



Carbon

ET QUELQUES IDÉES ...

Cédric Dumoulin



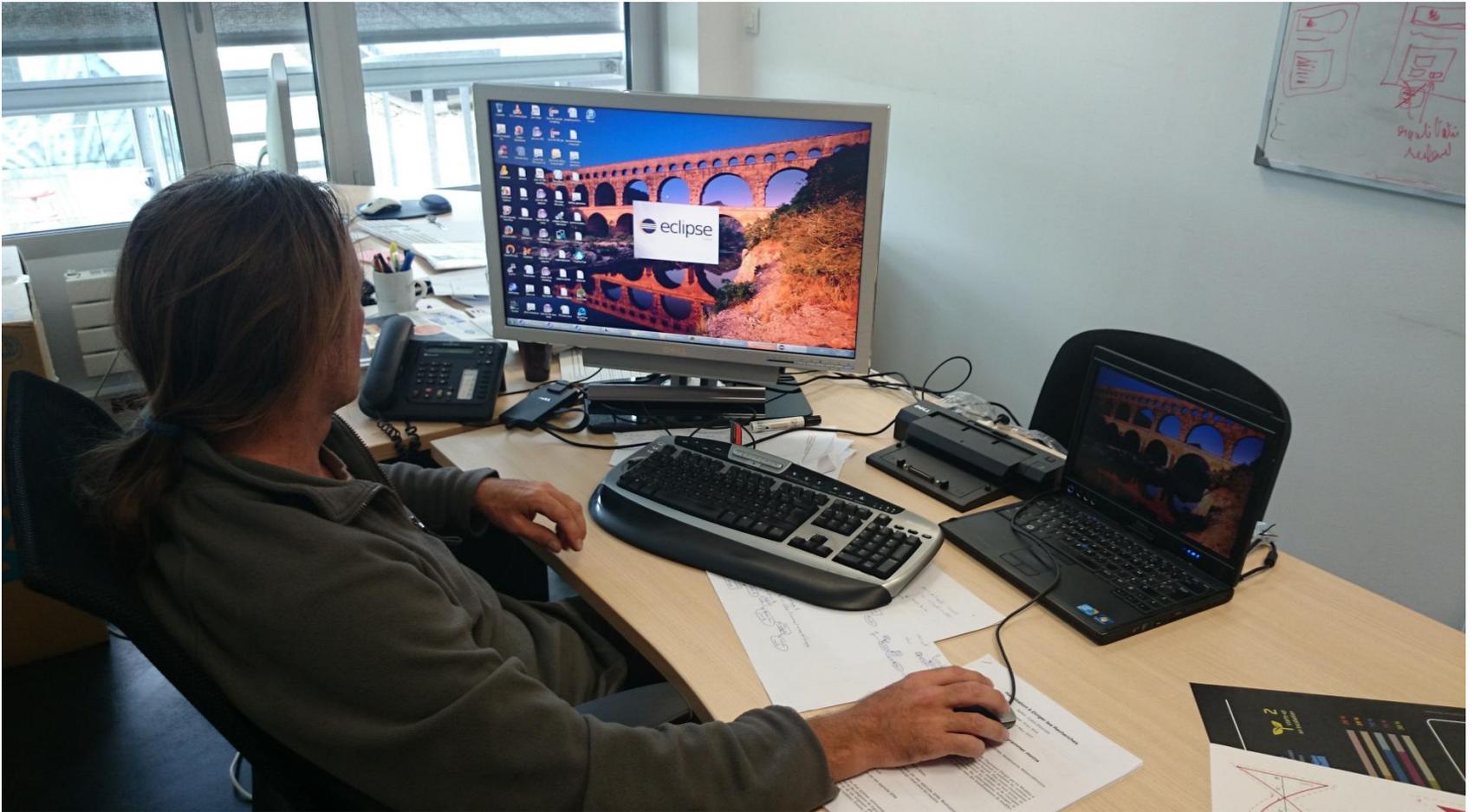
Carbon : Motivations

Scénario

3

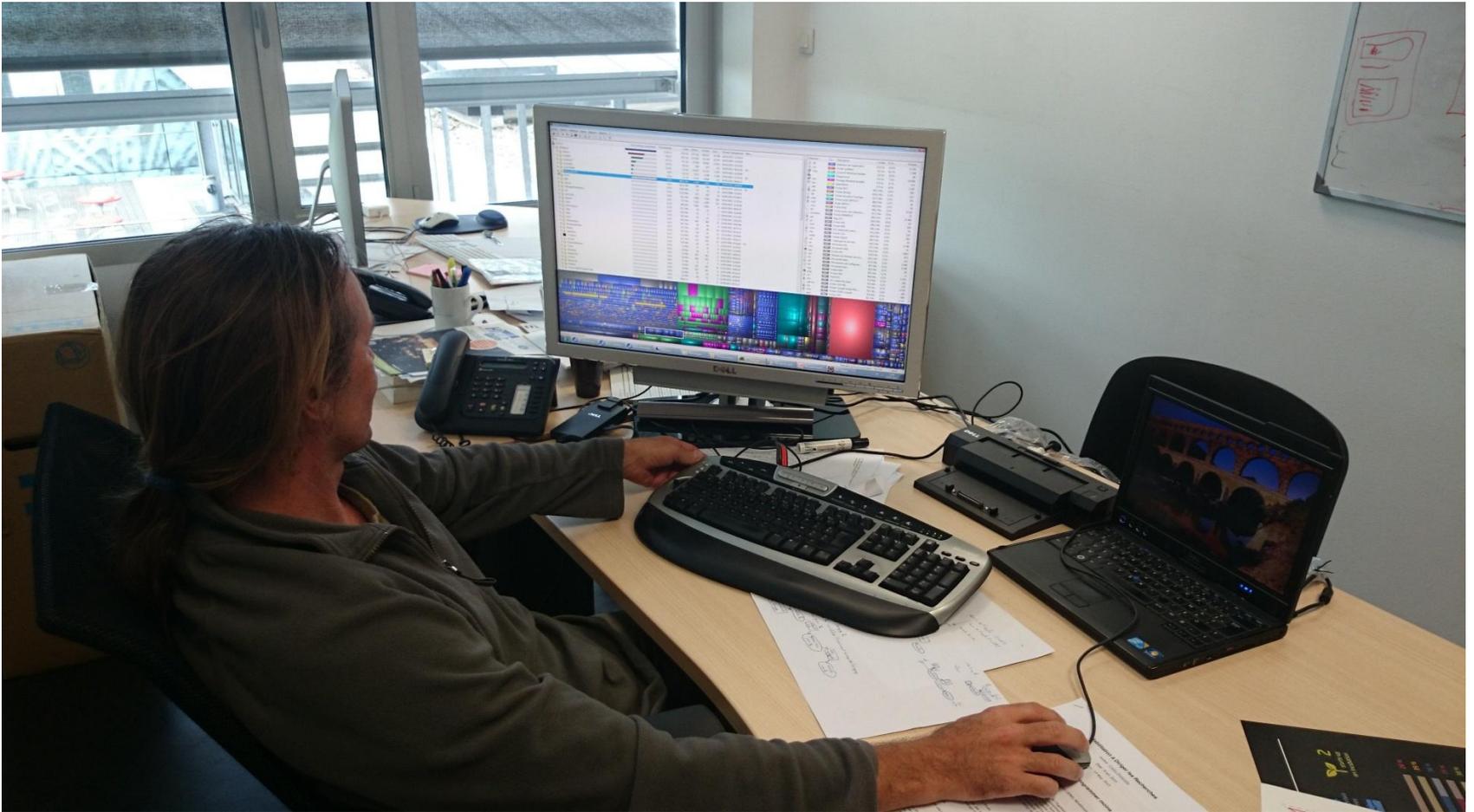
- Cas fictif d'un développeur dans une grande société. Appelons-le par exemple « Cédric »...
 - ▣ dans une équipe de 10 développeurs
 - ▣ il travaille actuellement sur 5 grands projets...
 - Enfin, il « travaillait » car il a dû s'absenter pendant 1 an. Il revient