IntrosPAC : un outil pour enseigner et comprendre PAC-Amodeus

Christophe Lachenal

Laboratoire CLIPS-IMAG
Domaine Universitaire, BP 53
38041 Grenoble Cedex 9 FRANCE
christophe.lachenal@imag.fr

Joëlle Coutaz

Laboratoire CLIPS-IMAG
Domaine Universitaire, BP 53
38041 Grenoble Cedex 9 FRANCE
joelle.coutaz@imag.fr

RESUME

Cet article présente IntrosPAC, un outil fondé sur la programmation par aspects, qui visualise l'architecture conceptuelle d'un système interactif à partir de l'analyse du code source supposé structuré selon les préceptes de PAC-Amodeus. La visualisation animée permet au programmeur débutant de comprendre la structure logicielle de son code, voire détecter les transgressions aux règles du modèle de référence. Utilisé pour l'enseignement des architectures logicielles en IHM, nous avons pu en apprécier les apports.

MOTS CLES : Modèle d'architecture, PAC-Amodeus, PAC, Programmation par aspects, Enseignement de l'IHM.

ABSTRACT

This article presents IntrosPAC, a software tool based on aspect oriented programming, which, from a PAC-Amodeus compliant source code, generates a graphic representation of the corresponding conceptual architecture. The graphic animation supports developers for understanding the structure of their code as well as for detecting violation of the principles of the reference architecture model. IntrosPAC has been successful in teaching software architecture modeling for interactive systems and in improving students understanding.

KEYWORDS : Software architecture modelage, PAC-Amodeus, PAC, Aspect Oriented Programming, Teaching HCI.

INTRODUCTION

PAC-Amodeus [8] propose une décomposition fonctionnelle normalisée applicable à tout système interactif. Comme tout modèle de référence, il intervient très tôt dans le processus de conception d'une solution implémentationnelle [4]. Eloigné du code concret, il est souvent difficile pour des développeurs, et notamment pour des étudiants, d'en comprendre la nature et l'intérêt.

Pour répondre à ces difficultés, nous proposons IntrosPAC, un outil fondé sur la programmation par aspects, qui visualise le cheminement des messages au sein d'un logiciel structuré selon les précepts de PAC-Amodeus. Cette visualisation animée permet au programmeur débutant de comprendre la structure logicielle de son code, voire détecter les transgressions aux règles du modèle de référence. Après un bref rappel de PAC-Amodeus, nous présentons IntrosPAC en détail : principes, apports et limites actuelles ainsi que les retours que nous avons eu sur son utilisation par des étudiants et des enseignants.

PAC-AMODEUS : PRINCIPES

Comme le montre la figure 1, PAC-Amodeus reprend le découpage fonctionnel du modèle Arch [10] et structure le contrôleur de dialogue en agents PAC [3]. Le découpage Arch met en valeur les fonctions essentielles d'un logiciel d'interaction et notamment le Contrôleur de Dialogue, point sensible de la conception logicielle. Il permet également, via les deux adaptateurs, de définir des points de résolution en cas d'incompatibilité de styles [2], et des lieux de réutilisation de code (par exemple, les services des boîtes à outils). La structuration du contrôleur de dialogue en agents PAC rend explicites les interactions à plusieurs fils et fournit le bon niveau de granularité en vue de la mise en œuvre.

La facette Abstraction (A) d'un agent PAC modélise sa compétence opératoire tout comme le Noyau Fonctionnel représente le fonctionnement conceptuel du système. Elle est généralement reliée à des objets conceptuels de l'Adaptateur de Noyau Fonctionnel. La facette Présentation (P) d'un agent définit son rendu vis-à-vis de l'utilisateur. Elle est reliée à un (ou plusieurs) objets de présentation et peut, si besoin est, maintenir des informations d'état en vue de contrôler son comportement perçu par l'utilisateur. Comme dans PAC, la facette contrôle d'un agent est un transformateur de formalisme entre les deux facettes horizontales qu'il relie et intervient dans la chaîne verticale des processus d'abstraction et de concrétisation.

