Incohérences au niveau de la modélisation

IBM *Software Group*

Claudio Grolimund: „Il importe d'identifier et éliminer les incohérences à l'échelle des modèles, a souligné le professeur Alexander Egyed dans le cadre de la conférence Rational Software Developper Platform 2009. En cette occasion, il a aussi présenté une procédure permettant de vérifier rapidement, précisément et automatiquement les incohérences apparues au niveau des modèles suite à une modification. Après son habilitation, Monsieur Egyed a travaillé sept ans dans le développement logiciel de projets de recherche industriels pour le compte de la Techknowledge Corporation aux Etats-Unis avant d'intégrer l'University College de Londres. Aujourd'hui à la tête de l'Institut für System Engineering & Automation de l'Université Johannes Kepler de Linz, le professeur Egyed s'occupe principalement du développement de logiciels à base de modèles. Lors d'un bref entretien, il nous a présenté sa démarche de travail en expliquant pourquoi les incohérences demeurent souvent un problème considérable y compris pour les développeurs les plus expérimentés."

Alexander Egyed: „Le propre des incohérences est qu'elles ne constituent pas en fait un véritable problème et présentent aussi bien des avantages que des inconvénients. Au niveau des logiciels, elles apparaissent simplement du fait qu'on tente de modeler différemment des points de vue distincts, un peu comme le dessin d'une maison qu'on va représenter sous différents angles, par exemple avec une vue de front et une vue de côté. On peut également penser à d'autres schémas tels qu'une représentation des conduits de câbles et de canalisations où règnent également des règles de cohérence bien définies. La hauteur d'une maison devrait ainsi être la même dans la vue latérale et dans la vue de face, faute de quoi de quoi ces deux représentations ne peuvent coïncider. Des règles de cohérence pour différentes perspectives peuvent donc également être établies dans le génie civil. La situation est similaire dans le

développement de logiciels où nous avons différents points de vue pouvant être décrits de différentes manières, et c'est précisément pour cela que nous devons nous occuper explicitement du problème de la consistance. On notera par ailleurs que les différences dans la modélisation n'entraînent pas que des inconvénients, et elles peuvent aussi tout à fait être intentionnelles. Dans un modèle, l'idée n'est en effet pas de décrire l'ensemble du produit en une fois, mais uniquement les éléments pour lesquels un argument ou une analyse s'avèrent déterminants. Si différents acteurs décrivent chacun une partie d'un système, il faut néanmoins veiller à ce que les différentes composantes puissent au final à nouveau s'agencer."

Claudio Grolimund: „Au cours de notre entretien, Monsieur Egyed a expliqué comment on peut découvrir, vérifier et éliminer rapidement les incohérences présentes après une modification."

Alexander Egyed: „Des règles de cohérence rudimentaires peuvent déjà entraîner des erreurs hautement complexes dans les modèles tels que je les ai évoqués à l'exemple de la maison. L'examen de cohérence consiste à décrire autant que possible toutes les règles importantes de manière à pouvoir identifier automatiquement les éventuels dysfonctionnements. De très nombreux outils d'automatisation ne font rien d'autre que d'évaluer de différentes manières ces règles de cohérence dans les modèles et de générer des valeurs correspondantes. Le résultat peut être soit cohérent, autrement dit correct, ou incohérent, c'est-à-dire erroné. Pour l'essentiel, le procédé est simple et les problèmes apparaissent entre autres lorsque l'on a affaire à un nombre de règles trop important ou si l'envergure du modèle est très grande, auquel cas la vérification peut être chronophage malgré l'automatisation. La procédure automatisée est cependant de loin préférable au traitement manuel, même s'il est tout sauf optimal que d'attendre longtemps les retours d'information suite à une erreur. Le problème apparaît fréquemment dans les environnements actuels de programmation, notamment dans Eclipse. On écrit une chaîne de signes, par exemple une pièce de code, et quelque chose apparaît en souligné afin de signaler une erreur. On n'obtient néanmoins qu'en partie des propositions de solution, et il va parfois falloir des heures pour obtenir ce feed-back dans la modélisation. D'aucuns renoncent à engager de tels processus de longue haleine, si bien qu'on va alors éventuellement travailler avec des erreurs