aujourd'hui, et son chef de projet lui demande de résoudre un problème dans un de ces 5 projets.

- « Facile », se dit-il, « je les connais bien, et même si ça fait un an que je n'ai pas touché, je vais m'y remettre sans problème ! »





- Cédric ouvre le projet:
 - 4.000 classes,
 - 50 packages,
 - 600.000 lignes de code !



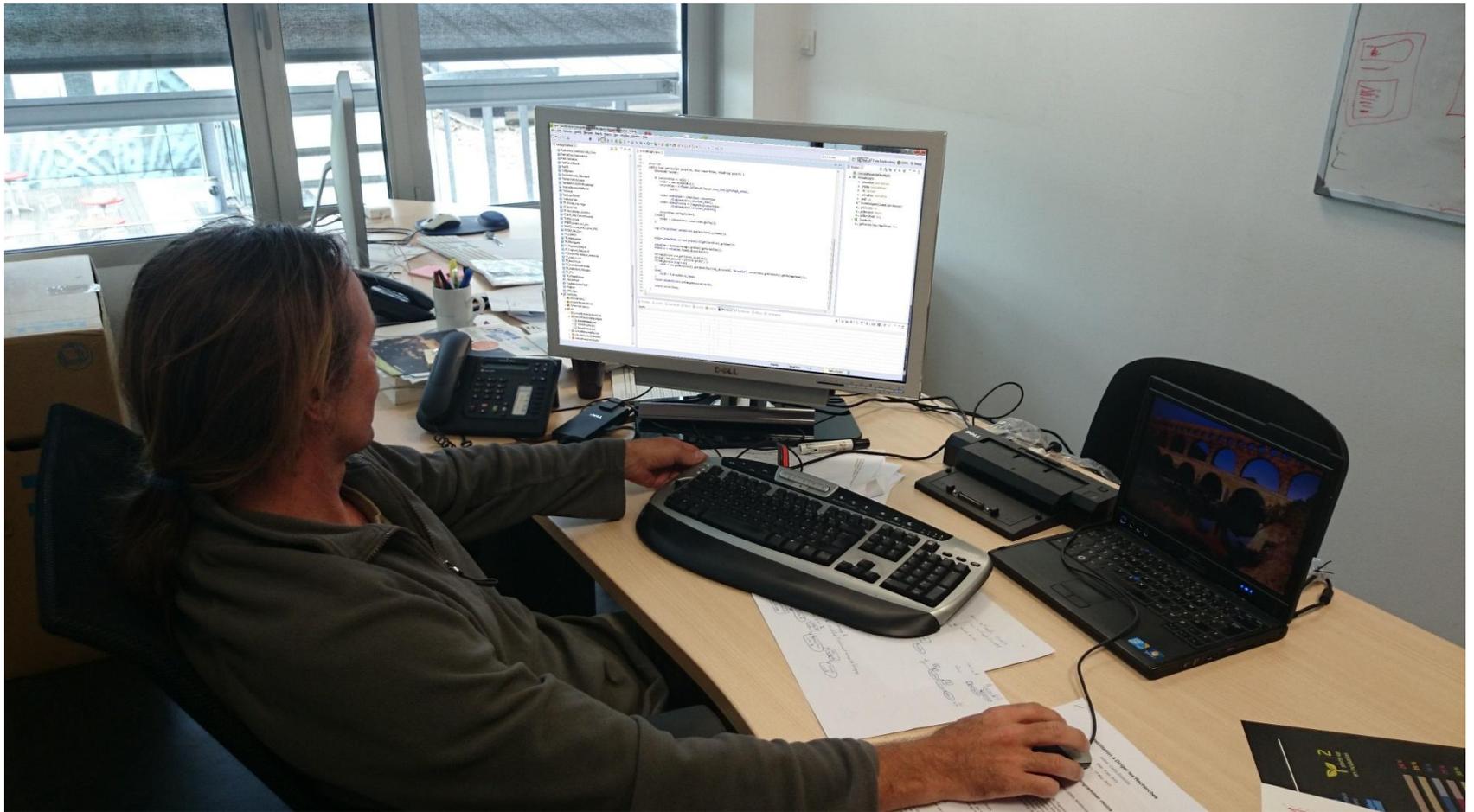
Je ne me rappelais pas qu'il y avait
autant de « trucs » là-dedans !





