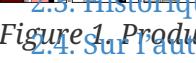
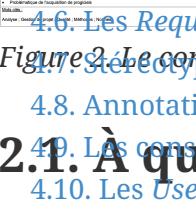
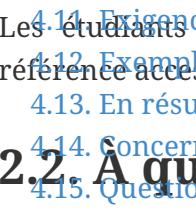
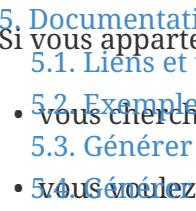
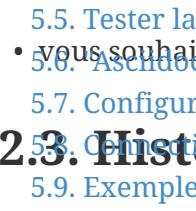


DUT-Info/S3/M3301 (MPA) : Support de cours

Jean-Michel Bruel

Table of Contents

1. Plan	1
2. Avant-propos	1
2.1. À qui est destiné ce document?	1
	1
2.2. À qui il n'est pas destiné?	1
	1
2.3. Historique	1
Figure 1. Produire des applications	2
	2
2.4. Sur l'autreur	2
2.5. Comment lire ce document?	2
Nous allons nous intéresser dans ce cours aux Méthodologies de la Production d'Application.	
2.6. Pourquoi parler de "document"?	3
2.7. Utilisation et autres mentions légales	3
Ce module, qui fait suite aux modules de programmation (M2103), conception (M2104) et d'IHM (M2105), est fortement corrélé au module de conception avancé (M3105):	3
3. Introduction	4
3.1. Points Nous suivons (comme tous les DUT informatique). le programme pédagogique national	4
NOTE	
3.2. Phase (PPN) (lien disponible ici)	5
3.3. Questions de révision	5
4. Les exigences	7
4.1. Fondements	8
	8
4.2. SysML	8
	8
4.3. L'organisation des Requirements	10
	10
4.4. Les Requirements properties	12
	12
4.5. Les Requirements links	12
	12
4.6. Les Requirements Diagrams	13
	13
4.7. Annotations des Requirements	13
4.8. Considerations sur la fragilité	14
	14
2.1. À qui est destiné ce document?	14
	14
4.10. Les Use Case Diagrams	14
	14
4.11. Exigences et tests	14
	14
4.12. Exemple complet	15
	15
4.13. En résumé	17
	17
4.14. Concernant le projet	18
	18
4.15. Questions de révision	18
5. Documentations et modèles générés	18
Si vous appartenez à une de ces catégories, ce document n'est pas pour vous :	
5.1. Liens et traçabilités	19
	19
5.2. Exemple concernant les exigences	19
	19
5.3. Générer un diagramme de classe à partir de Java commenté	21
	21
5.4. Générer de la perfection javadoc	22
	22
5.5. Tester la génération de doc	23
	23
5.6. AsciiDoc ne marche pas sur ma machine! ou UML™	29
	29
5.7. Configuration	32
	32
5.8. Connection entre Github et Travis	33
	33
5.9. Exemple de fichiers	34
	34
Ce document est tout nouveau (date de naissance 05/09/2014!), donc merci de votre indulgence	36
Liens utiles	37

Vous trouverez en référence (cf. Bibliographie) les ouvrages et autres documents utilisés.....	38
A propos de ce document....	39

2.4. Sur l'auteur

- Professeur à l'[Université de Toulouse](#), en poste à l'[IUT de Blagnac](#)
- Co-fondateur de l'association [SysML-France](#)
- Membre du comité éditorial de la revue [SoSyM](#)
- Membre du *Steering Committee* de la conférence ACM/IEEE [MODELS](#)
- Chef du département informatique de l'[IUT de Blagnac](#) 2009 à 2012
- Responsable de l'ancien module (Analyse et Conception des Systèmes d'Information)
- Marié à une merveilleuse femme, papa d'une merveilleuse fille

2.5. Comment lire ce document?

Ce document a été réalisé de manière à être lu de préférence dans sa version électronique (au format HTML ou PDF), ce qui permet de naviguer entre les références et les renvois interactivement, de consulter directement les documents référencés par une URL, etc.

WARNING

Si vous lisez la version papier de ce document, ces liens clickables ne vous servent à rien, et c'est votre punition pour avoir utilisé du papier au lieu du support électronique!

2.5.1. Conventions typographiques

J'ai utilisé un certain nombre de conventions personnelles pour rendre ce document le plus agréable à lire et le plus utile possible, grâce notamment à la puissance d'[AsciiDoc](#) :

- Les références bibliographiques présentées en fin de document (cf. [Bibliographie](#)).
- Les termes anglais (souvent incontournables) sont repérés en *italique*, non pas pour indiquer qu'il s'agit d'un mot anglais, mais pour indiquer au lecteur que nous employons volontairement ces termes (e.g., *Package*).

Le titre des figures indique (entre parenthèses) un M pour les figures issues de [Modelio](#), un MD pour les figures issues de [MagicDraw](#), un P pour les figures issues de [plantUML](#), un Py pour les figures issues de [Papyrus](#), un R pour les figures issues de [Rhapsody](#), un T pour les figures issues de [TOPCASED](#), un Y pour les figures issues de [yuml](#), et un UK pour les figures en anglais.

Pour les notes, conseils, avertissements, etc. voici la liste des pictogrammes utilisés :

NOTE Les notes comme celles-ci sont utilisées pour indiquer des éléments intéressant pour la majorité des lecteurs.

CAUTION Ces notes indiquent des points importants qui réclament votre attention.

TIP Celles-ci concernent en général des points de détail et permettent "d'aller plus loin".

Définition : Exemple (OMG UML v2.4.1, p. 152)

NOTE Ces notes concernent des définitions tirées de la spécification [UML™](#) et sont donc précisément référencées.

NOTE Modélisation UML incorrecte.

NOTE Modélisation UML partiellement correcte ou pouvant prêter à confusion.

NOTE Modélisation UML correcte.

2.6. Pourquoi parler de "document"?

Parce que j'ignore la version que vous êtes en train de lire. À partir de l'[original](#), plusieurs versions ont été générées grâce à [AsciiDoc](#) :

- Une version pour le web (Moodle) au format [html](#)
- Une version pour présentation en amphi au format [présentation](#)
- Une version pour impression au format [pdf](#)

2.7. Utilisation et autres mentions légales

Les images qui ne sont pas libres de droit contiennent un lien vers les sites où je les ai "empruntées".

N'hésitez pas à m'envoyer vos remarques en tout genre en m'écrivant [ici](#).

2.8. Organisation et généralités

Ce support de cours est destiné en priorité aux étudiants de l'[IUT de Blagnac](#).

Comme déjà indiqué, ce module est fortement corrélé au module de Conception et Programmation Objet Avancées (CPOA - M3105).

Il ne concerne que la partie modélisation du cours, les autres aspects étant couverts par Jean-Michel Inglebert. Il est prévu pour 2 cours d'1h30, complété par des mises en oeuvre en TD et en TP. Cette partie du cours va principalement porter sur [UML™](#), le langage universel de modélisation très utilisé en entreprise que vous maîtrisez déjà en partie.

3. Introduction

La matrice qui nous servira de "carte de base" pour placer les activités ou les modèles, sera celle-ci :

Table 1. La carte de base

	Requirements	Structure	Comportement	Transverse
Organisation				
Analyse				
Conception				
Implémentation				

Cette matrice permet de situer les différents éléments qui seront vus dans ce cours dans un cadre utile pour comparer ces éléments les uns aux autres. Je vous conseille de vous faire votre propre matrice. L'essentiel est de toujours bien se représenter les différents éléments qu'on aborde dans une carte mentale précise. Cela permet une meilleure mémorisation.

3.1. Points de vue

Dans un axe horizontal, j'ai différencié quatre grands points de vue :

Requirements

Les exigences et leur prises en compte sont un éléments critique pour le succès du développement de tout système. Sans explorer l'ensemble des activités d'ingénierie système (ce qui nécessiterait tout un volume du type de [Les exigences](#)) nous insisterons ce semestre sur cet aspect.

Structure

La description de l'architecture et des éléments constitutifs du système, avec les blocs, leurs relations, organisations internes, etc. constituera un point de vue important. C'est souvent la partie de modélisation qui pose le moins de problème aux débutants.

Comportement

Le comportement d'un système est du point de vue de l'utilisateur final beaucoup plus important que la structure elle-même. C'est la partie qu'il est la plus à même d'exprimer, de comprendre (vos modèles) et de valider.

Transverse

Un certains nombre de concepts sont transverses aux trois points de vue précédents. Il s'agira principalement de parler de cohérence ou de traçabilité entre les phases de développement ou entre les points de vue.