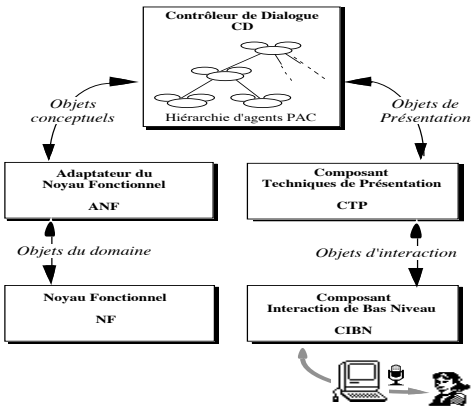

Figure 1. *Le modèle hybride PAC-Amodeus.*

La décomposition modulaire du Contrôleur de Dialogue en agents PAC se pratique par l'application systématique d'heuristiques et de motifs architecturaux. Ces heuristiques et motifs s'appuient sur une connaissance concrète facile à comprendre les spécifications externes de l'Interface Homme-Machine du système. On trouvera dans [8] le détail des justifications de ces règles et dans [9], un exemple complet de leur application.

Notre expérience dans l'enseignement des architectures logicielles des systèmes interactifs indique que les étudiants n'ont pas de difficultés à identifier, pour un système interactif donné, la hiérarchie d'agents PAC du Contrôleur de Dialogue. Par contre, nos étudiants ont du mal, dans le processus de réification, à passer de l'architecture conceptuelle à l'architecture implémentationnelle de leur système interactif et à maintenir les liens entre les deux représentations par suite de modifications du code. Ce constat nous a amené à réaliser IntrosPAC.

INTROSPAC REQUIS ET SOLUTION

IntrosPAC répond à deux requis essentiels: permettre à nos étudiants de comprendre les liens entre l'architecture conceptuelle et le code, instrumenter le code sans toutefois le modifier.

En réponse à ces requis, nous avons fait les choix suivants :

- Représentation graphique de l'architecture conceptuelle construite à partir du code. Cette représentation est analogue aux dessins pratiqués sur papier avant le codage elle reprend les « hickey » de PAC. Elle est animée, montrant le cheminement des flux de données au sein de l'architecture tandis que le code s'exécute.
- Implémentation d'IntrosPAC selon les principes de la Programmation par Aspect (Aspect Oriented Programming).

La programmation par aspect a été introduite pour exprimer à part les aspects transversaux d'un logiciel, typiquement des services qui se répètent dans la plupart des composants d'un logiciel, par exemple le contrôle d'accès ou le traçage d'exécution [5]. L'utilisation d'aspect permet de concentrer l'expression de ces services en un composant externe appelé au moment voulu comme s'il avait été explicité au sein du code source.

Ainsi IntrosPAC est un aspect il agit comme une sonde sans que le programme original, du point de vue du programmeur, soit pollué par des instructions d'inspection. IntrosPAC utilise les conventions de nommage des facettes des agents PAC et les mécanismes d'héritage pour venir se greffer sur un composant supposé respecter les règles de composition et d'échanges du style PAC. En l'occurrence, le composant qui nous intéresse est le Contrôleur de Dialogue d'un système interactif. Les règles du style PAC se résument ainsi :

- Les agents sont composés en arborescence.
- La facette Contrôle d'un agent connaît les facettes Abstraction, Présentation de cet agent, de même que les fils de l'agent, s'ils existent.
- La communication entre facettes est initiée soit par l'action de l'utilisateur sur une facette Présentation, soit par une action de l'adaptateur du noyau fonctionnel (ANF) sur une facette Abstraction.

IntrosPAC crée une pile d'appels qui lui permet de connaître l'objet appelant et l'objet appelé, la méthode appelée ainsi que la valeur des paramètres. Si l'appelant n'est pas connu et que l'appelé est une facette de type Présentation, alors il s'agit d'une action utilisateur. Par contre, si l'appelé est une facette de type Abstraction, il s'agit de l'exécution d'une callback de l'ANF (ou du NF, en l'absence d'ANF). Cette facette Abstraction est alors l'appelant.

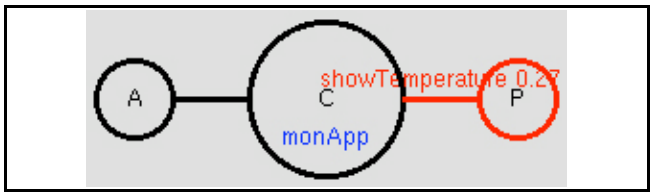

Figure 2 : Visualisation par IntrosPAC de l'échange de message au sein d'un agent entre ses facettes C (Contrôle) et P (Présentation) appel de la méthode showtempérature avec le paramètre 0.27).