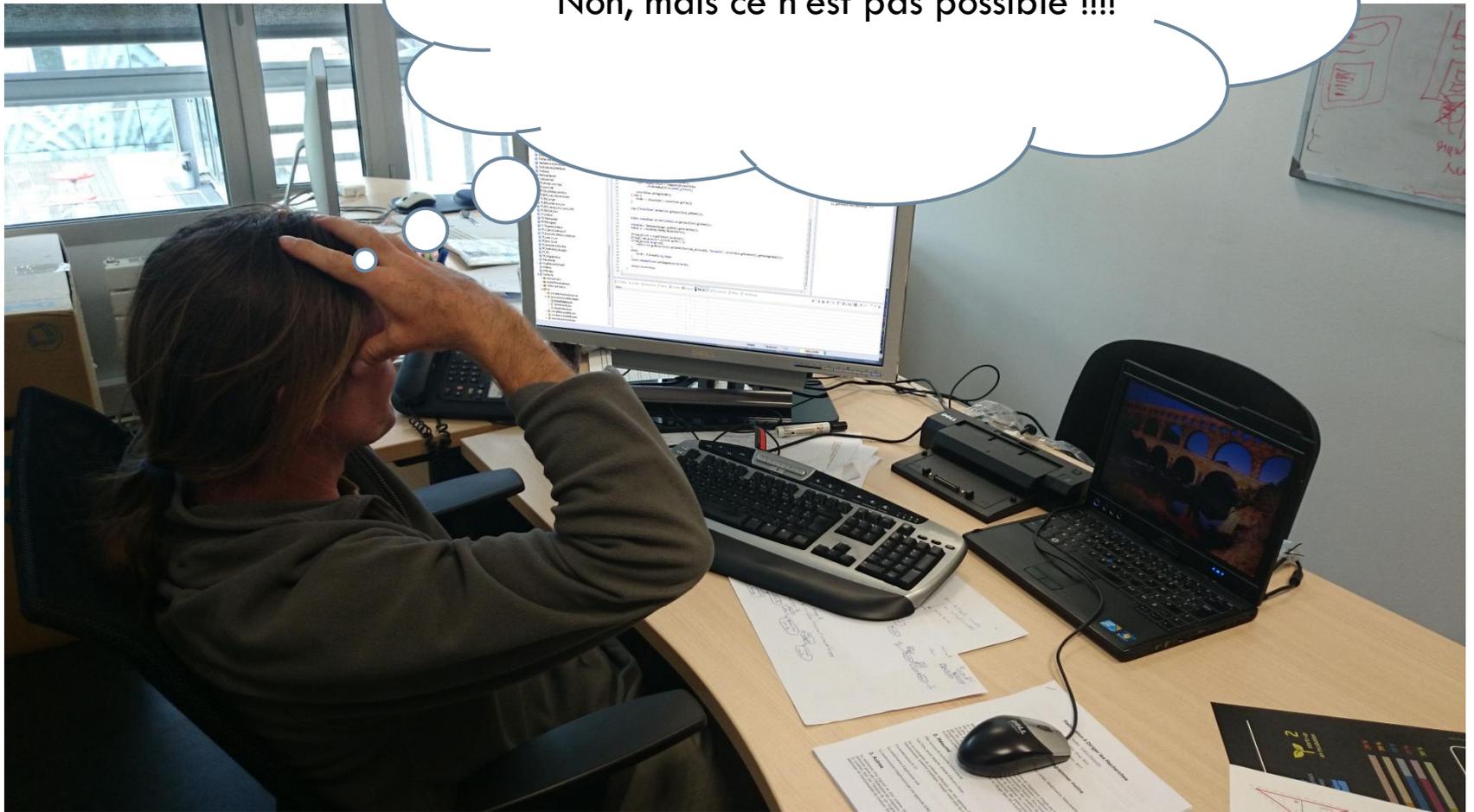


- Cédric ouvre la classe concernée...



- Mais malheureusement, comme souvent...