Ces différents points de vue ne doivent pas être confondus avec les différentes phases de

développement (cf. paragraphe suivant). Ils sont plutôt à rapprocher de la notion de préoccupation. C'est ainsi que j'ai choisi de distinguer trois points de vue qui se retrouvent souvent en modélisation : le point de vue des exigences qui permet de se focaliser sur les besoins des clients ; le point de vue structurel qui permet de se focaliser sur les différents composants du système ; et le point de vue comportemental qui permet de se focaliser sur le comportement du système. Ces trois points de vue n'étant pas indépendants les uns des autres, j'ai intégré un quatrième point de vue transversal.

3.2. Phase de développement

Dans un axe vertical, j'ai différencié quatre grandes phases du cycle de vie du développement :

Organisation

Une étape indépendante du type de cycle de développement envisagé (en V, agile, etc.) mais qui concerne la mise en place d'un cadre de travail qui permette un développement de qualité (outils, éditeurs, gestionnaire de version, de tâches, etc.).

NOTE | On pourrait rapprocher cette étape du "cycle 0" de [Scrum](#).

Analyse

Cette phase vise plutôt à examiner le domaine du problème. Elle se focalise sur les cahiers des charges et les exigences. L'analyse débouche sur un dossier d'analyse qui décrit les grandes lignes (cas d'utilisation, architecture principale) du système.

Conception

Cette phase vise plutôt à examiner le domaine de la solution. Elle débouche sur un dossier de conception qui décrit les détails conceptuels de la solution envisagée (structure détaillée, comportement, etc.)

Implémentation

Cette phase traite des développements finaux (construction ou approvisionnement en matériel, développement de codes, etc.).

Il s'agit ici classiquement des grandes étapes de développement d'un système. On pourrait être surpris par l'étape que j'ai appelé « organisation ». C'est une étape que je considère importante, particulièrement pour l'enseignement. Avant toute activité de modélisation ou de même de développement, il convient en effet de s'organiser en termes de choix d'outils, choix d'environnement, etc. Cette étape est souvent négligée par les étudiants. C'est pour cela que j'ai décidé de faire figurer cette étape de manière explicite. Bien sûr dans une organisation existante cette étape sera contrainte par les habitudes « maison ».

3.3. Questions de révision

1. Associez les diagrammes suivants avec leurs acronymes :

Diagramme de Paquetages		sd	Diagramme des Cas d'Utilisation
	dc	Diagramme de Séquences Système	
uc	Diagramme de Classes		pkg
Diagramme de Séquences		dss	

NOTE

2. Placez dans la matrice ci-dessous les différents diagrammes UML™ que vous connaissez déjà (sd, dc, uc, pkg, dss).

	Requirements	Structure	Comportement	Transverse
Organisation				
Analyse				
Conception				
Implémentation				

4. Les exigences

L'ingénierie des exigences est d'une importance capitale, surtout en Ingénierie Système. En général les exigences sont exprimées par des ingénieurs dédiés à cette activité. La complexité des systèmes modernes (embarqués, communicants, critiques, ...) rendent cruciale cette analyse.



Figure 3. 300 corps de métiers sont parfois présents sur un même chantier

Besoins, exigences : question de vocabulaire

La difficulté de l'emploi massif de l'anglais fait qu'il existe souvent une confusion entre les termes anglais et leurs traduction française. Nous précisons donc ici notre utilisation des termes :

Requirements

TIP

Exigences, c'est à dire une fonction ou une propriété que doit satisfaire le système considéré. Par nature une exigence doit pouvoir être **vérifiable**. En génie logiciel on parle plus classiquement des spécifications ("spec") pour parler des contraintes à respecter pour un système. Les ingénieurs systèmes ont depuis longtemps intégré le terme d'exigences comme traduction directe de *requirement*.

Besoins

Il s'agit des exigences du client. En **UML™** on va plus les retrouver dans les cas d'utilisation. Ils sont à l'origine des *requirements* tels que définis plus haut.

Il est important pour une exigence qu'elle ne soit pas ambiguë (contrairement au terme "en" dans la consigne exprimée par la maman dans l'illustration [ci-dessous](#) : "Ramène moi 1 bouteille de lait. S'il y a des oeufs, ramène m'en 6.").



Figure 4. Spécification ambiguë (taken from [here](#))

Dans le cadre de la matrice qui nous sert de plan, nous sommes ici :

	Requirements	Structure	Comportement	Transverse
Organisation				
Analyse				
Conception				
Implémentation				

4.1. Fondements

On abordera :

- L'organization des *Requirements*
- Les *Requirements properties*
- Les *Requirements links*
- Les *Requirements Diagrams*
- Les considérations sur la traçabilité
- Annotations des *Requirements*
- Les *Use Case Diagrams*

TIP L'ingénierie des exigences est une discipline à part entière et nous n'abordons ici que les aspects en lien avec la modélisation. Voir le livre de référence pour plus de détails ([\[Sommerville1997\]](#)) ou le guide de l'**AFIS** ([\[REQ2012\]](#)).

Il n'existe pas de diagramme **UML™** pour traiter des exigences en particulier. Nous allons donc nous servir dans cette partie d'un diagramme qui vient de **SysML™**.

4.2. SysML

4.2.1. Fiche d'identité

- Date de naissance non officielle : 2001!
- Première spécification adoptée à l'[OMG™](#) : 19 septembre 2007
- Version actuelle : [1.3](#) (12/06/2012)
- Paternité : [OMG™ / UML™ + INCOSE](#)
- Auteurs principaux :
 - Conrad Bock
 - Cris Kobryn
 - Sanford Friedenthal
- Logo officiel :

4.2.2. SysML, c'est...

Un ensemble de 9 types de diagrammes

- Diagrammes structuraux
 - Diagrammes de définition de blocs ([bdd](#))
 - Diagrammes internes de blocs ([ibd](#))
 - Diagrammes paramétriques ([par](#))
 - Diagrammes de packages ([pkg](#))
- Diagrammes comportementaux
 - Diagrammes de séquence ([sd](#))
 - Diagrammes d'activité ([act](#))
 - Diagrammes de cas d'utilisation ([uc](#))
 - Diagrammes d'états ([stm](#))
- Diagramme d'exigence ([req](#))

Un profil [UML™](#)

C'est à dire une **extension** de cette notation, un ensemble de nouveaux concepts et éléments qui sont définis à partir des éléments de base [d'UML™](#). Un exemple : le bloc [SysML™](#) n'est qu'une redéfinition de la classe [UML™](#).

Une notation

Une notation de plus en plus enseignée et connue et qui servira donc de plus en plus de **référence** à la modélisation des systèmes.

4.2.3. SysML, ce n'est pas...

Une méthode

En effet, contrairement à ce que beaucoup pensent en l'abordant, **SysML™** ne propose pas de démarche particulière de développement de système. C'est à la fois sa force (votre méthode existante pourra continuer à être utilisée) comme sa faiblesse car cette absence de guide méthodologique fait souvent défaut à son utilisation.

Un outil

Nous verrons en effet que **SysML™** ne fait que ce qu'on veut bien en faire. Comme tout langage il est limité dans son pouvoir d'expression, mais surtout il reste une simple notation qu'il convient d'utiliser avec des outils et des démarches associées.

Un raton laveur

C'est juste pour voir ceux qui suivent...

NOTE On ne dit pas "le SysML" mais tout simplement "SysML".

4.3. L'organisation des Requirements

Il ne s'agit pas ici de revenir sur les exigences elles-même, mais plutôt de voir comment **SysML™** permet de les exprimer, de les manipuler et surtout de les lier avec le reste du système.

4.3.1. Représentation de base

Un *Requirement* en **SysML™** n'est qu'un bloc particulier.

Définition : Requirements (OMG SysML v1.3, p. 139)

NOTE *A requirement specifies a capability or condition that must (or should) be satisfied... A requirement is defined as a stereotype of UML Class...*

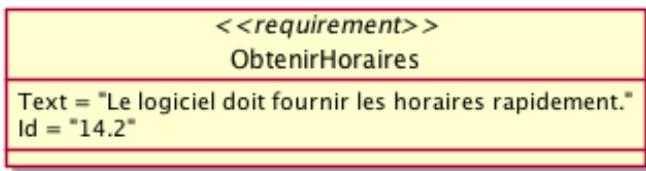


Figure 5. Un Requirement en SysML (P)

4.3.2. Différents types d'organisation

L'ingénierie des exigences aboutit généralement à une liste organisée d'exigences, que ce soit en terme

de fonctionnelles/non fonctionnelles, de prioritaires/secondaires, etc. Le principal support de SysML™ à cette organisation, outre la possibilité de les annoter (cf. section [Stéréotyper les exigences](#)), consiste à utiliser les *packages*.

