

La Modélisation au service de l'Agilité

Philippe Desfray – Journées Neptune – Mai 2013

SOFTEAM - Modeliosoft

Pourquoi poser la question Modélisation et Agilité?

- Outils de productivité bienvenus avec développements agiles (IDE, construction d'IHM, ...)
- L'Agilité est une affaire de processus, pas de technologie.



Modéliser n'a pas bonne presse chez les développeurs, encore moins chez les « Agiles »

- Ca n'a jamais été le cas
- Les processus agiles redonnent l'initiative aux développeurs
- Modéliser est perçu comme une activité bureaucratique, éloignée du concret de la réalisation
- Les modèles ne sont pas perçus comme des outils de productivité (la faute aux outils)
- Les cycles de vie (Waterfall, V, ...) honnis ont magnifié les modèles en phase amont (*les amis de mes ennemis sont mes ennemis*)

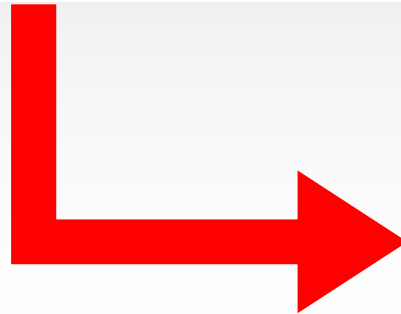
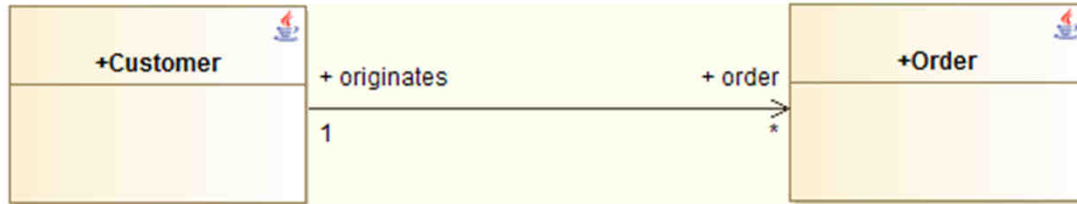
Fournir une valeur ajoutée « pragmatique » - contre exemples

- Le schéma « modèle immuable produisant un code non synchronisé » ne marche pas
 - Les changements et évolutions de code sont permanents en phase de développement
- Remplacer un artéfact syntaxique par un graphique n'a aucun intérêt a priori



```
package customerManagement;  
  
public class Customer {  
  
}
```

Fournir une valeur ajoutée « pragmatique » - un code utile et maintenu lors d'évolutions



- Une coopération modèle/code « fine »
 - Changer règle génération
 - Possibilité accesseur totalement manuel
 - Cohérence permanente code/modèle
- Un support aux changements
 - Changement nom « Order »
 - Changement Cible association
 - ...

```
package customerManagement;

import java.util.ArrayList;
import java.util.List;
import com.modeliosoft.modelio.javadesigner.annotations.objid;

public class Customer {
    public List<Order> order = new ArrayList<Order> ();

    public List<Order> getOrder() {
        // programmers code to be entered
        return this.order;
    }

    public void setOrder(final List<Order> value) {
        // programmers code to be entered
        this.order = value;
    }
}
```

Les valeur ajoutées d'un modèle

- Abstraction :
 - Réduit la complexité
 - Facilite la communication (ex: Développeur/Architecte)
- Génération
 - Systématise des manières de programmer, et d'organiser le code
 - Apporte plus en maintenabilité qu'en productivité pure
- Plusieurs utilisations possibles de modèles
 - Comprendre, poser le problème
 - Définir la solution
 - Produire/générer
- Conception et Analyse restent deux activités indispensables parfois malmenées en projets Agiles