Non, mais ce n'est pas possible !!!!



```

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;

    if (convertView == null) {
        holder = new ViewHolder();
        convertView = inflater.inflate(R.layout.item_list_affichage_animal,
            null);

        holder.animalName = (TextView) convertView
            .findViewById(R.id.animal_Name);
        holder.animalPicture = (ImageView)convertView
            .findViewById(R.id.animal_picture);

        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    Log.v("AnimalName", animalList.get(position).getName());

    holder.animalName.setText(animalList.get(position).getName());

    animalDao = DatabaseManager.getDao().getAnimalDao();
    Animal a = animalDao.findById(position+1);

    String picture = a.getPicture_location();
    String[] tab_picture = picture.split(",");
    if(tab_picture.length>0){
        resID = ctx.getResources().getIdentifier(tab_picture[0], "drawable", convertView.getContext().getPackageName());
    }
    else{
        resID = R.drawable.no_image;
    }
    holder.animalPicture.setImageResource(resID);

    return convertView;
}

```

- Eh oui, le développeur n'a mis aucun commentaire dans le code !
 - C'est bien connu, « **les commentaires, ça ne sert à rien...** », n'est-ce pas 😊

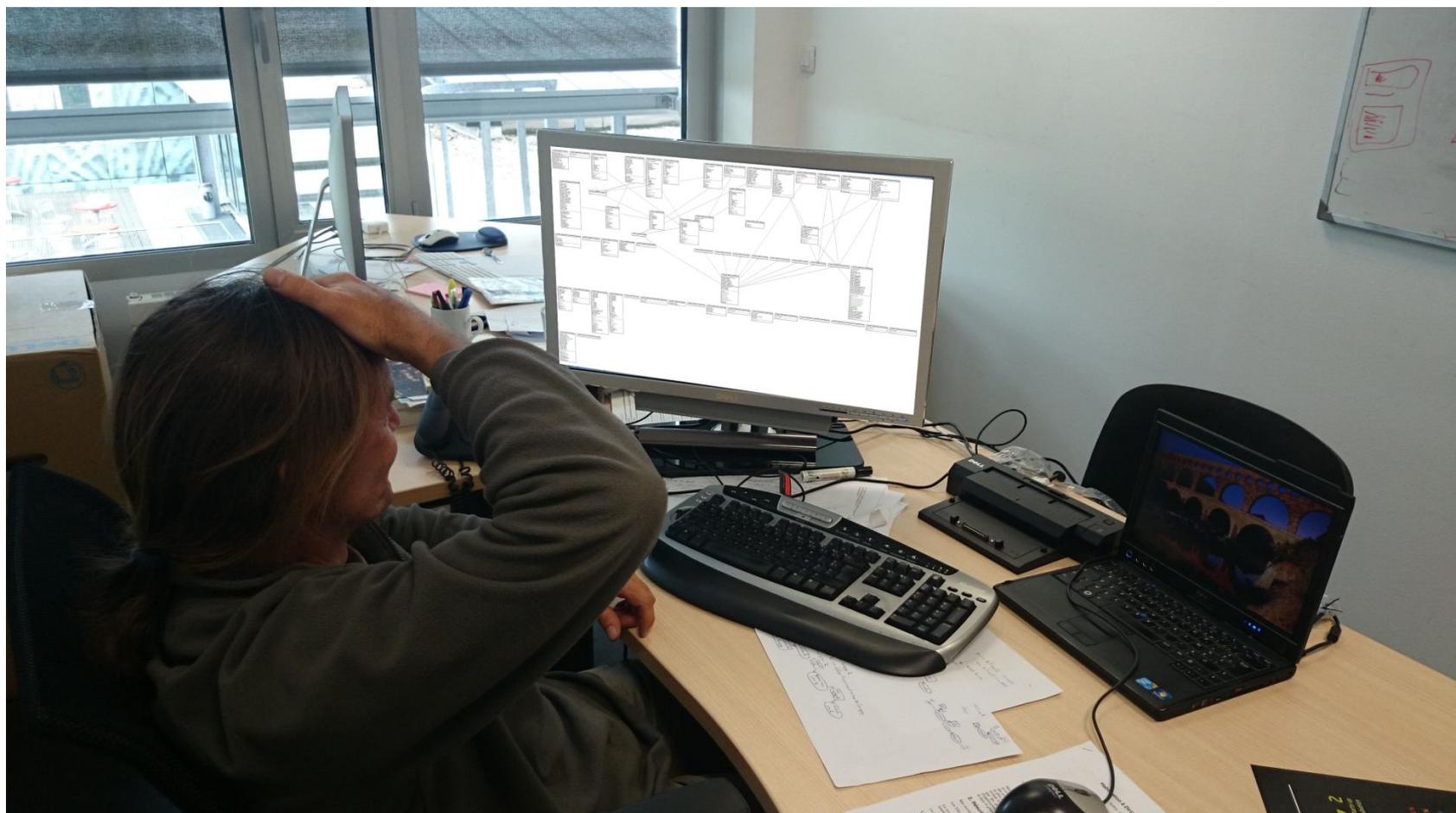
- Pas immédiat de savoir qui a écrit ça, pourquoi, etc.