Plusieurs types d'organisations sont possibles :

- Par niveau d'abstraction
 - Besoins généraux (en lien avec les *use cases* par exemple)
 - Besoins techniques (en lien avec les éléments de conception)
- Par point de vue
 - Besoins principaux (en lien avec les *use cases*)
 - Besoins spécifiques :
 - Fonctionnels
 - Marketing
 - Environnementaux
 - Business
 - ...
 - etc.

4.3.3. Tableaux de Requirements

Les *requirements* sont habituellement stockés dans des tableaux (feuilles excel le plus souvent!). Il est donc recommandé par la norme et possible dans de nombreux outils de représenter les exigences sous forme tabulaire.

Définition : Requirements Table (OMG SysML v1.3, p. 145)

NOTE *The tabular format is used to represent the requirements, their properties and relationships...*

ID	Name	Description	Type	Status	Priority
R-1	Req1	This requirement is a basic, general requirement that spans across multiple system components.	Functional	Pending Review	High
R-2	Req2	This requirement is a specific technical requirement related to the system's performance.	Technical	Approved	Medium
R-3	Req3	This requirement is a business requirement related to the system's functionality.	Business	Pending Review	Low
R-4	Req4	This requirement is a marketing requirement related to the system's user interface.	Marketing	Approved	Medium
R-5	Req5	This requirement is an environmental requirement related to the system's energy consumption.	Environmental	Pending Review	Low

Figure 6. Exemples tableau d'exigences (OMG SysML v1.3, p. 145)

La plupart des outils modernes permettent le passage entre outils classiques de gestion des exigences (comme [DOORS™](#)) et outils de modélisation [SysML™](#) (comme [Modelio](#), illustré ci-dessous).

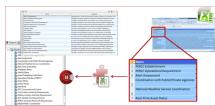


Figure 7. Import Modelio de tableau d'exigences (tiré de [Modelio2012])

4.4. Les Requirements properties

Il est possible d'indiquer un certain nombre de propriétés sur un *requirement* :

- *priority* (high, low, ...)
- *source* (stakeholder, law, technical, ...)
- *risk* (high, low, ...)
- *status* (proposed, aproved, ...)
- *verification method* (analysis, tests, ...)

Dans le cadre du module MPA nous ne retiendrons comme attribut d'un *requirement* que son identifiant et le texte le désignnat (les deux attributs obligatoire). La priorité sera donné par le client en terme de cycle (on traitera en premier les *requirements* prioritaires).

NOTE

Ainsi en [plantUML](#), une exigence ressemblera à ceci (cf. rendu [ici](#)):

```
class ObtenirHoraires <<requirement>> {
    Text = "Le logiciel doit fournir les horaires rapidement."
    Id = "14.2"
}
```

4.5. Les Requirements links

Les principales relations entre *requirement* sont :

Containment

Pour décrire la décomposition d'une exigence en plusieurs sous-exigences (→). Typiquement dès qu'une exigence est exprimée avec une conjonction "et" ("La voiture doit être rapide et économique").

Refinement

Pour décrire un ajout de précision ([\[refine\]](#)), comme par exemple une précision.

Derivation

Pour indiquer une différence de niveau d'abstraction ([\[deriveReqt\]](#)), par exemple entre un système et un de ses sous-systèmes.

TIP

Lorsqu'une exigence possède plusieurs cas [refine] qui pointent vers lui, on considère que ces différents cas sont des options possibles de raffinement (cf. [conventions]).



Figure 8. Exemples de relations entre exigences (M, UK)

TIP

Il existe ensuite les relations entre les besoins et les autres éléments de modélisation (les *block* ou les *class* principalement) comme [satisfy] ou [verify], mais nous les aborderons dans la partie *transverse*.



Figure 9. Relations liées aux requirements dans TOPCASED (T)

4.6. Les Requirements Diagrams

Voici un exemple de *req* un peu plus étayé, tiré de la norme (voir aussi [Exemples de rationale et problem](#) (tiré de SysML, UK)) :

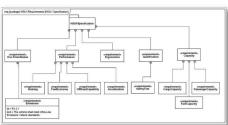


Figure 10. Exemples de composition d'exigences (tiré de *SysML*, UK)

4.7. Stéréotyper les Requirements

Tout comme pour n'importe quel bloc, il est possible de stéréotyper les *requirements*. Ceci permet de se définir ses propres priorités et classifications. Quelques exemples de stéréotypes utiles :

- [interfaceRequirement], [physicalRequirement], ...
- [FunctionalRequirement], [nonFunctionalRequirement]

4.8. Annotations des Requirements

Il est possible d'annoter les éléments de modélisation en précisant les raisons (*rationale*) ou les éventuels problèmes anticipés (*problem*).

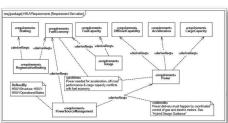


Figure 11. Exemples de rationale et problem (tiré de *SysML*, UK)

4.9. Les considérations sur la traçabilité

Une fois que les *requirements* ont été définis et organisés, il est utile de les lier au moins aux *use cases* (en utilisant **[refine]** par exemple) et aux éléments structurels (en utilisant **[satisfy]** par exemple), mais ceci sera abordé dans la partie **transverse**.

NOTE En général chaque *requirement* devrait être relié à au moins un *use case* (et vice-versa!).

4.10. Les Use Case Diagrams

Bien que nous traitions les cas d'utilisation dans la partie **comportement**, nous les abordons ici du fait de leur proximité avec les *requirements*.



Figure 12. Exemple de lien entre use case et requirements (T, UK)

Ce diagramme est celui que vous avez appris l'an dernier en **UMLTM**.



Figure 13. Exemple de diagramme des cas d'utilisation (B)

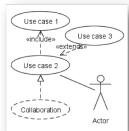


Figure 14. Autre exemple de diagrammes des cas d'utilisation (B)

TIP

Un acteur représente un rôle joué par un utilisateur humain. Il faut donc plutôt raisonner sur les rôles que sur les personnes elles-mêmes pour identifier les acteurs.

4.11. Exigences et tests

4.11.1. Principes

Pour ce qui ce concerne ce module nous allons nous contenter de maintenir des matrices croisant les exigences d'un côté et les tests de l'autre.

Par exemple :

Dans la réalité, les entreprises industrialisent le processus de vérification des exigences en utilisant des

outils adaptés. Illustration tirée de [TestsIndustriels2009] :

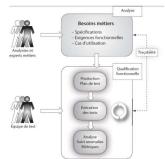


Figure 16. Le cycle de la qualification fonctionnelle en lien avec les besoins métiers

4.12. Exemple complet

En prenant un exemple tiré d'un exercice que vous avez traité l'an dernier voici un exemple cohérent :

<http://www.iict.ch/Tcom/Cours/OOP/Livre/UML21.pdf>

4.12.1. Le texte du cahier des charges

Le texte complet de l'exemple ne précise pas le cahier des charges de l'Autoradio (AR), considérant que tout le monde sait ce que c'est!

Rédigeons tout de même quelques extraits (numérotés) de texte possible :

1. L'AR est un dispositif qui permet d'écouter la radio de manière confortable et interactive.
2. L'AR doit être capable de mémoriser un certain nombre de station différentes.
3. L'Utilisateur de l'AR doit pouvoir choisir sa station parmi un choix donné.
4. L'Utilisateur de l'AR doit pouvoir régler le niveau sonore.
5. L'Utilisateur de l'AR doit pouvoir chercher une station en "balayant" les ondes FM.
6. ...

4.12.2. Expression des exigences

Nous pouvons, en analysant ce cahier des charges, déduire un certain nombre d'exigences. Nous les écrivons ici sous forme tabulaire, et en utilisant le langage [Gherkin](#).

NOTE Concernant [Gherkin](#), pour ceux qui veulent aller plus loin, il est possible d'automatiser la génération des tests à partir de ce genre de langage. C'est ce que fait [Cucumber](#) pour [Ruby](#).

Version tabulaire :



Figure 17. Exemple de liste d'exigences

Exemple de **version textuelle** formattée (fichier source [ici](#)):

```
#encoding: utf-8
Feature: Scénario simple d'utilisation de l'AutoRadio (AR)
  In order to verify que le son marche
  As an utilisateur lambda
  I should be able to execute ces scénarios et constater les effets
```

Scenario: Augmenter le son

```
Given un AR avec le son    0
When Je presse le bouton "Volume +"
Then Le son passe    1
And Je commence    entendre la radio
```

4.12.3. Plan de test

Créer un plan de test consiste à prévoir l'ensemble des tests à l'avance de manière à prévoir la **couverture** de ces tests.

