Conception de logiciels interactifs sûrs avec DJNN

Stéphane Conversy, Sébastien Leriche, Mathieu Magnaudet, Célia Picard, Daniel Prun

ENAC - Université de Toulouse, France

Stéphane Conversy[1], *Sébastien Leriche*[1], *Mathieu Magnaudet*[1], *Celia Picard*[1], *Daniel Prun*[1]

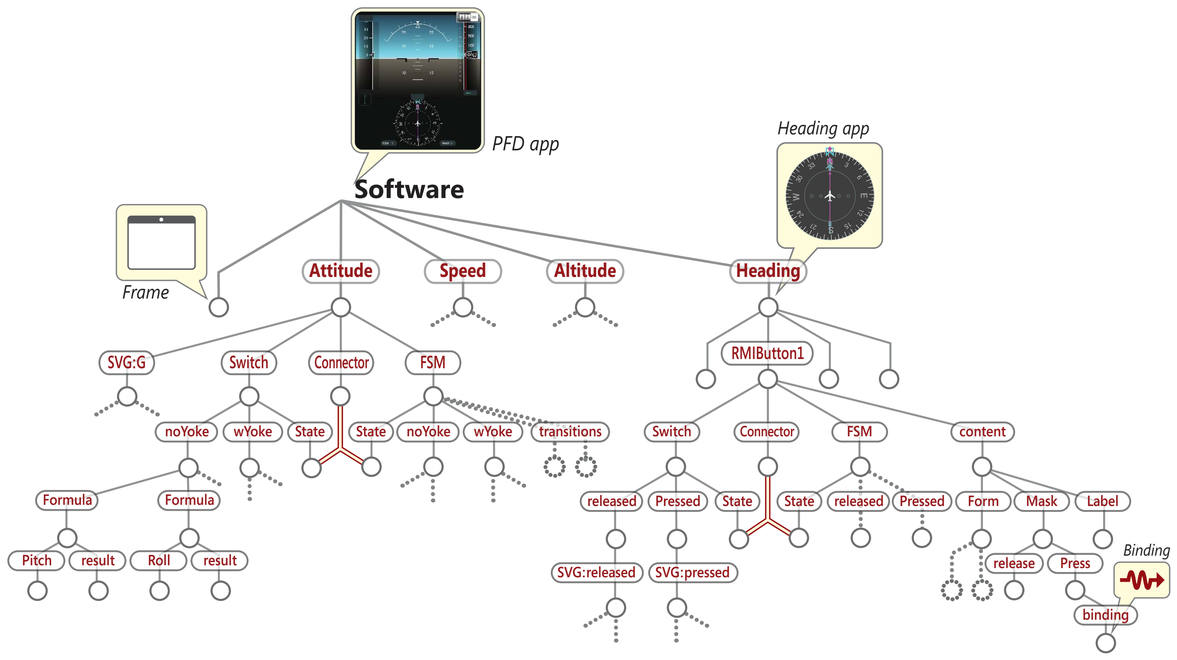
Contacts : (1) *{prenom.nom}@enac.fr* - Équipe Informatique Interactive *http://lii.enac.fr*

Conception de logiciels interactifs sûrs avec DJNN

Développement et vérification des logiciels interactifs

De nombreuses méthodes outillées ont été proposées pour le développement de logiciels sûrs et corrects d'un point de vue fonctionnel, pour autant les logiciels interactifs n'ont pas bénéficié de la même attention. Leur développement nécessite de manipuler de nombreux concepts hétérogènes (nature réactive, synchrone, flots de contrôle et de donnée, interface abstraite et concrète) et leur vérification repose principalement sur des évaluations de prototypes au cours de tests avec les utilisateurs finaux.

L'environnement **DJNN** propose un nouveau modèle pour les logiciels interactifs fondé sur un arbre de composants et un moteur d'exécution. Ce modèle permet d'envisager la vérification formelle de propriétés d'interaction.



DJNN : développement et exécution de logiciels interactifs

Une application **DJNN** est un **arbre de composants**.

Quatre composants élémentaires :

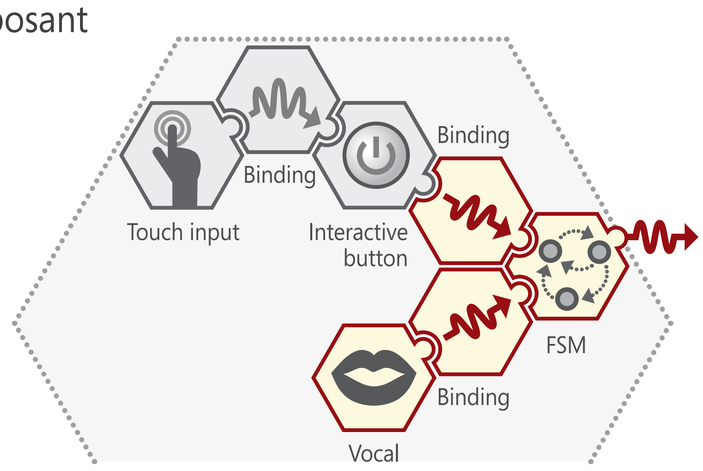
- **property** : stockage d'une valeur. La mise à jour de la valeur provoque l'activation du composant *property*.
- **binding** : création d'un couplage entre 2 composants. Quand la source est activée, alors la destination est activée.
- **assignment** : sur activation, transmission de valeur.
- **comparator** : sur activation, test d'égalité entre valeurs.

```
Component root {
  Frame f ("Hello World!", 0, 0, 500, 400)
  Exit quit (0, 1)
  f.close -> quit

  FillColor fc (200, 50, 50)
  Rectangle r (0, 0, 50, 100, 0, 0)
  f.height => r.height
}
```

Deux principes d'exécution :

- **activation** : les composants réagissent à des événements en effectuant une suite d'actions.
- **couplage** : une action effectuée par un composant peut être l'émission d'un événement déclenchant l'activation d'un autre composant



Résultats obtenus

- **SMALA** : un langage de description avec une syntaxe et une sémantique opérationnelle.
- Un **langage unique** pour exprimer des propriétés couvrant tous les niveaux de description (du modèle de tâches à l'interface concrète).
- **Vérification formelle** par l'étude statique de l'arbre des composants.
- **Applications** : cockpit d'hélicoptère hélectrique (Volta), interface de contrôle aérien pour le guidage au sol (MoTa, prototype de cockpit pour l'interaction multimodale (IoDe), FCU tactile ...

Objectifs de recherche

- Environnement de développement et de débogage :
  - **IDE adaptés** pour le développement et la mise au point d'applications interactives.
  - **Observateurs synchrones**
- Gestion de la **latence** (expression, vérification).
- Vers la **preuve formelle** de propriétés :
  - Traduction de Smala vers des outils supportant la vérification formelle (COQ, réseaux de Petri, event-B).

*Travaux élaborés dans le cadre du **Projet FORMEDICIS** (ANR-16-CE25-0007)*