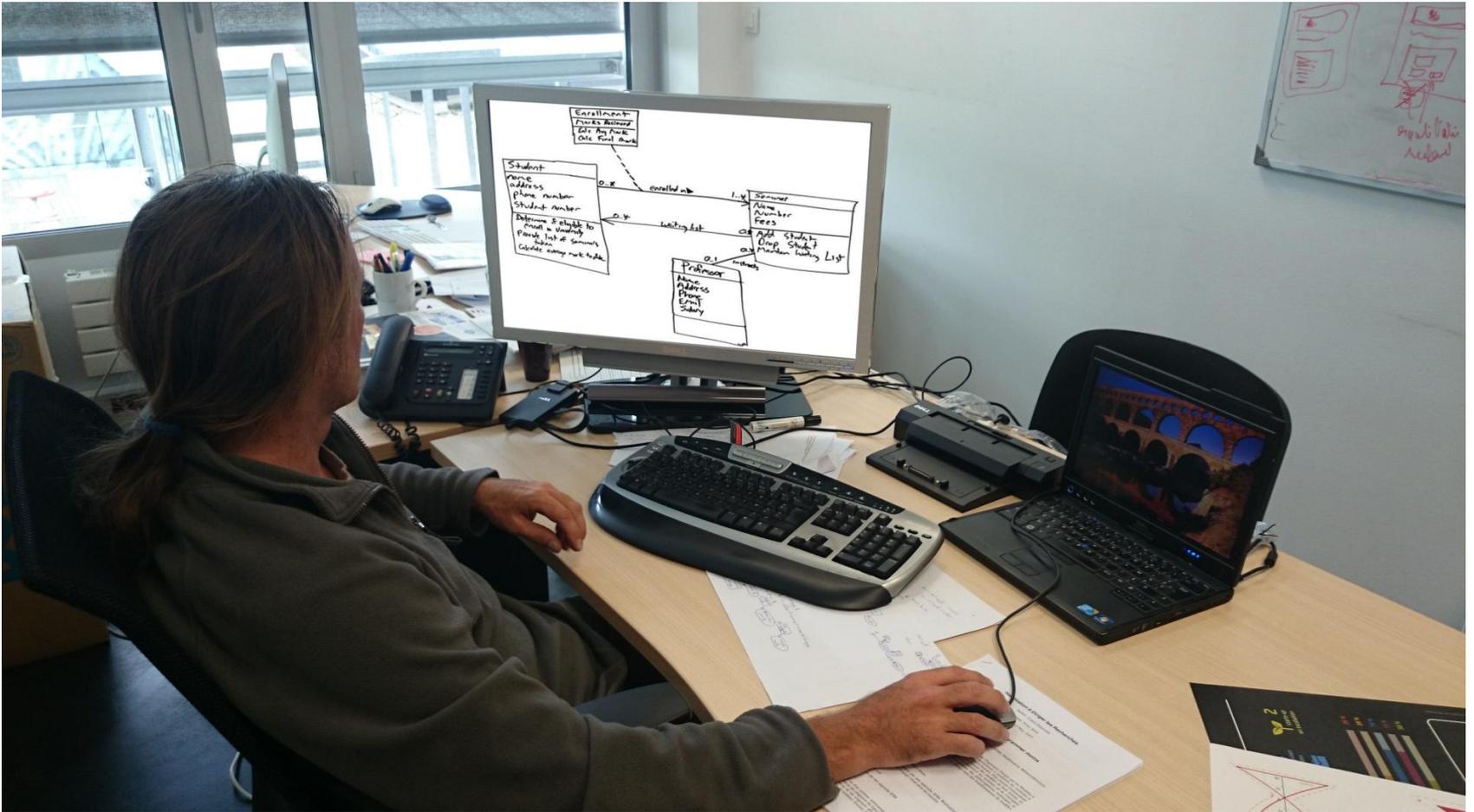
Pas grave, un petit « reverse » du projet et le tour sera joué...