A	B	C	D	E
Id	Nom	Test1	Test2	Test3
14	ContrôleSon	x		
15	StationMémorisée		x	
16	ContrôleStation		x	x

Figure 18. Exemple de plan de test simplifié

4.12.4. Analyse et la conception

Dans un cycle classique ("en V" par exemple), les modèles sont réalisés avant l'implémentation (codage).



Dans un cycle Agile, chaque cycle possèdera ses modèles, eux aussi versionnés, qui eux aussi évolueront en même temps que le code.

4.12.5. Lien et traçabilité

Plus encore que dans les méthodes classiques, il conviendra de vérifier que code et modèles sont bien cohérents. On pourra donc :

- générer les codes à partir des modèles
- générer les modèles à partir des codes (cf. [Exemple de code Java commenté pour la génération automatique de diagrammes plantUML](#))
- utiliser des outils intégrés comme [eclipse](#)

avoir un plan systématique de révision code/modèle

- ...

Exemple de code Java commenté pour la génération automatique de diagrammes plantUML

```
package demo;

class Controller {}
class EmbeddedAgent {}
class PowerManager {}

/**
 * @extends Controller
 * @extends EmbeddedAgent
 * @navassoc - - 1..* PowerManager
 * @note this is a note
 */
class SetTopController implements URLStreamHandler {
    public String name;

    int authorizationLevel;
    void startUp() {}
    void shutDown() {}
    void connect() {}
}

/** @depend - friend - SetTopController */
class ChannelIterator {}

interface URLStreamHandler {
    void OpenConnection();
    void parseURL();
    void setURL();
    void toExternalForm();
}
```



Figure 19. Exemple de diagramme généré

4.13. En résumé

Les exigences sont très importantes en ingénierie logiciel, du fait de la multiplication des sous-systèmes et donc des intermédiaires (fournisseurs, sous-traitants, etc.) avec qui les aspects contractuels seront souvent basés sur ces exigences. Il n'est donc pas étonnant qu'un diagramme et des

mécanismes dédiés aient été prévus en SysML™.

Table 2. Déclinaison des Exigences

	Requirements	Structure	Comportement	Transverse
Organisation	-, <<deriveReqt>>			
Analyse	[satisfy], [refine]	[satisfy] entre reqs et UC	[refine]	
Conception	\<<allocate>>			
Implémentation	\<<satisfy>>, \<<verify>>			

En terme de démarche, il est classique d'avoir de nombreux aller-retour entre la modélisation des exigences et la modélisation du système lui-même (cf. [Exemple de démarche \(SYSMOD Zigzag pattern\)](#)).

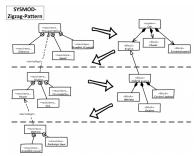


Figure 20. Exemple de démarche (SYSMOD Zigzag pattern)

4.14. Concernant le projet

- Je serai votre "référent" sur les aspects **modélisation**
- Jean-Michel Inglebert sera votre référent **méthodes agiles**
- Olivier Roques sera votre référent **IHM et java**
- André Péninou sera votre référent **client** et exigences

4.15. Questions de révision

1. Quelles sont les différences entre **besoins** et **exigences** ?

NOTE 2. Quelles sont les différences entre un **backlog de produit** et une **exigences**?

3. En quoi les cas d'utilisation sont-ils complémentaires des exigences?

5. Documentations et modèles générés

5.1. Liens et traçabilités

Plus encore que dans les méthodes classiques, il conviendra de vérifier que code et modèles sont bien cohérents. On pourra donc :

- générer les codes à partir des modèles
- générer les modèles à partir des codes (cf. [Exemple de code Java commenté pour la génération automatique de diagrammes plantUML](#))
- utiliser des outils intégrés comme [eclipse](#)
- avoir un plan systématique de révision code/modèle
- ...

5.2. Exemple concernant les exigences

On part des exigences (en Scrum, le Product Backlog) :

The screenshot shows a Redmine interface for a project titled "Projets DUT/INFO/S3/MPA > MPA Groupe 1B1 > Carnet maître". The main view displays a table of user stories categorized under "Sprint_0" and "Sprint_1". A specific user story from "Sprint_0" is highlighted with a red box:

Identifiant	Titre	Début	Fin	Statut	Etat	Créé par	Mise à jour
2268	Mise en place de l'infrastructure d'hébergement.	2014-09-15	2014-09-19	Nouveau	En cours	Administrateur	2014-09-15 10:45
2269	Intégration des fonctionnalités d'affichage et de gestion des données.	2014-09-22	2014-10-06	En cours	En cours	Administrateur	2014-09-15 10:45
2270	Documentations et de la documentation utilisante sous ASCIIDOC			En cours	En cours	Administrateur	2014-09-15 10:45
2271	Développement de la documentation hébergée sous ASCIIDOC			En cours	En cours	Administrateur	2014-09-15 10:45
2272	Mise en place du schéma de travail pour le Scrum			Terminé	Terminé	Administrateur	2014-09-15 10:45
2273	Test des fonctions d'affichage et de gestion des données.			Nouveau	En cours	Administrateur	2014-09-15 10:45

A red box highlights the first row of the table, specifically the "Titre" column for the story "Mise en place de l'infrastructure d'hébergement". The right side of the interface shows a sidebar titled "Carnet De Produit" with a list of completed iterations:

- Iterations complétées fermées (10):
 - 2268 En tant qu'administrateur, je veux pouvoir afficher la liste des projets du fichier sujet. En cours 0,0
 - 2269 En tant qu'administrateur, je veux pouvoir afficher la liste des types du fichier sujet. En cours 0,0
 - 2270 En tant qu'administrateur, je veux pouvoir afficher la liste des types du fichier sujet. En cours 0,0
 - 2271 En tant qu'administrateur, je veux pouvoir afficher la liste des échéances ou des inter. En cours 0,0
 - 2272 En tant qu'administrateur, je veux pouvoir afficher la liste des projets du fichier proj. En cours 0,0
 - 2273 En tant qu'administrateur, je veux pouvoir afficher deux THIN la liste des intervenants. En cours 0,0
 - 2274 En tant qu'administrateur, je veux pouvoir afficher la liste des intervenants pour ajouter le. En cours 0,0
 - 2275 En tant qu'administrateur, je veux pouvoir afficher la liste des projets pour ajouter un. En cours 0,0
 - 2276 En tant qu'administrateur, je veux pouvoir afficher la liste des projets pour ajouter un. En cours 0,0
 - 2277 En tant qu'administrateur, je veux pouvoir afficher la liste des groupes indiquant le. En cours 0,0
 - 2278 En tant qu'administrateur, je veux pouvoir afficher la liste des échéances pour ajouter. En cours 0,0
 - 2279 En tant qu'administrateur, je veux pouvoir afficher la liste des intervenants pour ajouter. En cours 0,0
 - 2280 En tant qu'administrateur, je veux pouvoir afficher la liste des intervenants pour ajouter. En cours 0,0
 - 2281 En tant qu'administrateur, je veux pouvoir afficher la liste des intervenants pour ajouter. En cours 0,0
 - 2282 En tant qu'administrateur, je veux pouvoir afficher la liste des intervenants pour ajouter le. En cours 0,0
 - 2283 En tant qu'administrateur, je veux pouvoir afficher la liste des projets pour ajouter un. En cours 0,0
 - 2284 En tant qu'administrateur, je veux pouvoir afficher la liste des projets pour ajouter un. En cours 0,0
 - 2285 En tant qu'administrateur, je veux pouvoir afficher la liste des projets pour ajouter un. Nouveau 0,0

Figure 21. Exemple de "Product Backlog"

On les liste dans Redmine :

Accès à la page Projets Administration Aide Général et test sur jeu Non compléter Déconnexion

Projets DUTINFO/S3/MPA » MPA Groupe 1B1

Recherche : MPA Groupe 1B1

Aperçu Activité Roadmap Carnets Tableau des tâches Releases Demandes Nouvelle demande Gantt Calendrier Annances Documents Wiki Fichiers Dépot

Demandes

Filtrer : Statut : Version cible : Type de carnet : Release : Scénarios : Ajouter le filtre