passées inaperçues et qui en entraîneront de nouvelles de leur côté. Si le problème n'est remarqué que tardivement, il sera difficile de se souvenir de ce que l'on voulait dire lorsqu'on a écrit la chose, et il faudra alors réfléchir pour savoir comment éliminer cette erreur ainsi que toutes celles qu'elle a entraînées. Jusqu'ici, nous n'avons en fait parlé que de l'identification, et seulement en partie de l'élimination des incohérences. Or, cette élimination représente le problème le plus important car il faut comprendre précisément où réside l'origine de l'incohérence. Une erreur que je fais peut avoir un impact sur cinq perspectives différentes et générer par conséquent cinq notifications d'erreurs. Cela ne signifie pas que j'ai fait cinq erreurs, et en l'occurrence il s'agit plutôt d'une erreur unique ayant plusieurs impacts. Comprendre cela, et, partant, conclure à l'effet d'une seule cause est un problème encore complètement irrésolu."

Claudio Grolimund: „Dans le cadre de sa présentation, Monsieur Egyed a évoqué une solution permettant d'évaluer rapidement, de façon précise et automatique, la cohérence d'un modèle soumis à différents changements. Il a également expliqué comment ce modèle fonctionnait et en quoi cette solution se différenciait des autres."

Alexander Egyed: „J'apprécie beaucoup l'approche consistant à tenter d'analyser une règle de cohérence. Quelqu'un va donc écrire ce qu'elle signifie et nous vérifions alors simplement ce qui se passe avant de chercher comment gérer au mieux les changements sur la base des résultats obtenus, la méthode fonctionnant aussi bien pour l'identification que pour la résolution d'erreurs. Le problème réside dans le fait que l'analyse automatique d'une règle de cohérence est un processus extrêmement complexe bien qu'il ne serve pour l'essentiel qu'à la description et à la découverte d'incohérences simples. Nous procédons quant à nous de manière exactement inverse et proposons une solution radicalement nouvelle où nous observons la règle de cohérence sans du tout essayer de la comprendre. Nous nous contentons de l'entourer d'un petit wrap qui n'a d'autre tâche que de montrer comment la règle réagit au modèle. Peu importe pour quelle raison celle-ci vérifie un élément du modèle, l'important est plutôt pour nous qu'elle soumette précisément cet élément de modèle à un examen de cohérence. Si elle a vérifié un élément de modèle durant l'évaluation et que ce dernier vient à changer, il importera alors de redéfinir la règle. La technologie que nous avons développée pour ce faire se base sur un profileur de modèles dont l'unique tâche

consiste à observer le vérificateur System C. Cette information nous fournit en principe tout ce dont nous avons besoin pour savoir quand et comment un changement a un impact sur un modèle et dans quelle mesure ceci peut influer sur les incohérences. Le grand avantage de cette approche est que l'information va non seulement nous permettre de découvrir comment agit une incohérence, mais également de décider où éliminer des erreurs. Revenons maintenant à l'exemple évoqué précédemment où j'expliquais qu'il convient de faire une différence entre la cause et l'identification de l'erreur, l'incohérence n'étant que l'annonce indiquant qu'une erreur est survenue quelque part. Cela signifie que les endroits où on corrige des incohérences se réfèrent précisément à l'information du profileur de modèles qui a vérifié auparavant cette règle de cohérence. On peut ainsi trouver les dix ou vingt éléments de modèles qu'il convient d'étudier en cas de manifestation d'une incohérence dans un modèle relativement important comportant de nombreux éléments."

Claudio Grolimund: „Monsieur Egyed a encore évoqué pour terminer sa vision de l'avenir du développement logiciel basé sur les modèles, ainsi que les possibilités apparaissant dans le domaine de la garantie de la consistance."

Alexander Egyed: „L'avenir du développement logiciel à base de modèles réside à mon avis dans leur intégration au niveau du processus. La phase de modélisation est évidemment très importante pour le développement logiciel, et plus on met de temps à remarquer la présence d'une erreur, plus il devient onéreux d'y remédier. Intervenir dès l'analyse initiale implique un certain coût, mais l'élimination de la même erreur lors du test ou durant l'intégration dans le cadre de la maintenance coûte environ trente fois plus cher. Si on ne s'aperçoit d'une erreur que sur place auprès du client et que le logiciel n'est pas utilisable, les coûts pour y remédier peuvent alors être deux cents fois plus élevés. La modélisation fait partie de cette phase de développement où l'élimination des erreurs est la moins chère, pour autant que l'élimination n'ait pas pu se faire lors de l'analyse initiale."

Claudio Grolimund: „Nous remercions Monsieur Egyed pour cet entretien et profitons de l'occasion pour mentionner les liens de notre site Web sur la question du développement logiciel à base de modèles.“