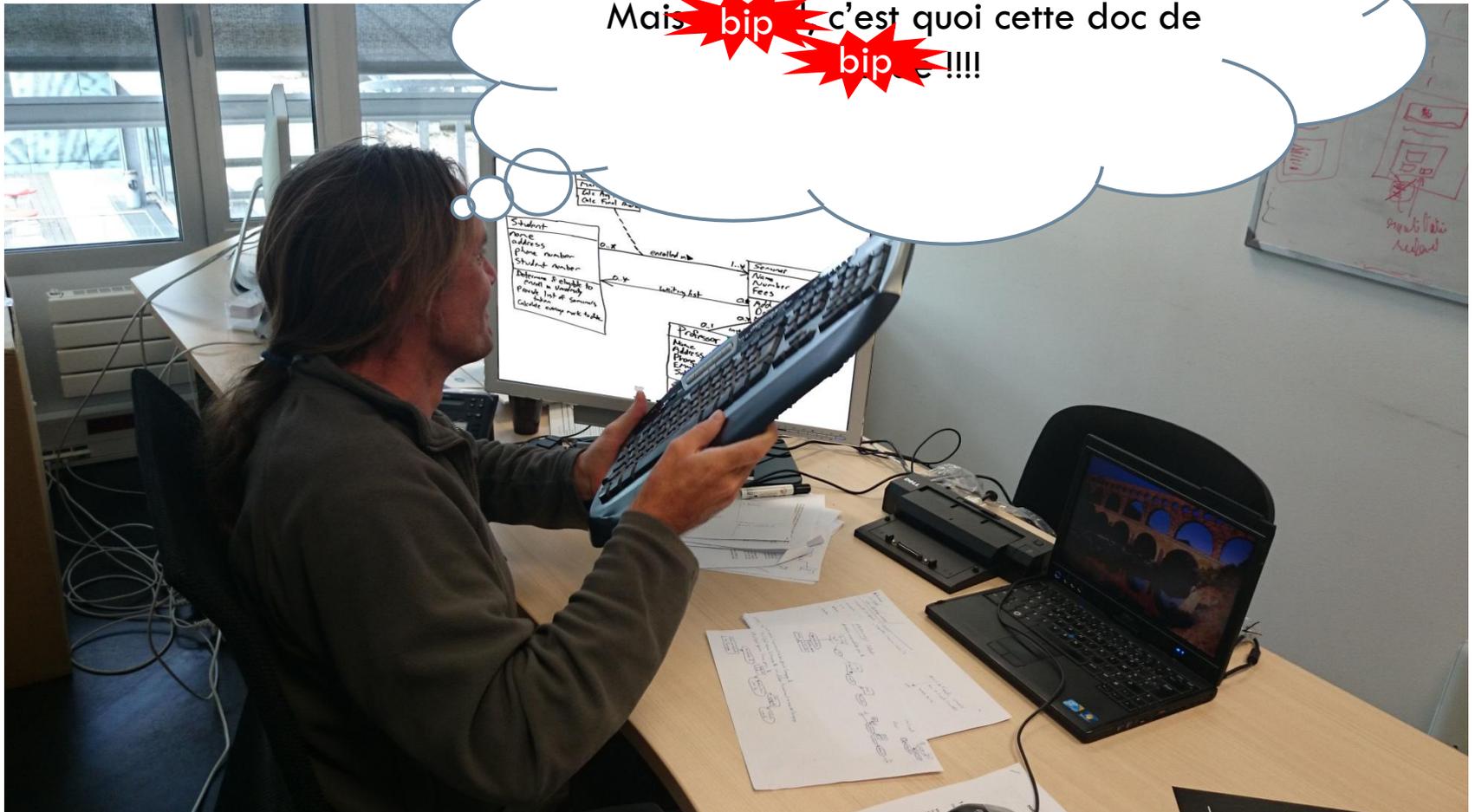


- Après 20 minutes passées à réorganiser le schéma, le résultat est...











Conclusions/Pistes de réflexions

25

- ❑ **Le code ne suffit pas, le code n'est pas tout**
- ❑ **Les modèles en amont** (quand ils sont fait) **ne suffisent pas** (pas de mise à jour)
- ❑ Les modèles ne sont pas (rarement ?) faits en amont, ou alors **incomplets**, ou **pas utilisé par la suite**
- ❑ **Outils de modélisation trop complexes**, pas adaptés à « la tâche »
- ❑ **Développeurs toujours avec simple IDE textuel**, alors que IHM a fait des progrès immenses en 30 ans !
- ❑ On a **besoin de modèles, mais aussi d'interactions évoluées pour « naviguer » dans le code/projet**

Conclusions/Pistes de réflexions

26

- On a **besoin de visualiser et d'interagir** avec tout ce qui est en relation avec le projet
 - Code
 - Modèles
 - Documentation
 - Compte-rendus de réunions
 - Photos
 - Enregistrements audios
 - ...
- Besoin de modèles différents à différents moments, **multi-vues...**
- On a besoin de **connecter le code et le « reste »**
- **Prise en compte du contexte**
 - qui fait quoi, où, avec quels périphériques, avec qui, ...

Equipe Carbon



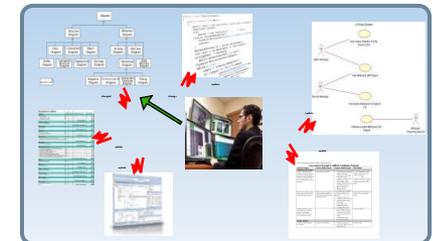
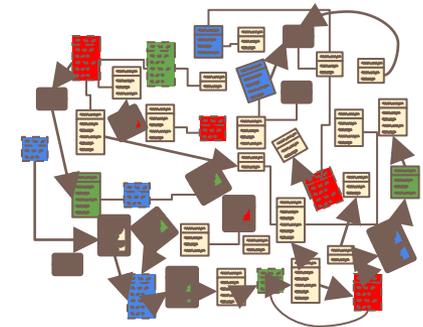
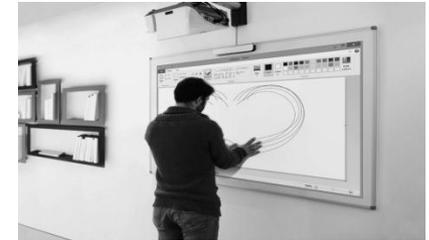
Trois axes



- Améliorer l'activité de modélisation logicielle
 - ▣ par de nouvelles techniques d'interactions

- Notations Visuelles
 - ▣ Rendre les diagrammes utilisables par les humains

- Améliorer les interactions entre sources d'informations de la conception logicielle





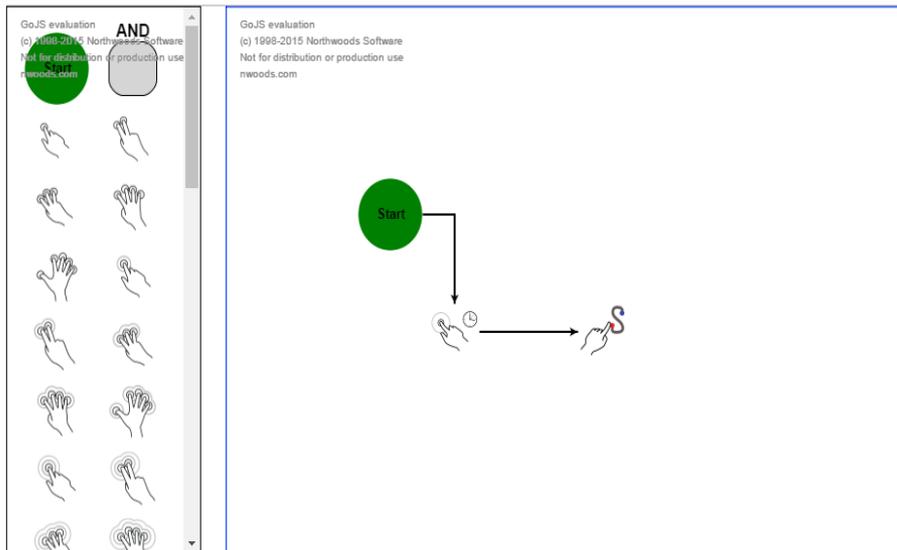
Improvement of Software Development Activity by New Techniques of Interaction