Voir toutes les demandes Historique Calendrier Gantt

Intérêts Sprint_0 Sprint_1

MPA Groupe 1B1

Carnet de produit

Appliquer Effacer Sauvegarder

#	Tracker	Statut	Priorité	Sujet	Assigné à	Mis à jour	Position
2205	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des intervenants du fichier intervenants2014_2015.csv dans l'IHM		22/09/2014 08:58	126187
2206	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des sujets du fichier projets2014_2015.csv dans l'IHM		22/09/2014 08:58	126199
2210	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des étudiants du fichier etudiants2014_2015.csv dans l'IHM, en supprimant certains puis l'enregistrer dans un fichier .csv de mon choix		22/09/2014 08:58	126202
2208	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des projets du fichier projets2014_2015.csv dans l'IHM		22/09/2014 08:58	126205
2221	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher dans l'IHM la liste des intervenants indiquée pour chaque projet, je suis, le rôle, le groupe et le projet		22/09/2014 08:58	126211
2220	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des étudiants du fichier etudiants2014_2015.csv dans l'IHM, en supprimant certains puis l'enregistrer dans un fichier .csv de mon choix		22/09/2014 08:58	126217
2213	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des sujets puis ajouter un nouveau sujet et sauvegarder la liste des projets dans un fichier .csv de mon choix		22/09/2014 08:58	126230
2215	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des projets puis ajouter un nouveau projet et sauvegarder la liste des projets dans un fichier .csv de mon choix		22/09/2014 08:58	126256
2219	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des groupes puis ajouter un nouveau groupe et sauvegarder la liste des groupes dans un fichier .csv de mon choix		22/09/2014 08:58	126257
2211	Story	En cours	Normal	En tant qu'administrateur, je veux pouvoir afficher la liste des étudiants puis ajouter un nouvel étudiant et sauvegarder la liste des étudiants dans un fichier .csv de mon choix		22/09/2014 08:58	126258
2217	Story	En cours	Normal	En tant qu'administrateur, une fois la liste des étudiants ou des intervenants ou des sujets ou des projets affichée dans l'IHM, je veux		22/09/2014 08:58	126259

Figure 22. La liste précise (redmine)

On les exporte dans un format manipulable (exemple .csv) :

Options Appliquer Effacer Sauvegarder

MPA Groupe 1B1

Carnet de produit

2205 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des intervenants du fichier intervenants2014_2015.csv dans l'IHM 22/09/2014 08:58 126187

2206 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des sujets du fichier projets2014_2015.csv dans l'IHM 22/09/2014 08:58 126199

2210 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des étudiants du fichier etudiants2014_2015.csv dans l'IHM, en supprimant certains puis l'enregistrer dans un fichier .csv de mon choix 22/09/2014 08:58 126202

2208 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des projets du fichier projets2014_2015.csv dans l'IHM 22/09/2014 08:58 126205

2221 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher dans l'IHM la liste des intervenants indiquée pour chaque projet, je suis, le rôle, le groupe et le projet 22/09/2014 08:58 126211

2220 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des étudiants du fichier etudiants2014_2015.csv dans l'IHM, en supprimant certains puis l'enregistrer dans un fichier .csv de mon choix 22/09/2014 08:58 126217

2213 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des sujets puis ajouter un nouveau sujet et sauvegarder la liste des projets dans un fichier .csv de mon choix 22/09/2014 08:58 126230

2215 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des projets puis ajouter un nouveau projet et sauvegarder la liste des projets dans un fichier .csv de mon choix 22/09/2014 08:58 126256

2219 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des groupes puis ajouter un nouveau groupe et sauvegarder la liste des groupes dans un fichier .csv de mon choix 22/09/2014 08:58 126257

2211 Story En cours Normal En tant qu'administrateur, je veux pouvoir afficher la liste des étudiants puis ajouter un nouvel étudiant et sauvegarder la liste des étudiants dans un fichier .csv de mon choix 22/09/2014 08:58 126258

2217 Story En cours Normal En tant qu'administrateur, une fois la liste des étudiants ou des intervenants ou des sujets ou des projets affichée dans l'IHM, je veux

Options d'exportation CSV * 22/09/2014 08:58 126259

Formats disponibles : PDF CSV XLS

Colonnes sélectionnées Toutes les colonnes Description Exporter Annuler

1(14/14)

Figure 23. Export des scénarios du "Product Backlog"

On les passe dans une moulinette maison pour obtenir du PlantUML :

Exemple de conversion en PlantUML

```
@startuml

class Req_2205 <<Requirements>> {
Id = 2205
Text = "En tant qu'administrateur, je veux pouvoir afficher la liste des intervenants du fichier intervenants2014_2015.csv dans l'IHM"
}

class Req_2213 <<Requirements>> {
Id = 2213
Text = "En tant qu'administrateur, je veux pouvoir afficher la liste des sujets puis ajouter un nouveau sujet et sauvegarder la liste des projets dans un fichier .csv de mon choix"
}

class Req_2220 <<Requirements>> {
Id = 2220
Text = "En tant qu'administrateur, je veux pouvoir afficher la liste des étudiants indiquant le groupe, le sujet, le projet et l'ensemble des intervenants du projet dans l'IHM"
}

...
@enduml
```

Le fichier complet est disponible [ici](#). Ce qui donne le rendu ci-dessous :



Figure 24. Diagramme des exigences (SysML) résultant

NOTE

Les exemples ne sont pas des corrections du projet en cours, mais sont donnés ici uniquement à titre d'illustration.

On réalise qu'il serait nécessaire de disposer d'une API digne de ce nom pour aller directement chercher les exigences dans [redmine](#).

5.3. Générer un diagramme de classe à partir de Java commenté

Exemple de code Java commenté pour la génération automatique de diagrammes plantUML

```
package demo;

class Controller {}
class EmbeddedAgent {}
class PowerManager {}

/**
 * @extends Controller
 * @extends EmbeddedAgent
 * @navassoc - - 1..* PowerManager
 * @note this is a note
 */
class SetTopController implements URLStreamHandler {
    public String name;

    int authorizationLevel;
    void startUp() {}
    void shutDown() {}
    void connect() {}
}

/** @depend - friend - SetTopController */
class ChannelIterator {}

interface URLStreamHandler {
    void OpenConnection();
    void parseURL();
    void setURL();
    void toExternalForm();
}
```



Figure 25. Exemple de diagramme généré

5.4. Générer de l'asciidoc par javadoc

Pour aller un cran plus loin, vous pouvez même envisager de d'intégrer de l'[Asciidoc](#) au moment de générer la documentation avec [Javadoc](#)!



Figure 26. Exemple de diagramme généré

TIP Pour plus d'info : [Asciidoclet](#)

On peut même utiliser la puissance des variables. Par exemple les commentaires Java suivants seront mis à jour en ajoutant les options suivantes au doclet :

```
-a product-name=OPTI -a version=1.0
```

```
/**  
 * {product-name} will change your life!  
 * @version {version}  
 */
```

TIP Cette astuce est aussi valable en [AsciiDoc](#)!

5.5. Tester la génération de doc

À partir du moment où la documentation est générée, on peut tester le programme qui la génère, comme n'importe quel programme.

Dans les exemples qui suivent on peut :

- tester l'existence des fichiers qui sont inclus (e.g., `include::foo.txt[]`)
- tester la conformité avec les règles définies dans le sujet
- tester la conformité aux bonnes pratiques

5.5.1. Les fichiers inclus existent-ils bien?

En utilisant un simple script [Ruby](#) (facilement intégrable en CI avec [Travis](#) par exemple) voici ce qu'on obtient :

```
bruel@pc1:~/dev/Papyrus4Education$ ruby checkImages.rb
asciidoc source : main.html
"opening main.html"
OK /Users/bruel/dev/Papyrus4Education/images/1.0.0/StylesheetsDefinition.png
OK /Users/bruel/dev/Papyrus4Education/images/1.0.0/without.png
OK /Users/bruel/dev/Papyrus4Education/images/1.0.0/with.png
OK /Users/bruel/dev/Papyrus4Education/images/word.png
OK /Users/bruel/dev/images/icons/note.png
OK /Users/bruel/dev/images/icons/note.png
OK /Users/bruel/dev/images/icons/note.png
NOK /Users/bruel/dev/Papyrus4Education/images/88x32.png
bruel@pc1:~/dev/Papyrus4Education$
```

Figure 27. Testing if all images are correct

Voici le détail du [script](#) :

```

# CheckImage.rb
# 2014 -- JMB (initial code from JM Inglebert)
#-----
re = Regexp.new("\.html$") # asciidoc result file (easier!)

dir = Dir.new('.')
dir.each {|fn|
  if ( fn =~ re ) then
    print "asciidoc source : " + fn + "\n"
    paths = []
    # find all image: or image:: asciidoc macros
    File.open(fn) { |f|
      p 'opening' << fn
      content = f.read
      paths = content.scan(/ file : nom du fichier doc
# => membre : le nom d'un membre du groupe au hasard
# => groupe : le num ro du groupe
# => sprint : le num ro du sprint
#
unless ARGV.length == 4
  puts "Usage: ruby checkDoc.rb file membre groupe sprint\n"
  exit
end