Apporter une approche souple, non dogmatique

- Le « modèle d'abord » ne doit pas être imposé
- « Modéliser tout le temps » non plus
- Autoriser un référentiel souple : « exemples », « tentatives », « idées », « vision » sont des modèles utiles
- Par exemple : Use Cases \neq User Stories. L'un explicite les exigences en amont, l'autre fournit une courte phrase déterminant ce que le système doit faire
 - *Log in to my diversified portfolio portal*
 - *See my daily loss or profit.*
 - *Check my current long term interest rates.*

Valeur ajoutée des modèles sur le code existant

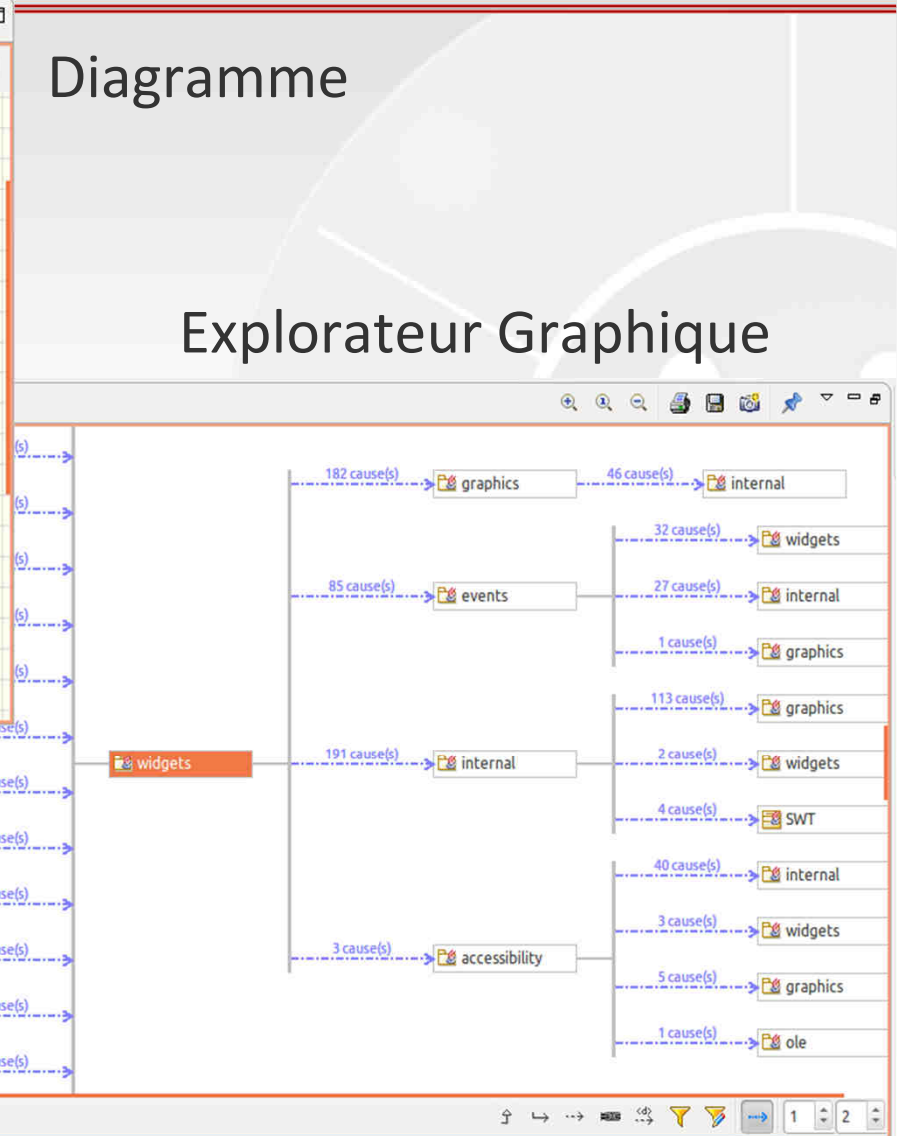
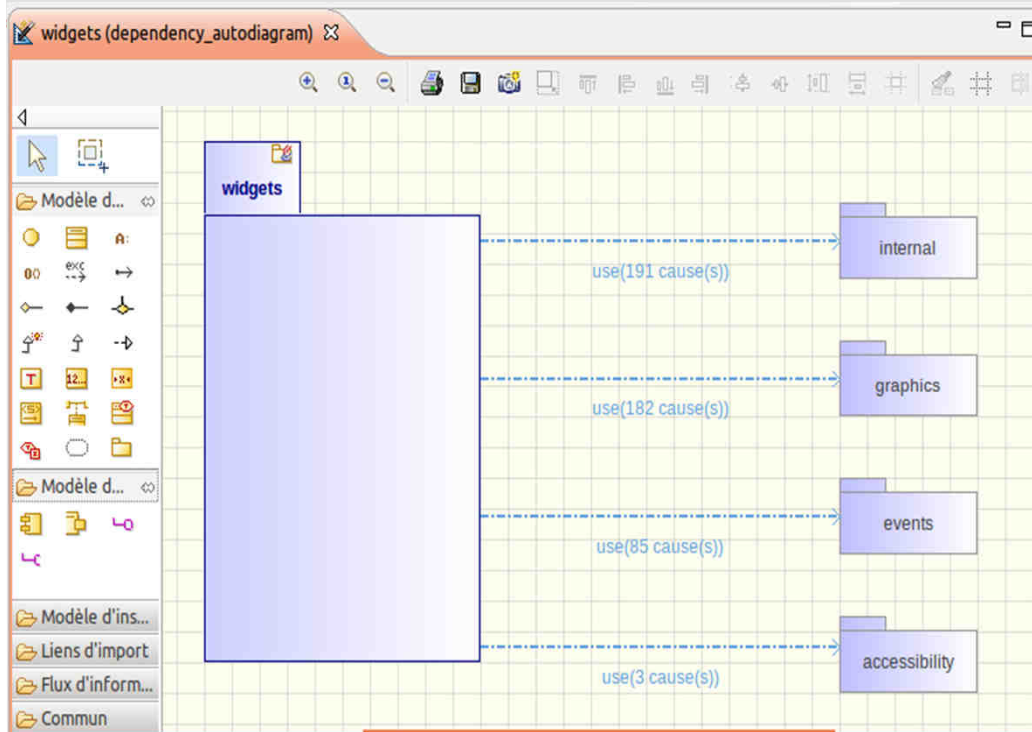
- L'existant en code est énorme : applications, frameworks, librairies ...
- Le programmeur doit pouvoir bénéficier des modèles à tout moment (en cours de développement, en maintenance, en mode « pompier » sur existant en crise, ...)
- Un support puissant du « reverse engineering » doit permettre
 - De basculer « code pure » vers code guidé par modèle
 - De faire un audit « niveau modèle » d'un code complexe
 - De rétro-documenter
 - de restructurer et moderniser le code