Jean-Claude Tarby

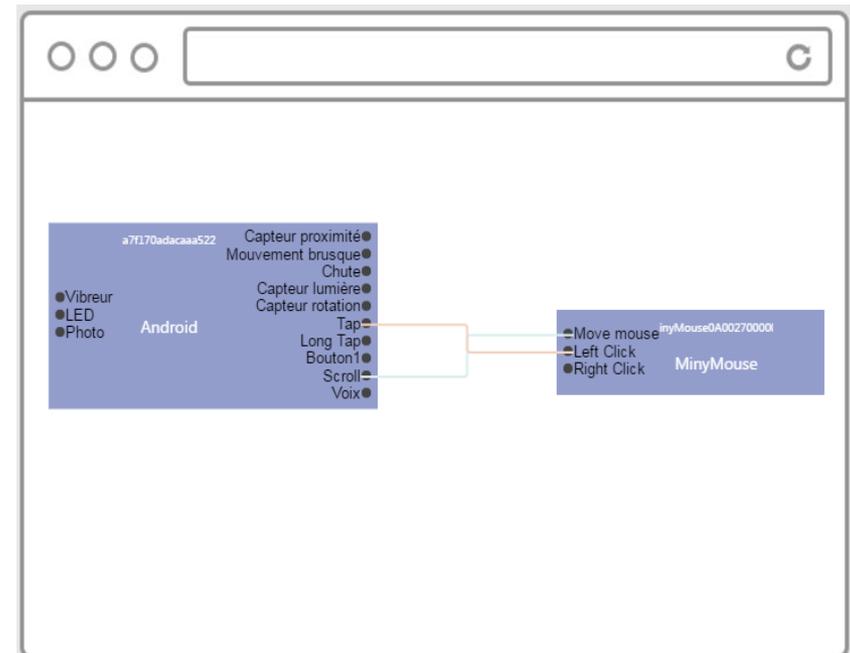
Mickael Duruisseau (doctorant)

Improvement of Software Development Activity by New Techniques of Interaction

- Interaction definition by composition
- Binding of interactions and actions
 - ▣ By user (?)...



Interaction definition

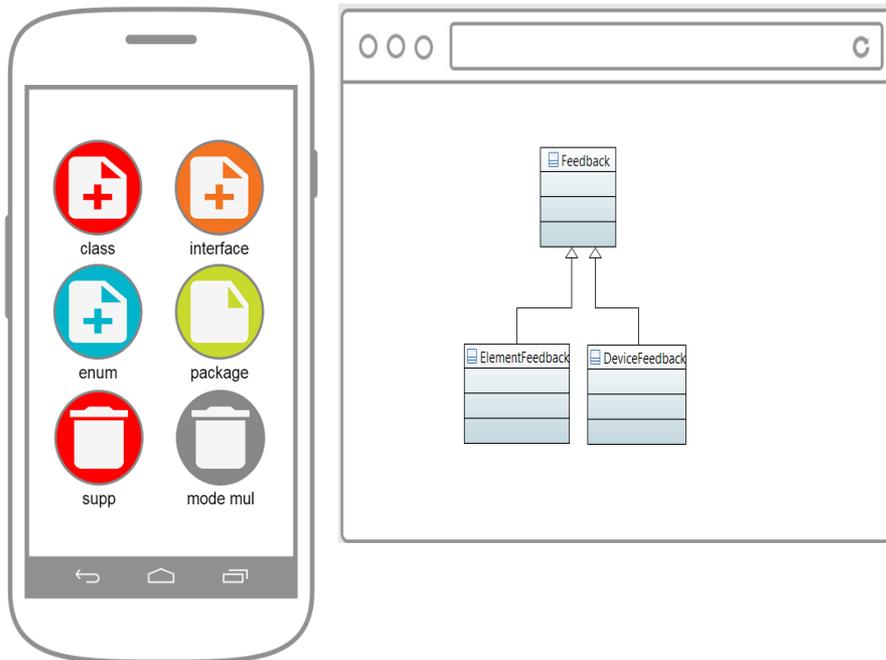


Bind between interaction and action

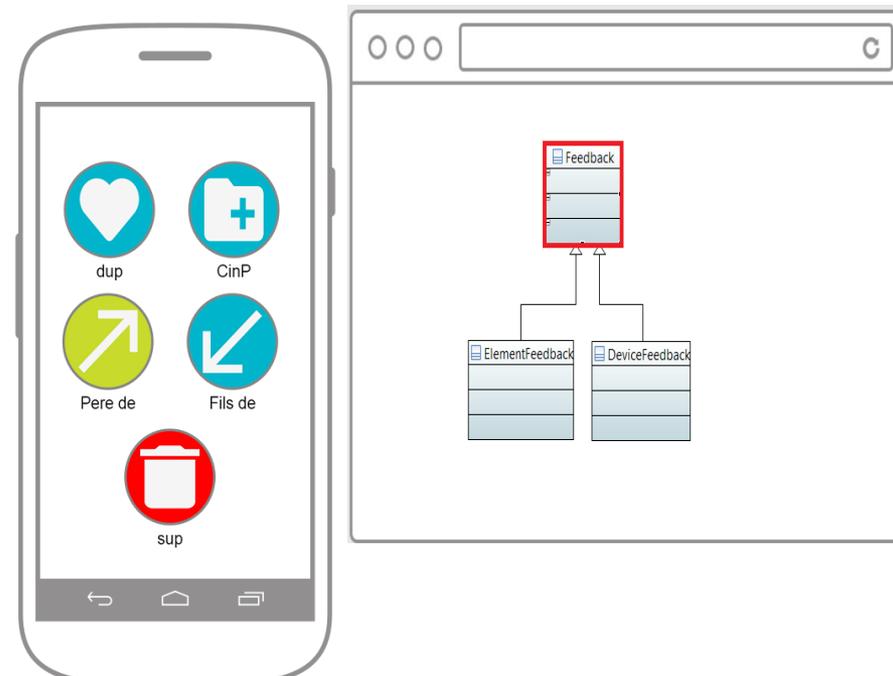
Contextual Remote

- Remote interactions
 - Interaction on digital tablet ... action on remote application