```

#La documentation utilisateur devra fournir les informations suivantes :

```

file = ARGV[0]
membre = Regexp.new(ARGV[1])
groupe = Regexp.new(ARGV[2])
sprint = Regexp.new(ARGV[3])

f = File.open(file)
content = f.read

# liste des membres de l' quipe

res = content.scan(membre)
print "Membre : " + (res != [] ? " OK " : " NOK ") + "\n"

# num ro du groupe

res = content.scan(groupe)
print "Groupe : " + (res != [] ? " OK " : " NOK ") + "\n"

# Universit Toulouse 2

res = content.scan(Regexp.new("Universit Toulouse 2"))
print "UT2 : " + (res != [] ? " OK " : " NOK ") + "\n"

# IUT de Blagnac

res = content.scan(Regexp.new("IUT de Blagnac"))
print "IUT : " + (res != [] ? " OK " : " NOK ") + "\n"

```

```

# DUT INFO S3/Module MPA

res = content.scan(Regexp.new("DUT INFO S3/Module MPA"))
print "MPA : " + (res != [] ? " OK " : " NOK ") + "\n"

# le nom du projet : OPTI

res = content.scan(Regexp.new("OPTI"))
print "OPTI : " + (res != [] ? " OK " : " NOK ") + "\n"

# le SPRINT concern

res = content.scan(Regexp.new(sprint))
print "SPRINT : " + (res != [] ? " OK " : " NOK ") + "\n"

# La documentation utilisateur :

# contiendra une section présentant le backlog de produit du projet OPTI connu en début de SPRINT

# décrira toutes les fonctionnalités disponibles à la fin du SPRINT

```

```

bruel@MacBookPro-de-JMB:~/dev/mpa/coursjmb/test$ ruby checkDoc.rb ./cours.html JMB 1A1 1
Membre : OK
Groupe : OK
UT2 : NOK
IUT : OK
MPA : NOK
OPTI : OK
SPRINT : OK
bruel@MacBookPro-de-JMB:~/dev/mpa/coursjmb/test$ ||

```

Figure 28. Test de contenu de doc

5.5.4. Règles sur les bonnes pratiques



HOME GETTING STARTED DOWNLOADS DOCUMENTATION SUPPORT GETTING I
ABOUT MORE ▾



Figure 29. Un exemple de la vie réelle : la documentation [http://www.eclipse.org/papyrus/\[Papyrus\]](http://www.eclipse.org/papyrus/[Papyrus])

Voici un exemple qui vérifie que la documentation est conforme aux règles [Eclipse Doc Style Guide](#).

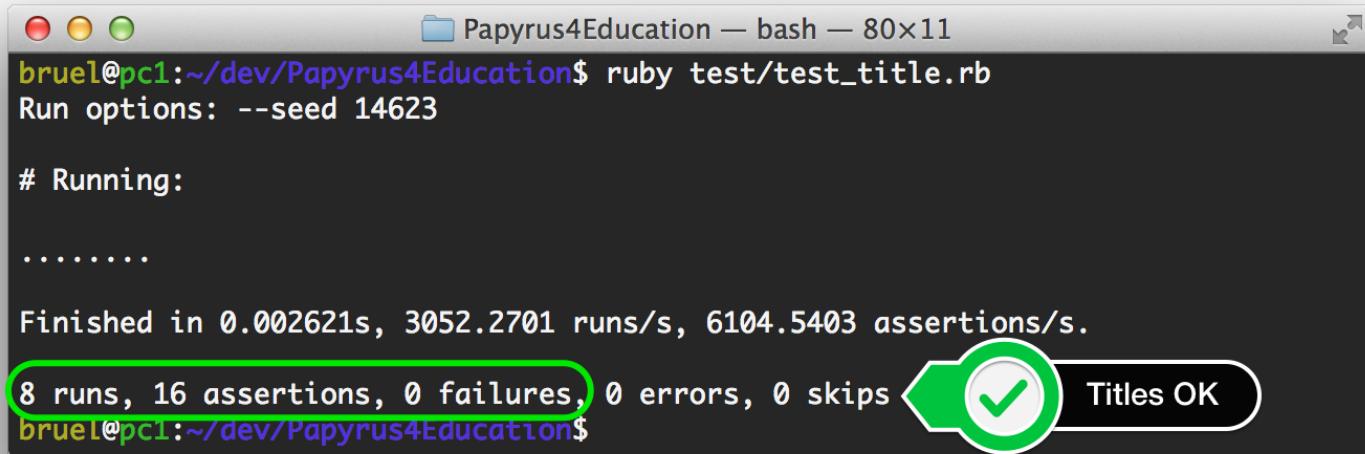
Par exemple la [Eclipse Doc Style Guide](#) préconise :

Use sentence capitalization for all titles.

Voilà le [script](#) de test pour cette règle.

```
#require 'minitest/spec'  
require 'minitest/autorun'  
  
#-----  
# Checking Titles  
#-----  
MAIN='main.wiki'  
File.open(MAIN) { |f|  
    content = f.read  
    titles = content.scan(/+= (.*) =+/)  
    # test that titles have Capital first letter  
    titles.flatten.each {|title|  
        describe File do  
            it "Wiki should use sentence capitalization for all titles" do  
                assert_match(/\[A-Z\].+$/, title)  
            end  
        end  
    }  
}
```

Et le résultat :



```
Papyrus4Education — bash — 80x11
bruel@pc1:~/dev/Papyrus4Education$ ruby test/test_title.rb
Run options: --seed 14623

# Running:

.....
Finished in 0.002621s, 3052.2701 runs/s, 6104.5403 assertions/s.

8 runs, 16 assertions, 0 failures, 0 errors, 0 skips
bruel@pc1:~/dev/Papyrus4Education$
```

Figure 30. Checking that head titles are capitalized

NOTE Bien sûr tout n'est pas testable. Exemple de règle difficile à instrumenter : "Wherever possible, make statements positive."

5.6. "Asciidoc ne marche pas sur ma machine!"

TIP ALors utiliser l'intégration continue ([CI]) et la virtualisation!

Nous allons détailler l'exemple d'un cas concret :

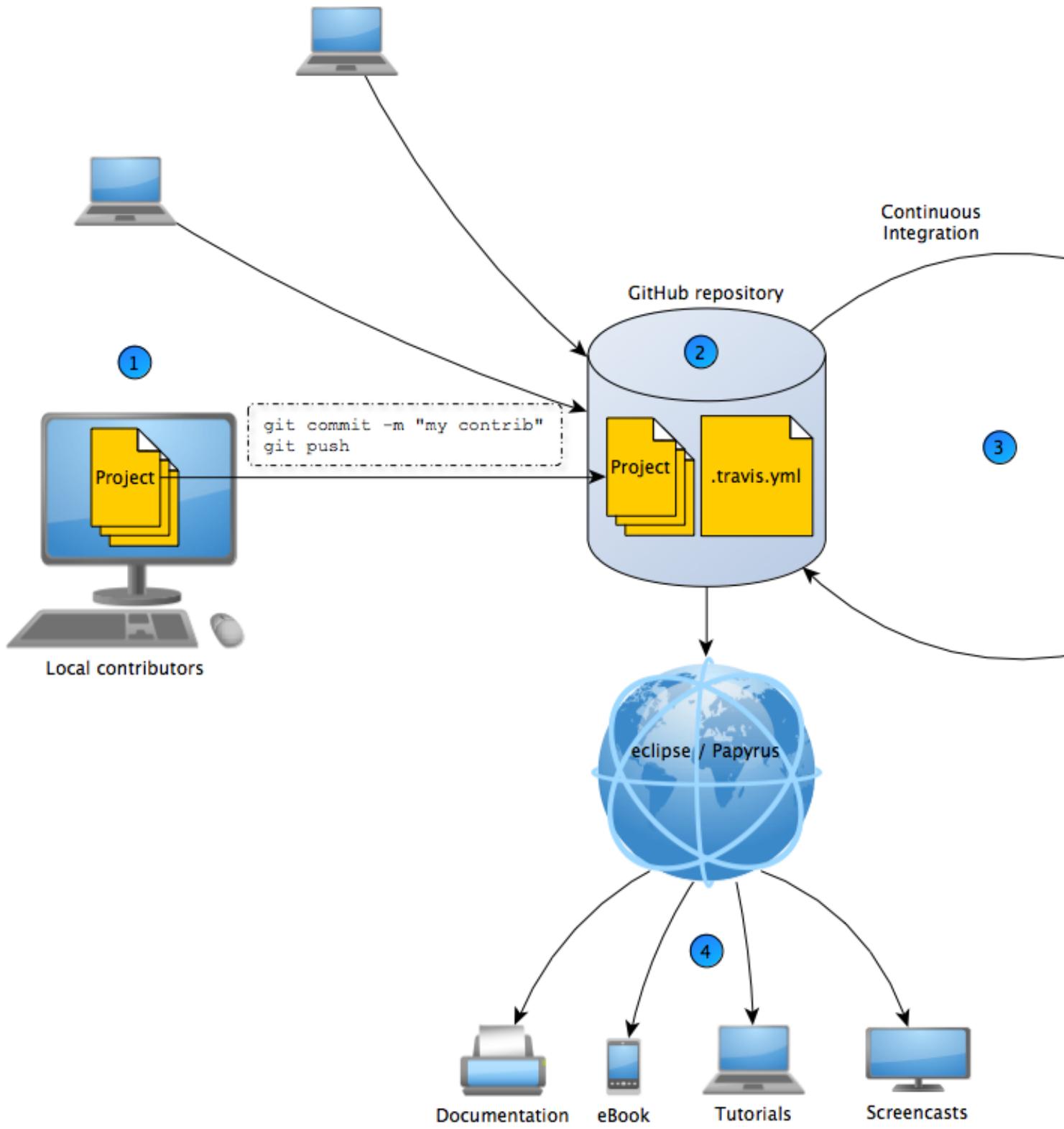


Figure 31. Le cas de la (future) documentation [http://www.eclipse.org/papyrus/\[Papyrus\]](http://www.eclipse.org/papyrus/[Papyrus])

5.6.1. [0] Le client fournit les templates, CSS stylesheets, ...