Synthèse d'un code existant

Diagramme

Explorateur Graphique



Synthèse des dépendances

Rétrodocumentation

Javadoc augmentée

The screenshot shows a Mozilla Firefox browser window displaying the Javadoc for the `org.eclipse.swt.widgets.Menu` class. The page is titled "Menu - Mozilla Firefox" and has a "Menu" tab open. The left sidebar shows a list of classes under "All Classes" and "Packages". The main content area shows the class details for `Menu`, including its inheritance hierarchy, a list of styles, events, and a structure diagram. The structure diagram shows the `Menu` class with a `+ handle : integer` attribute and three associations: `- backgroundImage` to `Image`, `- parent` to `Decorations`, and `- imageList` to `ImageList`. All associations have a multiplicity of `0..1`.

```
graph TD
    Menu[Menu] -- "- backgroundImage" --> Image[Image]
    Menu -- "- parent" --> Decorations[Decorations]
    Menu -- "- imageList" --> ImageList[ImageList]
```

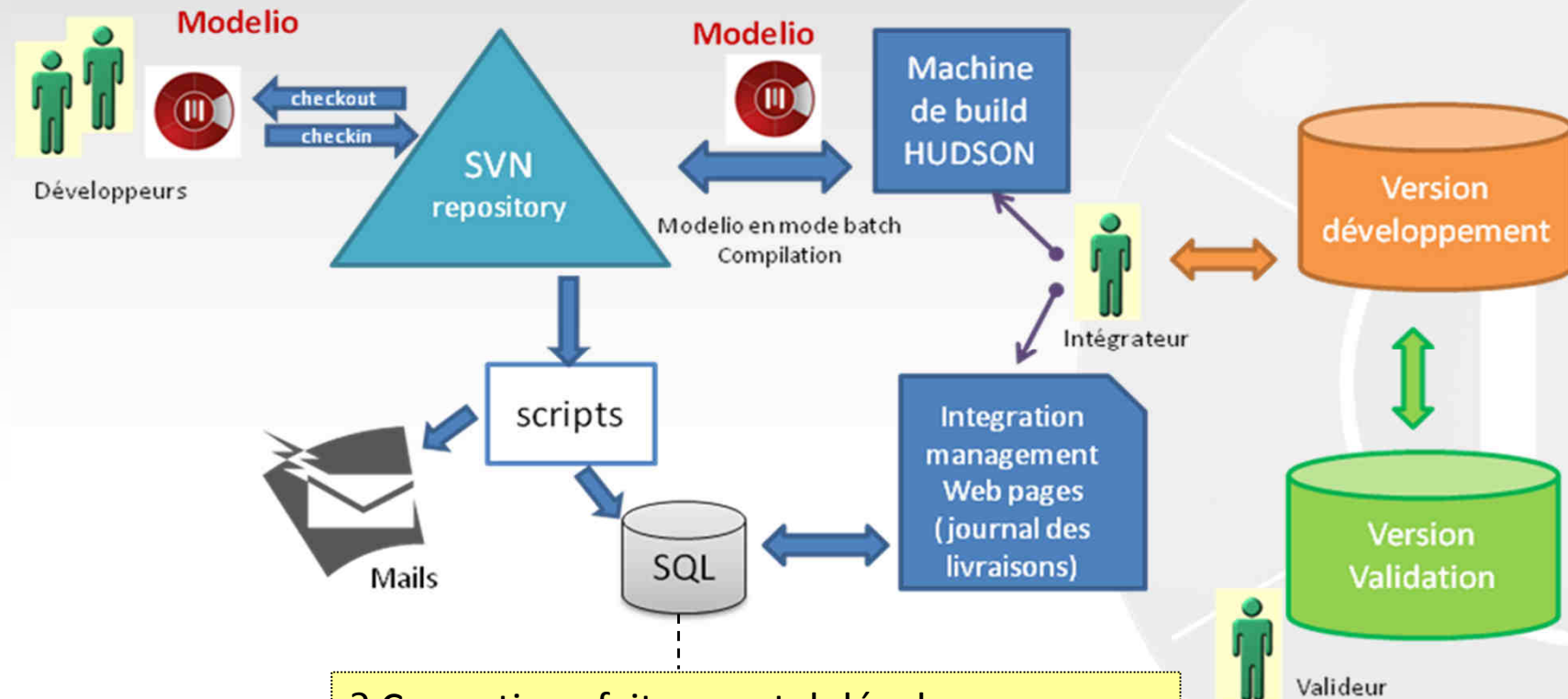


MDA Agile :

L'outillage doit être intégré dans la chaîne de production

- MDA nécessite une organisation d'équipe dédiée, des outils adaptés pour passer à l'échelle de grandes équipes
- Sa mise en place, en phase initiale doit aboutir à une chaîne de production intégrée pour un développement agile
- Domaines essentiels souvent rébarbatifs pour les développeurs peuvent être facilités par une approche guidée par le modèle
 - Gestion de version et configuration,
 - Intégration,
 - Validation,
 - Organisation du travail de groupe