Context : Empty selection



Context : 1 element in selection





Improve interactions between heterogeneous information sources

Cedric Dumoulin



Improve interactions between heterogeneous information sources

Problem

The collage illustrates the problem of outdated documentation in software development. It features several key elements:

- UML Diagram Hierarchy:** A tree structure showing the relationship between different types of diagrams, from 'Diagram' down to specific types like 'Class Diagram' and 'Sequence Diagram'.
- Code Snippet:** A Java code snippet for a 'TopClientExample' class, showing network communication logic.
- CD Sales System:** A use case diagram showing interactions between a 'Band Manager', 'Record Manager', and 'Billboard Reporting Service'.
- Person at Computer:** A photograph of a person working at a computer workstation with multiple monitors.
- Data Table:** A screenshot of a table with columns for 'Parameters', 'Status', and 'Value', showing various system parameters.
- Compliance Program Table:** A screenshot of a table titled 'Gap Analysis Example 2: Affiliate Compliance Program' with columns for 'Current Status Program', 'Targeted Status Program', and 'The Issues'.

Red lightning bolts and the word 'update' are placed around the images, indicating that the documentation is outdated and needs to be updated. A green arrow points from the code snippet to the UML diagram, suggesting a change in the code that affects the diagram.

Software development produces a significant amount of information
Information are often related, changing one requires updating the others

Who has never facing an outdated documentation, with old screen captures, or where a menu entries don't exist anymore or have changed ?

Améliorer les interactions entre sources d'informations de la conception logicielle

Proposal
No more outdated document !

The collage illustrates various sources of information in software development:

- UML Diagrams:** A hierarchy of diagram types including Class, Component, Object, Profile, Composite Structure, Deployment, Package, Activity, Use Case, Interaction, and State Machine Diagrams.
- Code Snippet:** A Java code snippet for a `TopClientExample` class, showing network communication logic.
- Developer:** A person working at a workstation with multiple monitors.
- Data Table:** A table with columns for 'Proprietary criteria' and various data points.
- UML Use Case Diagram:** A diagram showing actors like 'Band Manager' and 'Record Manager' interacting with use cases like 'View Tasks Specific For My Bands CDs'.
- Compliance Report:** A document titled 'Cap Analysis Example 2 - Affiliates Compliance Program' with a table of criteria and status.

Red lightning bolts labeled 'update' point to the code, the developer, and the compliance report. A green arrow labeled 'change' points to the UML diagrams. Another red lightning bolt labeled 'update' points to the data table.

- We want to propose mechanisms:
 - ▣ to link information from different sources of information;
 - ▣ to measure the impact of a change in an information source;
 - ▣ to propagate changes;
 - ▣ to maintain the consistency of (heterogeneous) information sources as automatically as possible.



Améliorer la réutilisation des solutions à des problèmes de conceptions ou de réalisations déjà rencontrés

Idée en cours ...

Motivations

- Ma fonction doit lire un fichier dans un projet Eclipse. Facile ! J'ai déjà fait ce genre de truc plus de 10 fois ! Mais comment fait-on cela déjà ? Où ai-je déjà écrit ce genre de code ?
- Tous le code et la documentation sont là : sur mon disque et dans les dépôts GIT ! Comment rechercher la dedans ?

Motivations

- Un petit *grep* ou une *recherche textuelle* ?
 - Oui, mais quel mot clé utiliser ?

- Demander aux autres développeurs ?
 - Ils sont en vacances, et *pas disponible* !
 - Ceux qui sont là sont nouveau et *ne connaisse pas bien le projet* !
 - Ceux qui connaissaient bien le projet sont parties !

-

Motivations

- Demander sur un site d'entraide ou un moteur de recherche ?
 - Genre <http://stackoverflow.com/>
 - Mais mon problème est spécifique à mon projet, ils ne peuvent pas m'aider à retrouver mon code !

Constatations

- De nombreux problèmes sont posés et résolus lors de la conception et de la réalisation
 - Le code représente la solution
 - Il n'y a pas ou peu de trace du problème posé
- → on a perdu de la connaissance !
- Il est difficile de retrouver, de manière automatisée, des solutions à des problèmes de conceptions ou de réalisations déjà rencontrés

Questions

- Existe t-il d'autres mécanismes permettant de retrouver les solutions à des problèmes déjà posé ?
- Existe t-il un mécanisme qui permettrait d'exprimer les problèmes posés, et de les retrouver facilement (en reposant le problème d'une façon similaire) ?
- Comment exprimer les problèmes posés ?
- Comment relier le problème et les solutions ?

Pistes

- Documenter les problèmes dans un bug et exploiter les bugs (similaire a ce que fait Martin Monperrus) ?
- Annoter le code en exprimant le problème ?
- Exprimer les problèmes en langage (pseudo) naturel, et faire des recherches avec des moteurs comme Google ?

That's all Folks!

