La traçabilité dans les lignes de produits logiciels

Nicolas Anquetil*, Joost Noppen*, Ismênia Galvão**

** Équipe ASCOLA*
Département Informatique,
École des Mines de Nantes
La Chantrerie
4, rue Alfred Kastler. B.P. 20722
F-44307 NANTES Cedex 3

{nicolas.anquetil,johannes.noppen}@emn.fr

*** Department of Computer Science, University of Twente*
P.O. Box 217, 7500 AE Enschede, The Netherlands

galvaoi@ewi.utwente.nl

RÉSUMÉ. *La traçabilité est la possibilité de relié entre eux tous les artefacts produits lors du développement d'un logiciel, des besoins jusqu'au code. Dans le cadre des lignes de produits logiciels, le problème de la traçabilité est accrut par certaines spécificités de cette méthode de développement. Dans cet article nous discutons les problèmes liés à la traçabilité dans les lignes de produits logiciels et proposons une catégorisation des liens de traçabilité. Un modèle de gestion de la traçabilité en présence d'incertitude est aussi présenté.*

ABSTRACT. *Traceability is defined as the possibility to correlate the various artefacts (from requirement to source code) generated when developing a software. In the context of software product line development, the problem is made even harder due to some characteristics specific to this software development approach. In this paper we discuss these problems specific to traceability for software product line development. We propose a taxonomy of traceability links. We also present a model to manage traceability of design decisions in the presence of uncertainty.*

MOTS-CLÉS : *traceability, ligne de produits logiciels, variabilité, décision de projet, incertitude*

KEYWORDS: *traceability, software product line, variability, design decision, uncertainty*

1. Introduction

Le projet européen AMPLE (*Aspect-oriented Model-driven Product Line Engineering*, http://www.ample-project.net/) se propose de combiner les techniques de pointes en développement par aspects (AOSD) et en développement de logiciel dirigée par les modèles (MDD) pour avancer l'état de l'art en lignes de produits logiciels (SPL). Entre autre chose, cela suppose de pouvoir traiter adéquatement la traçabilité dans les lignes de produits, et cette propriété est une partie importante de la recherche dans le projet AMPLE. Cet article résume nos recherches sur la traçabilité dans les lignes de produits logiciels. Les résultats présentés ici sont le fruits des travaux de tous les chercheurs du projet AMPLE et non pas seulement des auteurs du présent article.

Nous commencerons (Section 2) par faire un rappel sur la traçabilité en génie logiciel et aussi dans le cadre plus restreint des lignes de produits logiciels. Puis (Section 3) une taxonomie de traçabilité sera présentée qui permet de structurer le traitement des liens de traçabilité. Un méta-modèle pour le stockage de ces liens est défini, et un exemple d'outil logiciel de gestion de ces liens sera présenté (Section 4). Enfin l'utilisation de la traçabilité dans le cadre de l'incertitude liée aux décision de projets sera discutée (Section 5) avant de présenter nos conclusions (Section 6).

2. Traçabilité dans les lignes de produits logiciels

2.1. *Traçabilité en génie logiciel*

On peut citer plusieurs définition de la traçabilité (traductions des auteurs):

– c'est la capacité de décrire et suivre le cycle de vie des besoins vers l'avant ou l'arrière (Gotel, Finkelstein, 1994).

– c'est la capacité d'établir des liens entre plusieurs produits du développement de logiciel, spécialement des produits reliés entre eux par une relation prédécesseur/successeur ou maître/subordonné (Geraci, 1991).

En résumé, le premier objectif de la traçabilité fut la capacité de suivre l'évolution des besoins jusqu'au code (et du code jusqu'au besoins qui l'ont engendré) au travers des différents produits intermédiaires. Bien que cela n'apparaisse pas dans les deux définitions ci-dessus, cet objectif s'est ensuite élargit, par exemple, pour inclure d'autres relations comme la similitude entre besoins ou entre composants.

La traçabilité peut répondre à plusieurs nécessité:

– Vérification de la complétude d'implantation des besoins,

– Aide à l'analyse d'impact quand un artefact abstrait est modifié (quels artefacts plus concrets seront possiblement affectés?)