Intégrer l'atelier de modélisation dans la chaîne de production (exemple)

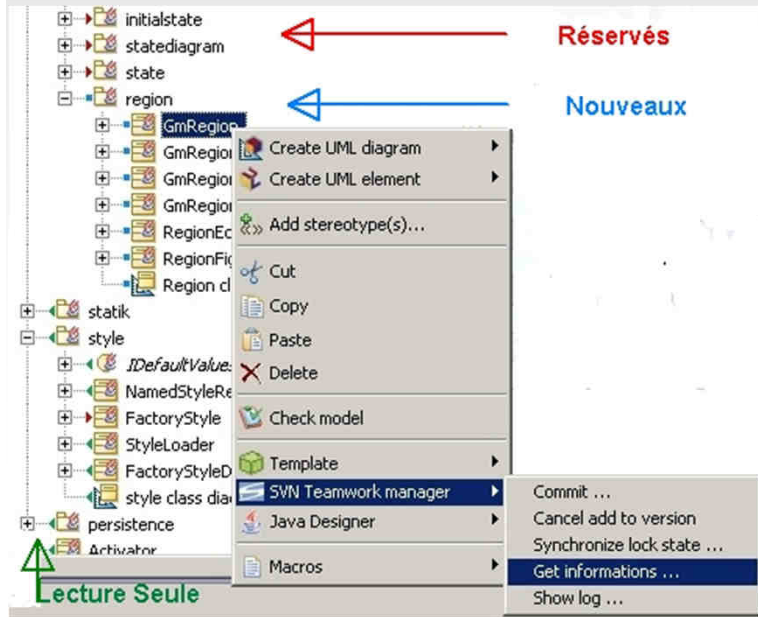


- ? Corrections faites par tel développeurs
- ? Anomalies corrigées
- ? Interventions sur tel composant

Model Driven, MDA, Intégration continue



Customisation Modelio: Outillage Dédié



Modelio
Teamwork manager

Delivery form

Delivery Form

Filling in this form accurately is a key point of our process!

Delivery contents identification

Id: 17911
User: none
Date: null-date
Target: modelio2.0.plugins
Toolkit: toolkit 3.4.x

Model modifications

Linked deliveries: External files rev.: Resource files rev.:

Activity identification

Indicate here the reason for this delivery

Fix a Mantis bug 1274 Development task ID: N/A

Visible changes

Visible changes? No Yes

As the author of this delivery you have to officially declare whether this delivery introduces modifications visible by the end-user or not. You will not be allowed to validate this delivery form before having chosen one of the proposed options.

Any end-user visible change (features, behaviors, look...) MUST BE listed here. This required information will be used by the validation team and will feed the documentation and/or the release notes document.

The style colors are now propagated from the parent to its children unless a specific style is set on the children.

Technical details

List here the technical details you think useful for the members of the team.

Had to refactor the Style->parentStyle association. Requires code generation and recompilation.

Submit

Paramétrage Modelio



MDA: plusieurs pratiques au sein du même projet

- Génération complète (Metamodèle → framework): Travail exclusif au niveau du modèle
- Génération partielle (conception technique): Génération en mode « Model Driven ». Le modèle est maître. La modification de code est encadrée (balisée). Architecture globale, classes essentielles et structurantes.
- Génération « roundtrip » (codage): Le code est maître. génération depuis modèle, modification/synchronisation à partir de code. Classes techniques, complétementant l'architecture (ex: IHM, « ActionListener », ...)

