undescribed yet.

LOCAL

This section is undescribed yet.

METHODS

This section is undescribed yet.

COMMENTS USAGE

FILES

This section is undescribed yet.

NAMESPACES

This section is undescribed yet.

TYPES

This section is undescribed yet.

VARIABLES

This section is undescribed yet.

INSTANCE

This section is undescribed yet.

METHOD

This section is undescribed yet.

ALGORITHMS

This section is undescribed yet.

CODE FORMATTING

INDENTATIONS

This section is undescribed yet.

LINES

This section is undescribed yet.

BRACKETS

This section is undescribed yet.

DECLARATIONS

This section is undescribed yet.

SIGNATURES

This section is undescribed yet.

STATEMENTS

This section is undescribed yet.

ALLOCATIONS

This section is undescribed yet.

UI DESIGN

CONSOLE

This section is undescribed yet.

FORMS

This section is undescribed yet.

WEB

This section is undescribed yet.

MOBILE

This section is undescribed yet.

TV

This section is undescribed yet.

USING GIT

NAMING REPOSITORY

<project-name>

Examples:

- Guidelines
- Core-Library

NAMING BRANCHING

<issue -type>/<issue-item-and-summary>/<issue-id>

Examples:

- method/text-chapters-dev/1
- test/ui-form-settings/45
- bug/install-icons-desktop/99

NAMING TAG

<version-or-stage>

Examples:

- v0.1
- v1.2.3
- v2.0.0-rc0

NAMING COMMIT

The seven rules from `chris.beams.io/posts/git-commit`:

- *Separate subject from body with a blank line.*
- *Limit the subject line to 50 characters.*
- *Capitalize the subject line.*
- *Do not end the subject line with a period.*
- *Use the imperative mood in the subject line.*
- *Wrap the body at 72 characters.*
- *Use the body to explain what and why vs how.*

Common commits actions are:

- Add, Rename, Remove, Delete.
- Update, Change, Fix, Move.
- Rework, Refactor, Clean.
- Initial commit, Merge, Release.

A domain can be specified by using an issue-item token:

```
ui: Fix the main form size
db: Add a script to create a table
manual: Update thefile.html
```

USING GITHUB

GITHUB PROJECTS VS ZENHUB

One GitHub project can be used as a storyboard for one or more use case diagrams.

While this not allows advanced management yet, the author uses ZenHub and Epics.

MILESTONES

Milestones allow identifying project big steps such as from inception to committed release.

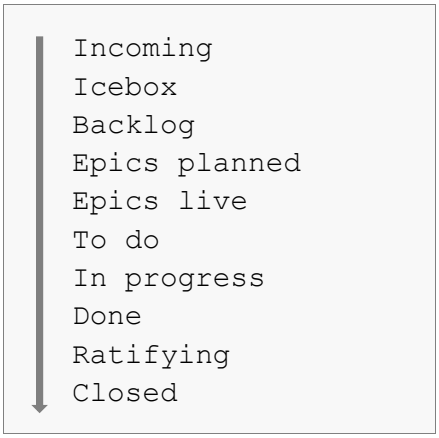
- *Learning* for issues related to *Activities for training and technology intelligence.*
- *Inception* for *Initialization of the project*.
- *Elaboration* for *Analysis and high-level design*.
- *Construction* for *Low-level design and implementation*.
- *Transition* for *Quality testing and releasing*.

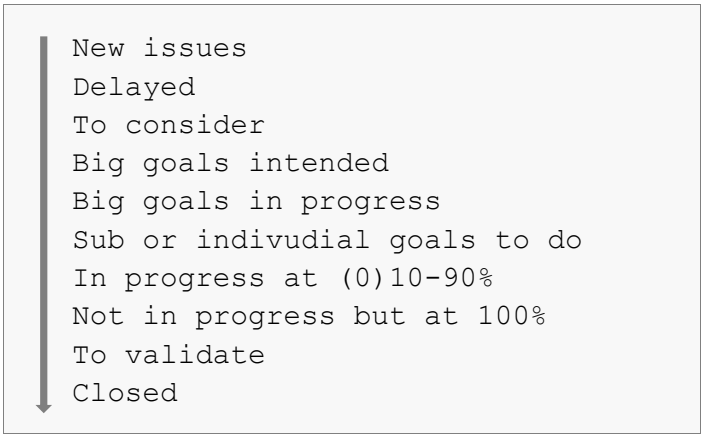
For simple or non-software projects such as this guide milestones can be:

- *Version 1*
- *Version 2*

PIPELINES

ZenHub Pipelines allow setting the stage of issues like on a Kanban workflow board.





On small projects Incoming, Icebox, Backlog, Done and Ratifying can be omitted, and Epics planned and live pipelines can be one while the in progress label indicates the difference.