– Répondre à des nécessités contractuelles

2.2. *Traçabilité et lignes de produits*

Les lignes de produits sont une approche de développement logiciel où l'on définit une architecture générale pour plusieurs application dans un domaine d'application donné. L'architecture générale prévoit des points de variation (et quelles peuvent être les variations) qui permettent de générer différentes application à partir de cette architecture unique.

Le processus de développement d'applications dans cette approche est normalement divisé en 2 étapes (v. Figure 1):

– L'ingénierie du domaine où est développé l'architecture générale à partir d'une analyse du domaine d'application et de ses besoins traditionnels, des variations prévues, et de composants logiciels réutilisables.

– L'ingénierie des produits où sont dérivées les applications individuelles à partir des besoins spécifique du client, à l'intérieur des limites autorisées par l'architecture générale, c'est à dire en choisissant les options pertinentes aux points de variation prévus.

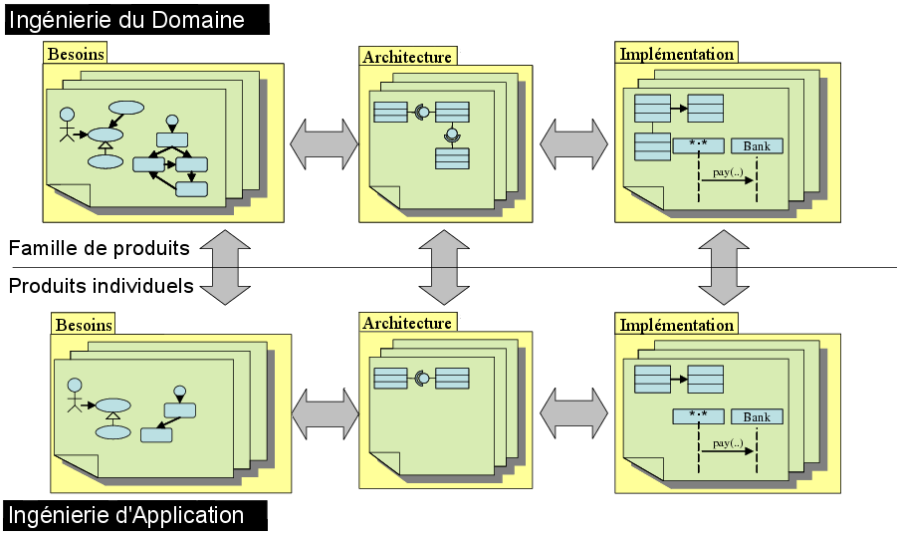

Figure 1. *Processus de développement d'applications avec l'approche des Lignes de Produits Logiciels. Les flèches grises représentent les relations de traçabilité les plus courantes*

Les préoccupation de traçabilité existent aussi dans les lignes de produits mais elles présentent des difficultés additionnelles:

– il y a plus de types de produits que pour un développement traditionnel (ex.: modèle de variabilité),

– il y a deux processus (ingénieries du domaine et d'application),

– il y a de nouveaux liens de traçabilité (ex.: entre une application et la famille de produits),

– en résumé, il y a un accroissement de la complexité (ex.: plusieurs applications, ajout de la variabilité).

Les lignes de produits introduisent de nouveaux produits, comme par exemple le modèle de variabilité (qui explicite ce qui peut varier dans l'architecture générale), les points de variation (dans le modèle de variabilité) et leurs différentes variantes. Ceci augmente d'autant le nombre de liens de traçabilité qu'il faut gérer.

Le fait qu'il y ait deux processus est aussi un facteur de difficulté, car l'on doit gérer l'évolution des produits de ces deux processus « en parallèle ». Par exemple, un modèle d'architecture au niveau de la famille de produits réalise des besoins du domaine d'application, un autre modèle d'architecture, cette fois au niveau d'un produit donné, réalise les besoins d'une application spécifique.

Les lignes de produits introduisent aussi de nouveau liens de traçabilité entre les produits d'une application donnée et la ligne de produit (ou la famille) dont elle est dérivé. Ainsi, les besoins d'une application donnée, doivent être reliés aux modèles architecturaux qui les réalisent, mais aussi à la description des besoins génériques dont ils sont dérivés au niveau de la famille de produits.

Tout ceci produit un accroissement de la complexité en terme de quantité (plus de liens à gérer, plus de types de liens) et de qualité (gestion de deux processus « en parallèle »).