Hors de propos, mais en tant que programmeur, c'est très pratique.

```

myStyle.css
16 Class {
17     fillColor: #C3D7DD;
18 }
19 Association {
20     lineColor: blue;
21 }

23 -----
24 /* Hiding/Showing Class' compartments */
25 -----
26 Class > Compartment[kind="operations"] {
27     visible:false;
28 }
29
30 Class > Compartment[kind="attributes"] {
31     visible:true;
32 }
33
34 Class > Compartment[kind="nestedclassifiers"] {
35     visible:false;
36 }

38 -----
39 /* Hiding/Showing Associations' labels */
40 -----
41 Association > Label:sourceMultiplicity {
42     visible:true;
43 }
44 Association > Label:targetMultiplicity {
45     visible:true;
46 }
47
48 Association > Label:sourceRole {
49     visible:false;
50     /* maskLabel: name multiplicity; */
51 }
52
53 Association > Label:targetRole {
54     visible:false;
55     /* maskLabel: name multiplicity; */
56 }
57
58 Association > Label:name {
59     visible:true;
60 }
61

papyrus_theme.css
1 ****
2 * Copyright (c) 2012 CEA
3 *
4 * All rights reserved. This
5 * are made available under
6 * which accompanies this
7 * http://www.eclipse.org/
8 *
9 * Contributors:
10 * Camille Letavernier (C
11 *
12 /*
13 * Papyrus CSS v0.9
14 */
15
16 *
17 *
18 /**
19 * GMF Display
20 */
21
22
23 fillColor:#C3D1D5;
24
25 /* fontHeight:9; */
26
27 /**
28 * Gradient
29 */
30
31 /* gradientColor:white
32 /* gradientStyle:vertic
33
34 /**
35 * Global gradient (E
36 */
37
38 gradient: white vertical;
39
40 /**
41 * Global gradient (2
42 */
43
44
45
46 */

Properties
Model Validation

```

Figure 32. Un exemple de CSS <http://www.eclipse.org/papyrus/>[Papyrus]

NOTE Vous n'auriez pas en avoir bénéficié en MPA?

5.6.2. [1] Les contributeurs fournissent les inputs, en s'occupant du contenu

De la même façon que vous avez conçus vos fichiers sources [AsciiDoc](#).

5.6.3. [2] Tout est versionné et contrôlé

Dans notre exemple, nous utilisons [GitHub](#).

5.6.4. [3] Travis est utilisé pour lancer les tests et générer les outputs

Cf. détails sur la configuration ci-après.

5.6.5. [4] Le résultat est disponible sur le web sur de multiples formats

Les formats courants sont supportés :

- PDF
- ebook
- HTML
- Slides (Deck.js, Reveal.js, slidy2, ...)
- Docx
- ...

5.7. Configuration

Le but de ces notes est d'expliquer le fonctionnement de la génération de document HTML par intégration continue en utilisant [Travis](#) en complément des dépôts [GitHub](#).

La démarche globale (cf. [Un processus d'intégration continue](#)):

1. Se connecter à [Travis](#) avec ses identifiants [GitHub](#).
2. Configurer les dépôts à "intégrer" de manière continue
3. Ajouter un fichier .travis.yml à la racine du dépôt (penser à l'ajouter - git add)
4. Lors du git push (après un git commit bien sûr), [Travis](#) va générer une machine virtuelle qui va exécuter le script et indiquer un succès () si tout c'est bien passé.

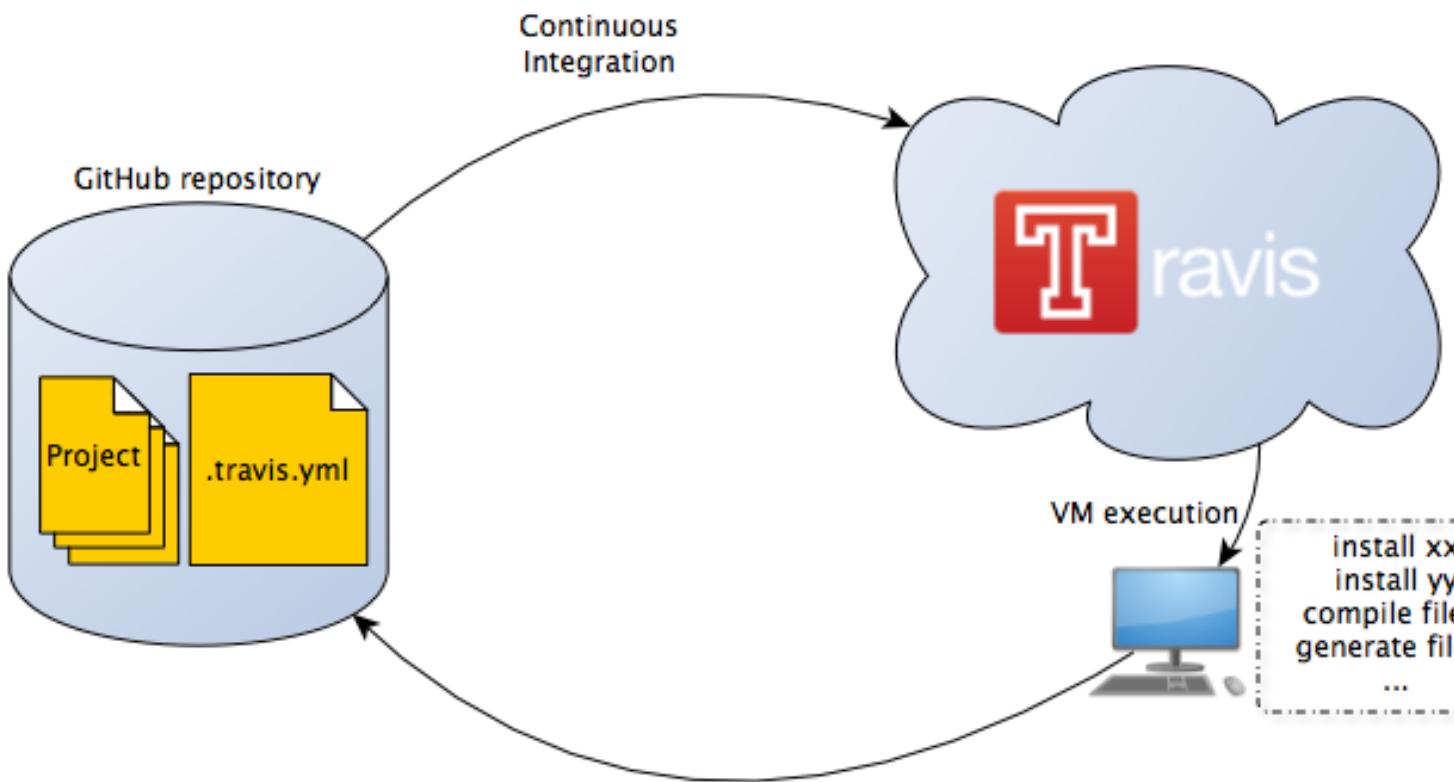


Figure 33. Un processus d'intégration continue

5.8. Connection entre Github et Travis

Pour ne pas avoir à préciser les mots de passe en clair dans vos scripts, il est possible d'indiquer les éléments de connexions de manière crypté. Ainsi il faut :

1. Associer un token à votre dépôt sous [GitHub](#) (cf. explications [ici](#)).
2. Installer [Travis](#)

```
gem install travis
```