À partir des informations sur les relations appelant-appelé et des règles du style PAC, IntrosPAC construit la visualisation graphique animée PAC correspondant au code. Comme le montre la figure 2, chaque facette d'un agent est visualisée sous forme d'un cercle et les références entre les facettes sous forme de liaisons. Le type de l'agent est affiché au cœur de la facette Contrôle C et le nom de la méthode appelée ainsi que la valeur de ses paramètres sont indiqués sur la liaison.

Lorsqu'une facette source appelle une méthode d'une facette destinataire, la source passe en rouge, puis la liaison entre les facettes, enfin la facette destinataire. Pour traduire le flux de données, la facette source passe ensuite en noir, puis la liaison, enfin la facette destinataire. Pour exprimer l'effet de flux, IntrosPAC utilise, entre chaque appel, un délai d'exécution paramétrable. Ainsi, à partir de la représentation graphique animée, le programmeur peut étudier les appels entre les facettes et analyser les effects de ces appels.

IntrosPAC est réalisé en tcl/tk selon le style à objet. Il utilise les mécanismes de traçage d'exécution de tcl pour intercepter les appels de méthodes. IntrosPAC, de par son principe de fonctionnement, n'est pas lié à un langage d'implémentation. Il peut facilement être écrit pour un autre langage, par exemple Java avec AspectJ [1].

APPORTS D'INTROSPAC

IntrosPAC est une aide pour :

- La conception d'un composant logiciel selon le modèle PAC : le programmeur « voit » de manière concrète le lien entre l'architecture conceptuelle et le code.
- L'évolution de code : la représentation graphique et les flux de données facilitent l'identification des points d'introduction de modifications.
- La vérification de la conformité entre le modèle et le code : un composant non conforme au style PAC ne peut pas se dessiner sous forme d'une arborescence d'agents PAC. Au sein d'un agent, des appels entre facettes non conformes aux règles, se traduisent par des liaisons non conformes au « chickey » PAC.
- La mise au point : l'affichage des noms de méthodes et des valeurs de paramètres avec leurs résultats offrent une technique de mise au point du code proprement dit.

Ces propriétés font d'IntrosPAC un outil utile et efficace pour l'enseignement de l'architecture des systèmes interactifs : utile pour l'étudiant qui comprend mieux la nature de PAC, efficace pour l'enseignant pour évaluer l'architecture conçue par l'étudiant.

RETOUR SUR L'UTILISATION D'INTROSPAC

IntrosPAC a été utilisé par des enseignants en IHM (au nombre de 3) auprès d'étudiants (environ 80 au total) en informatique (niveau bac+5) durant l'année 2002-2003. Les étudiants non pas été contraints à utiliser IntrosPAC. L'outil leur a simplement été donné au début du projet et ils ont été libres de l'utiliser ou non. IntrosPAC a rencontré un certain succès auprès des étudiants et des enseignants qui s'en sont servis tout au long du projet. Nous relatons dans cette partie les retours informels que nous avons eu sur l'utilisation d'IntrosPAC.

Une des clefs du succès d'IntrosPAC auprès des étudiants est qu'il ne nécessite pas de modification dans le code du programme mais le respect de quelques régles de nommage simples. Une question posée à plusieurs reprises par les étudiants montre le coté « magique » du fonctionnement d'IntrosPAC : « IntrosPAC va-t-il fonctionner avec mon programme ? ». D'ailleurs une autre question du même type est souvent revenue : « Que dois-je modifier pour utiliser IntrosPAC ? » . Ainsi on comprend que les étudiants ne se sont pas rendu compte que le respect des consignes données en cours pour programmer les agents PAC est la seule condition nécessaire pour le fonctionnement d'IntrosPAC. Ces consignes n'ont d'ailleurs fait l'objet d'aucun commentaire et ont été vues comme une aide pour le codage. L'utilisation d'IntrosPAC leur est donc apparue transparente.

De leur coté, les enseignants ont apprécié l'apport graphique de la visualisation de l'architecture qui a permis au dialogue de s'instorer plus facilement avec les étudiants et a servi de support aux remarques. Ainsi, IntrosPAC a permis une meilleure compréhension des problèmes rencontrés par les étudiants vis-à-vis de l'architecture PAC. Notamment, le choix de la facette (abstraction, contrôle ou présentation) au sein de laquelle doit avoir lieu une conversion d'une donnée d'une unité vers une autre. Enfin, IntrosPAC a aussi été une aide pour le suivi et la notation des étudiants. Cependant malgré le succès rencontré, IntrosPAC montre quelques limites.

