

Modélisation et implémentation de l'architecture PAC à l'aide des patrons Proxy et Abstract Factory

Thierry Duval

► **To cite this version:**

Thierry Duval. Modélisation et implémentation de l'architecture PAC à l'aide des patrons Proxy et Abstract Factory. 2010. <inria-00534111>

HAL Id: inria-00534111

<https://hal.inria.fr/inria-00534111>

Submitted on 8 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

De la même façon, la présence d'une interface de contrôle va permettre au composant de présentation d'être indépendant des caractéristiques effectives du composant de contrôle.

À l'exécution, les facettes des composants PAC seront bien entendu des présentations, des contrôles, ou des abstractions et non pas des interfaces de ces composants.

LIENS AVEC LES PATRONS DE CONCEPTION

Notre démarche s'appuie également sur les patrons de conception Proxy, Fabrique Abstraite, et Singleton, dont on pourra trouver une description dans [6].

Utilisation du proxy

Le patron de conception proxy est ici utilisé pour pouvoir substituer un composant contrôle au composant abstraction auquel il est associé. C'est de cette façon que le contrôle pourra intercepter tous les messages à destination du composant abstraction, jouant ainsi son rôle de contrôleur d'accès et de maintien de cohérence avec son composant présentation associé.

Utilisation de la fabrique abstraite

Le patron de conception fabrique abstraite va permettre de regrouper dans un ou plusieurs composants dédiés l'ensemble des méthodes de création des objets d'une application. C'est ce qui va nous permettre de faire évoluer une application en remplaçant la fabrique initiale, qui crée des composants abstraction, par une fabrique qui va alors créer des composants de contrôle.

De la même façon, utiliser une fabrique abstraite pour la création des composants de présentation permet de faire évoluer facilement l'application interactive lors d'un changement de l'API graphique utilisée.

Utilisation du singleton

Le patron singleton n'est pas essentiel à notre méthode, mais il permet de s'assurer qu'on ne créera qu'une seule fabrique de composants de chaque type (abstraction, contrôle et présentation) pour une application donnée.

ILLUSTRATION DE LA DÉMARCHÉ

Nous illustrons cette démarche méthodologique à l'aide d'un exemple en Java en utilisant l'API Swing : il s'agit d'ajouter de l'interactivité à un noyau fonctionnel initialement non graphique et non interactif. L'accent est mis sur la nécessité de séparer efficacement le contrôle d'une part du noyau fonctionnel, et d'autre part de l'API graphique utilisée. La démarche permet également de remplacer très facilement l'API Swing par d'autres API graphiques comme par exemple SWT.

CONCLUSION

Cette méthodologie conduit donc à une séparation efficace des différentes facettes des composants PAC d'un système interactif. Elle peut aussi bien s'appliquer à la conception d'un nouveau système interactif qu'à l'évolution d'un système non interactif à rendre interactif.

La démarche est présentée en cours de Master 2 à l'Université de Rennes 1 depuis 2000 [5], elle a aussi été présentée sous forme d'un tutorial lors des conférences IHM 2005 et IHM 2009. Elle a également été utilisée dans le cadre du développement de plusieurs versions d'un logiciel commercial [2] dans un but d'optimiser la réutilisation de code.

BIBLIOGRAPHIE

1. Coutaz, J. Pac: An object oriented model for implementing user interfaces. *SIGCHI Bull.*, 19(2):37–41, 1987.
2. Degryny, F., and Duval, T. Utilisation du modèle pac-amodeus pour une réutilisation optimale de code dans le développement de plusieurs versions d'un logiciel commercial. In *IHM 2004: Proceedings of the 16th conference on Association Francophone d'Interaction Homme-Machine*, pages 149–156, New York, NY, USA, 2004. ACM.
3. Duval, T., and Nigay, L. Implémentation d'une application de simulation selon le modèle pac-amodeus. In *IHM 1999: Proceedings of the 11th conference on Association Francophone d'Interaction Homme-Machine*, pages 86–93, New York, NY, USA, 1999. ACM.
4. Duval, T., and Pennaneac'h, F. Using the pac-amodeus model and design patterns to make interactive an existing object-oriented kernel. In *TOOLS '00: Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 33)*, pages 407–418, Washington, DC, USA, 2000. IEEE Computer Society.
5. Duval, T., and Tarby, J.-C. Améliorer la conception des applications interactives par l'utilisation conjointe du modèle pac et des patrons de conception. In *IHM'06: Proceedings of the 18th International Conference of the Association Francophone d'Interaction Homme-Machine*, pages 225–232, New York, NY, USA, 2006. ACM.
6. Gamma E., Helm R., Johnson R., and Vlissides J. *Design Patterns : Elements of reusable Object-Oriented Software*. Addison-Wesley, 1995.
7. Nigay, L., and Coutaz, J. A design space for multimodal systems: concurrent processing and data fusion. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 172–178, New York, NY, USA, 1993. ACM.
8. UIMS 1992. A metamodel for the runtime architecture of an interactive system: the uims tool developers workshop. *SIGCHI Bull.*, 24(1):32–37, 1992.