3. Crypter (et ajouter au fichier `.travis.yml` en même temps) votre token

```
travis encrypt MY_SECRET_ENV=super_secret --add env.matrix
```

Il est possible de valider la syntaxe du fichier `.travis.yml` en installant un utilitaire :

NOTE

```
gem install travis-lint
travis-lint
```

5.9. Exemple de fichiers

5.9.1. Un fichier Travis

".travis.yml"

```
#language: python
#python:
#- '2.7'
install:
- gem install asciidoctor
- gem install link-checker

script:
#- asciidoctor --version
#- ls -alF
#- whereis bash
- asciidoctor -a icons -a linkless! -a numbered faq.txt -o output/faq.html
#- ls -al output
#- bash update-doc.sh
- rake check_links

# Following generated by
#
# travis login --pro
# travis encrypt -r jmbruel/PapyrusFAQ GH_TOKEN=<token generated on GitHub> --add
env.global
env:
global:
secure: cEaBJq020bnKSWQHP0c.....WMQX8Py4icaC8Cn9162AqE4=

# if everything OK, then deploy on the web
after_success:
- bash update-doc.sh
```

5.9.2. Un fichier de script lancé par Travis

update-doc.sh

```
#!/bin/bash
echo -e "Starting to update doc\n"

#copy data we're interested in to other place
cp -R output $HOME/output

#go to home and setup git
cd $HOME
git config --global user.email "jbruel#travis@gmail.com"
git config --global user.name "Jean-Michel Bruel"
git config --global push.default simple

#using token clone io pages branch
git clone --quiet https://$GH_TOKEN@github.com/jmbruel/jmbruel.github.io.git doc > /dev/null

#go into directory and copy data we're interested in to that directory
cd doc
cp $HOME/output/faq.html ./index.html
ls -al index.html

#add, commit and push files without provoking Continuous Integration
git add -f index.html
git commit -m "Travis build $TRAVIS_BUILD_NUMBER pushed to io pages -- [skip ci]"
git push -fq origin master > /dev/null

echo -e "Done magic with output\n"
```

TIP Notez le [skip ci], qui permet d'éviter la récursivité du processus (modif du push donc relance du script!).

5.9.3. Branches et SVN

Pas normal de ne pas pouvoir présenter une release qui fonctionne.

TIP Voir l'équivalent pour git [ici](#).

5.9.4. Questions de révision

- NOTE** 1. Pour ceux qui ont le support avec eux (ordi, tablette, portable) : trouvez un exemple de violation de la bonne pratique **[DRY]** dans ce support de cours (indice : mes codes **Ruby** sont souvent écrits à l'arrache)
- 2. Pourquoi les concepteurs de **Papyrus** insistent-ils sur la qualité de leur documentation?

Glossaire

Acronymes UML/SysML

act

Raccourcis pour Diagramme d'**ACT**ivité dans une cartouche **SysML™**

bdd

Raccourcis pour **Block Definition Diagram** dans une cartouche **SysML™**

dc

Diagramme de **Classe** **UML™**

dss

Diagramme de **Séquence Système** (un sd où seul le système dans sa globalité est représenté [Il ne s'agit pas d'un acronyme **SysML™** à proprement parler mais nous l'utilisons beaucoup.])

ibd

Raccourcis pour **Internal Block Diagram** dans une cartouche **SysML™**

par

Raccourcis pour **Parametric Diagram** dans une cartouche **SysML™**

pkg

Raccourcis pour **PaKaGe Diagram** dans une cartouche **SysML™**

req

Raccourcis pour **REQuirements Diagram** dans une cartouche **SysML™**

sd

Raccourcis pour **SEQquence Diagram** dans une cartouche **SysML™**

stm

Raccourcis pour **STate Machine** dans une cartouche **SysML™**

uc

Raccourcis pour **Use Case Diagram** dans une cartouche **SysML™**

Définitions générales

Ressources

Les définitions ci-dessous sont regroupées à titre indicatif. Je vous invite à consulter les sources suivantes :

NOTE

- Glossaire du [Software Engineering Institute](#)
- [IEEE Computer Dictionary Online](#)
- [Wikipedia](#) – Version française

CI

Continuous Integration : (Intégration Continue) est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée (source link: [ici](#)).

DRY

Don't Repeat Yourself : Un bon principe qui veut qu'on évite de répéter des tâches manuelles (comme les tests) en utilisant plutôt des scripts et des programmes.

OMG

Object Management Group : L'organisme international chargé des principales normes liés à l'objet (CORBA, UML, etc.).

TDD

Test Driven Development : Développements dirigés par les tests. On écrit les tests avant d'écrire le code. On travaille son code tant que les tests ne passent pas.

TRL

Technology Readiness Level : Système de mesure employé par des agences gouvernementales américaines et par de nombreuses compagnies (et agences) mondiales afin d'évaluer le niveau de maturité d'une technologie (cf. [Wikipedia](#)).

STI2D

Sciences et Technologies de l'Industrie et du Développement Durable : série du baccalauréat qui met l'accent sur les technologies transversales et qui a introduit en 2011 l'enseignement de [SysML™](#).

SysML

System Modeling Language™ : Le langage de modélisation de systèmes maintenu par l'[OMG™](#).

UML

Unified Modeling Language™ : Le langage de modélisation généraliste maintenu par l'[OMG™](#).

Liens utiles

- Le site officiel d'[UML™](#) : <http://www.uml.org/>

- Un site très bien fait sur UML™ : <http://www.uml-sysml.org/>

Références

Voici quelques références utiles.

- [ENS2013] L. Gendre et J.-M. Virely, SysML. Tutoriel du 13/06/2013. ENS Cachan.
- [FIO2012] Fiorèse S., Meinadier J., Découvrir et comprendre l'ingénierie système, AFIS 2012.
- [FMS] A. Moore, R. Steiner, S. Friedenthal, A Practical Guide to SysML, The MK/OMG Press, MK/OMG Press, 2011 (2nd edition).
- [Fejoz2013] Loïc Fejoz. SysML4STI2D, présentation de SysML en STI2D, 2004. Disponible [ici](#).
- [Fowler2004] Martin Fowler. UML 2.0 INFORMATIQUE PROFESSIONNELLE, 2004.
- [HAS2012] Haskins C., SE Handbook Working Group, INCOSE Systems Engineering Handbook: Version 3.2.2, International Council on Systems Engineering, 2012.
- [Harmony] Bruce Powel Douglass. Real-Time Agility: The Harmony/ESW Method for Real-Time and Embedded Systems Development. Addison-Wesley Professional, 2009. ISBN-10: 0-321-54549-4
- [KAP2007] Kapurc S., NASA Systems Engineering Handbook, 2007 ([pdf](#)).
- [MeDICIS] ENSI Bourges/PRiSM.
- [Modelio2012] Systems Engineering using Modelio, INCOSE 2012 Tool Vendor Challenge Case Study. Disponible [ici](#).
- [OMG2009] The Current State of Model Based Systems Engineering: Results from the OMG SysML Request for Information. Mary Bone and Robert Cloutier, 2009. Disponible [ici](#).
- [REQ2012] Guide Bonnes Pratiques en Ingénierie des Exigences, AFIS 2012.
- [Roques2010] **Pascal Roques**. SysML par l'exemple - Un langage de modélisation pour systèmes complexes. Eyrolles. À obtenir [ici](#).
- [SeeBook2012] Embedded Systems Analysis and Modeling with SysML, UML and AADL, F. Kordon, J. Hugues, A. Canals, A. Dohet, Wiley, 2013.
- [Sommerville1997] Ian Sommerville, Pete Sawyer. Requirements Engineering: A Good Practice Guide. Wiley, 1997.
- [Styles] Scott W. Ambler. The Elements of UML 2.0 Style. Cambridge University Press, 2005. ISBN: 0-521-61678-6
- [SysML] OMG. Systems modeling language version 1.3. Technical report, 2012. Available [here](#).

- [Walsh1999] Norman Walsh & Leonard Muellner. DocBook - The Definitive Guide. O'Reilly & Associates. 1999. ISBN 1-56592-580-7.
- [TAOUP] Eric Steven Raymond. 'The Art of Unix Programming'. Addison-Wesley. ISBN 0-13-142901-9.
- [TestsIndustriels2009] Guide méthodologique de l'industrialisation et référentiel de bonnes pratiques. Disponible [ici](#).

A propos de ce document...

Document généré par [Jean-Michel Bruel](#) via [Asciidoctor](#) (version 0.1.5) de 'Dan Allen', lui-même basé sur [AsciiDoc](#). Pour l'instant ce document est libre d'utilisation et géré par la 'Licence Creative Commons'. [licence Creative Commons Paternité - Partage à l'Identique 3.0 non transposé](#).