LIMITES ET PERSPECTIVES

La première limite d'InstrosPAC est qu'il ne permet pas dans sa version actuelle la visualisation de l'évolution de la hiérarchie des agents : InstrosPAC ne trace pas les constructeurs ni les destructeurs. Ainsi, il ne peut pas être utilisé sur une application qui crée et détruit dynamiquement des agents.

La seconde limite tient aux temps de réponse : si de très nombreux appels traversent la hiérarchie (notamment des callback système) alors la temporisation pose problème. Pour remédier à cela, l'ajout d'une interface de contrôle (stop, avance rapide) est envisageable et fait partie des perspectives de travail retenues. Cependant avec les sys-

tèmes interactifs type «□temps réel□, cette solution n'est pas viable (trop de messages échangés en un lapse de temps trop court).

Une perspective plus pédagogique à ce travail est la possibilité d'utiliser les règles du style PAC pour proposer un environnement de conception se plaçant en amont d'IntrosPAC et pouvant être utilisé dès le début de l'apprentissage du style d'architecture PAC. Cet outil pourrait permettre de manière graphique□

- la création d'agents PAC (composés de facettes C, A ou P suivant les besoins),
- la création des relations de parenté entre ces agents (mise en place de la hiérarchie),
- la mise en place de la communication entre les facettes (facilitant notamment la mise en œuvre de l'échange de messages dans une hiérarchie importante).

L'utilisation d'un tel outil en complémentarité avec IntrosPAC devrait faciliter la compréhension du style d'architecture PAC et le passage du modèle théorique au modèle implementationnel qui sont les deux problèmes clefs recontrés par les étudiants.

CONCLUSION

IntrosPAC traduit sous forme graphique le lien entre un code exécutable et son architecture conceptuelle dont il permet d'étudier le comportant à l'exécution. La simplicité de ses principes autorise une implémentation dans tout langage pourvu des mécanismes nécessaires à la programmation par aspects. Ces mécanismes, on le rappelle, garantissent, du point de vue du programmeur, l'intégrité du code source. Notre idée peut être étendue à d'autres modèles d'architecture comme MVC [6]. Dans le cas de MVC, le patron «□observeur-observable□ aidant à l'introspection.

IntrosPAC a été utilisé en 2002-2003 par des enseignants en IHM auprès d'étudiants en informatique (niveau bac+5). Les étudiants ont apprécié la simplicité d'utilisation et la valeur ajoutée de l'outil. Comparativement aux années antérieures, l'apport d'IntrosPAC s'est traduit par une meilleure implication des étudiants mais aussi par des résultats aux examens en net progrès.

Une vidéo de l'utilisation d'IntrosPAC est disponible à l'URL□□http://iihm.imag.fr/lachenal/IntrosPAC/IntrosPAC.html

BIBLIOGRAPHIE

1. http://www.aspectj.org.
2. Bass L, Clements P et Kazman R., *Software Architecture in Practice*, Addison Wesley Publ., ISBN 0-201-19930-0, 1998.
3. Coutaz J. *PAC: an Implementation Model for Dialog Design*. In Proc. Interact'87, (Stuttgart, Sept. 1987), H-J. Bullinger, B. Shackel ed., North Holland, pp. 431-436.
4. Coutaz J, Nigay L. Chapitre 7 : *Architecture logicielle conceptuelle des systèmes interactifs*. 38 pages, Analyse et Conception de l'Interaction Homme-Machine dans les systèmes d'information, Kolski Ed., Hermes Publ., 2001, pp.207-246.
5. Gregor K et al. *Aspect-Oriented Programming*. In proceedings of ECOOP'97, LNCS 1241, Springer-Verlag, pp. 220-242, Juin 97.
6. Krasner et al. *A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltak-80*. JOOP'88. pp. 26-49. 1988.
7. Nigay L, Coutaz J. *Building user interface : Organizing software agent.* ESPRIT'91. pp. 707-719. Novembre 1991.
8. Nigay L. *Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales*. Thèse de doctorat Informatique préparée au Laboratoire de Génie Informatique (IMAG), Université Joseph Fourier, 28 janvier 1994, 315 pages.
9. Nigay L, Coutaz J. *Software architecture modelling: Bridging two worls using ergonomics and software properties*. Formal Methods in Human-Computer Interaction, Springer-Verlag: London Publ. pp. 49-73
10. *The UIMS Workshop Tool Developers : A Metamodel for the Runtime Architecture of an Interactive System*, SIGCHI Bulletin, 24, 1, pp. 32-37, Janvier 1992.