Passage à l'échelle MDA: on évite

- Cascade de transformations de modèles/mises à jour manuelles
 - Mise à jour des modifs d'un modèle vers un modèle transformé: OK
 - ~~Mise à jour des modifs d'un modèle vers un modèle transformé modifié manuellement~~
- Plusieurs modèles/artefacts partiellement déduits les uns des autres, chacun gérés en configuration (ex: modèle et code séparés)
- Un modèle de haut niveau sur lequel interviennent un nombre limité de personnes génère/met à jour des modifications disséminées dans un modèle de plus bas niveau sur lesquelles interviennent plusieurs personnes

Modelio : Des outils pour assister les développeurs sur les grands projets

Résultat de l'outil diffmerge
Cette vue présente les différences détectées entre le projet actuellement ouvert et le projet de référence.

Reste 0 différences sur 22

Métaclasses	Objet	Type de différence	Solution choisie	Description
↔	order	AssociationEnd créé(e)	Non applicable	AssociationEnd 'supplier' créé(e) sous Class 'Order'
↔	product	AssociationEnd créé(e)	Non applicable	AssociationEnd 'supplier' créé(e) sous Class 'Product'
↔	description	Attribut créé(e)	Non applicable	Attribut 'description' créé(e) sous Class 'Product'
↔	email	Attribut créé(e)	Non applicable	Attribut 'email' créé(e) sous Class 'Client'
↔	price	Attribut créé(e)	Non applicable	Attribut 'price' créé(e) sous Class 'Product'
↔	Supplier	Class créé(e)	Non applicable	Class 'Supplier' créé(e) sous Package 'Products'
↔	application	Connections réorganisé(e)s	Non applicable	Éléments réordonnés sous Association 'application'
↔	carries	Connections réorganisé(e)s	Non applicable	Éléments réordonnés sous Association 'carries'
↔	Functional Domains	Valeur d'attribut changée	Non applicable	Le contenu du diagramme a changé
↔	client	Valeur d'attribut changée	Non applicable	Multiplicité maximum valait *, vaut maintenant 1
↔	modifyDetails	Operation créé(e)	Non applicable	Operation 'modifyDetails' créé(e) sous Class 'Product'

Conserv. l'état courant Fusionner le contenu des notes Passer à l'état de référence

Modèle Courant

- Client
 - +name: string
 - +address: Address
 - +email: string
 - +account: Account [0..1]
 - +form: ApplicationForm [0..1]
- ApplicationForm
- ApplicationField

Modèle de Référence

- Client
 - +name: string
 - +address: Address
 - +account: Account [*]
 - +form: ApplicationForm [0..1]
- ApplicationForm
- ApplicationField
- Client

Propriétés - Annotations

Attribut	Valeur
Nom	email
Type	string (from _S_PredefinedTypes)
Visibilité	Publique
Multiplicité minimum	1
Multiplicité maximum	1

OK Appliquer Annuler

Diff/Merge de modèles

DiscountTravel - Modelio by Modeliosoft

File Edit Configuration MDA Views Help

Search:

Links Editor

UML diagram showing relationships between Client, Sales person, Customer, Order, IndividualTrip, Bill, and Sales. It includes associations like 'order', 'orderLine', 'bill', and 'Sales', along with stereotypes like 'flow', 'responsible of', 'communicates with', and 'realizes'.

Element: UML Extensions > « business entity »

Class	Name	Value
Order	Name	Order
Order	Visibility	Public
Order	Abstract	<input type="checkbox"/>

1 items selected

Analyse d'impact sur modèles

Gains de productivité: ne pas oublier l'essentiel

- Coût codage = 30% coût de réalisation
- Coût de réalisation = 30% coût de possession d'un logiciel
 - ➔ Maintenabilité constitue un enjeu très important
- Une approche guidée par le modèle bien gérée
 - Améliore l'analyse et la conception, l'architecture
 - Automatise (partiellement) le codage
 - Assiste la gestion de configuration, de version, test et validation donc améliore la maintenabilité
- Gains obtenus par des procédures et de l'outillage adaptés.
- Pour être compatibles avec l'agilité, les outils de modélisation doivent apporter des bénéfices tangibles aux développeurs