Epics allow distinguishing Featured User Stories from Action User Stories.

Visitors of the project's page that are not connected to an account extended with ZenHub can't see this layering yet and only `in progress` label and `closed` issues can be used to distinguish them from other, so manual labels must be used in addition to moving cards.

ISSUE AS USER STORY

An Issue is used as a user story by indicating its description containing tasks in checklist.

```
[Issue #1] Prepare the repository

           As a developer,
           I want to establish the repository,
           so I can construct the software.

           ☐ Create the repository
           ☐ Setup the repository
           ☐ Specify the license
```

ISSUE ESTIMATE

Estimate field is used to define the issue complexity from `1` to `5` or to `10` for example, by considering knowledge, competence, technicity and range required by the issue.

Epic Issue estimation is usually not done because it is finished when all linked issues are finished and this value can be viewed on Issue details in the panel added by ZenHub.

Estimate time is out of scope of this document and falls under any appropriate methodology, but timings can be defined and adjusted using the burndown chart as the project progress.

ISSUES HIERARCHY

Visitors of the project's page that are not connected to an account extended with ZenHub can't see this design yet without check-listing sub-issues in the description.

FEATURED USER STORY AS A HIGH-GOAL THAT ENCAPSULATES LOW-GOALS

An Issue is used as a complex story containing references to other issues by using ZenHub Epic label.

It should contain a checklist of all sub-issues as high-tasks.

```
[Epic Issue #1]  Prepare the repository

                 ☐ Create the repository #2
                 ☐ Setup the repository #3
                 ☐ Specify the license #4
```

ACTION USER STORY AS LOW-GOAL TO ACHIEVE HIGH-GOAL

An Issue is used as a simple story acting as a card of what a user want by attaching it to an Epic Issue.

```
[Issue #2] Create the repository

          ☐ Add a repository
          ☐ Create a first branch

[Issue #3] Setup the repository

          ☐ Define Milestones
          ☐ Define Labels

[Issue #4] Specify the license

          ☐ Examine available licenses
          ☐ Choose a license
          ☐ Publish the LICENSE file
```

ISSUE LABELS

EPIC

ZenHub allows using special stories called Epic to gather other stories.

Color is Dark Blue #3E4B9E.

GROUP

Group defines the area concerned by the issue.

Color is Teal #006B75.

```
group: project (management)
group: training (learning)
group: analysis (requirements gathering)
group: design (modeling)
group: code (implementation)
group: manual (documentation and guide)
group: deploy (setup and migration)
group: user (assistance and communication)
```

TYPE

Type defines the gender of the issue.

Color is Green #0E8A16.

```
type: legal (license)
type: layout (organization and planning)
type: method (guideline)
type: admin (supervision)
type: feature (functionality)
type: improve (extend feature)
type: check (test, revision and validation)
type: bug (error)
type: feedback (reaction)
```

ITEM

Item defines the thing affected by the issue.

Color is Blue #1D76DB.

```
item: app (product and executable)
item: diagram (representation)
item: data (information)
item: source (code file)
item: install (packager)
item: text (writing)
item: tool (third party software)
item: ui (user interface)
item: ux (user experience)
item: other
```

PRIORITY

There is no medium priority since it is a loss of time to set and read it.

Thus it is easy to see the cards with low or high priority and others are ordinary.

```
prio: critical [Dark Red #900000]
prio: high [Red #CA2525]
prio: low [Dark Cyan #BFDADC]
```

IN PROGRESS

In progress defines an issue being solved and it is used in conjunction with some State label.

Color is Yellow #FFD700.

STATE

State indicates the progress of the work not towards the time but the remaining tasks.

Six points of a Gaussian curve are used to estimate the `In progress` pipeline.

This percentage is not about time because the tens first and last parts are generally longer while at the middle the things can be very fast:

- *When the task starts there is no really competence and no good visibility.*
  It has not started because it is taking its place to run on the racetrack.
  Things often seem to be simple and easy even for big task.
  It is not uncommon to spend a quarter of the time on this inception phase.
- *When the task comes to its end, there is a need to begin checking that all is really fine.*
  It is not running anymore because and it is shutting down on the racetrack.
  Things are more complex and more interactive even they look effortless.
  This transition phase can sometimes be more half the time.

This percentage may be reevaluated according to addition or cancellation of the complexity.

Color is Yellow #FFD700.

```
state: todo (selected) [Pale Green #C2E0C6]
state: delayed (deferred) [Gray #CACACA]
state: cancelled (abandoned) [Light Gray #EAEAEA]
state: wontfix (failed) [Dark Gray #707070]
state: 10% (work started)
state: 25%
state: 50%
state: 75%
state: 90% (almost completed)
state: 100% (done) [Light Yellow #FFF3B5]
```