3. *Taxonomie de traçabilité*

Pour faciliter la gestion de la traçabilité et combattre sa complexité accrue dans le contexte des lignes de produits, il nous a semblé utiles de catégoriser les liens de traçabilité possibles. De cette façon, il devrait être possible de traiter de manière indépendantes différent type de traçabilité et ainsi diminuer la complexité. Cela peut aussi faciliter le développement d'outils automatisés de gestion de la traçabilité en abordant chaque catégorie séparément. Une première version de la taxonomie a été présentée dans (Shakil Khan, *et al.*, 2008). Elle présente une classification plus fine que les dimensions qui seront présentées ici.

Le génie logiciel reconnaît typiquement deux formes de traçabilité (ex. (Grady, 2006, p.59), (Pfleeger, 2005, p.526)): vertical et horizontal. La traçabilité verticale se fait le long du processus de développement des besoins, aux modèles intermédiaires, au code. La traçabilité horizontale se fait entre produits d'un même niveau d'abstraction (entre les descriptions des besoins, entre modèles, entre composants logiciels) qui sont sémantiquement reliés (une modification de l'un pourrait suggérer un changement similaire de l'autre). Malheureusement, les auteurs ne concordent pas sur la direction des axes. Certains prennent une position inverse

où la traçabilité horizontale suit le processus de développement et la verticale est entre produits d'un même niveau d'abstraction.

Nous conservons ces deux dimensions de traçabilité pour les lignes de produits logiciels. Nous appellerons la traçabilité selon le processus de développement: traçabilité inter, car elle relie des produits à différent niveaux d'abstraction. Nous appellerons la traçabilité entre produits d'un même niveau d'abstraction: traçabilité intra. La traçabilité inter peut se produire entre produits dans l'ingénierie du domaine (famille de produit) ou dans l'ingénierie des produits (applications). De même, la traçabilité intra se produirait entre produits d'un même type d'ingénierie.

Donc, pour représenter les liens entre les produits d'une ingénierie (un besoin d'une application relié au même besoin dans le domaine), nous devons ajouter une troisième dimension, orthogonale aux deux autres, et qui représente la variabilité. La variabilité est une partie fondamentale des lignes de produits et il est naturel que ceci se répercute sur la traçabilité. En dehors des relations entre applications et famille de produits, la dimension de variabilité servira aussi pour représenter les relations entre points de variation (ex. s'ils sont exclusifs l'un de l'autre), entre variantes et point de variation (ex. si une variante impose un point de variabilité), entre variantes, ou finalement entre variantes et produits de plus bas niveau d'abstraction qui les réalisent.

Parce que le projet AMPLE se propose aussi de gérer les versions des produits (gestion de configuration), nous considérons aussi une dimension de temps ou dévolution. Dans cette dimension, un produits est relié à ses versions antérieure et postérieure.

4. Un méta modèle de traçabilité

Pour gérer concrètement les liens de traçabilité, nous avons besoins d'outils qui extraient, stockent et recherchent ces liens. Les membres du projet AMPLE ont définis un meta-modèle présenté en Figure 2. Le méta-modèle définit les classes de type d'artefact (« ArtefactType ») et type de lien (« TraceLinkType ») qui permettent de catégoriser les liens et artefacts. Par exemple, la taxonomie de liens de traçabilité peut être représenté grâce à ces types de liens. Les liens de traçabilité ont plusieurs sources et destinations (des artefacts) ainsi que des propriétés et une possible explication (« rationale »). Il n'y a pas de restriction spéciale sur les types possibles d'artefacts ou de liens se qui permet d'avoir un méta-modèle très neutre et adaptable à une grande variété de situations. Par exemple, on peut accommoder la programmation à aspects ou des processus de développements très différents comme par exemple le développement dirigé par les modèles. Les gestion de configuration peut aussi être contemplée en donnant à chaque version d'un artefact, un « Artefact » différent.

Le méta-modèle est maintenant implémenté dans Eclipse à partir d'un modèle Ecore. Des outils de gestions des liens sont en cours de développement. La Figure 3

montre une vue d'un *plugin* de Eclipse avec les types d'artefacts et de liens à gauche et une fenêtre d'information (à droite) sur les liens, ici calcule de quelques métriques.

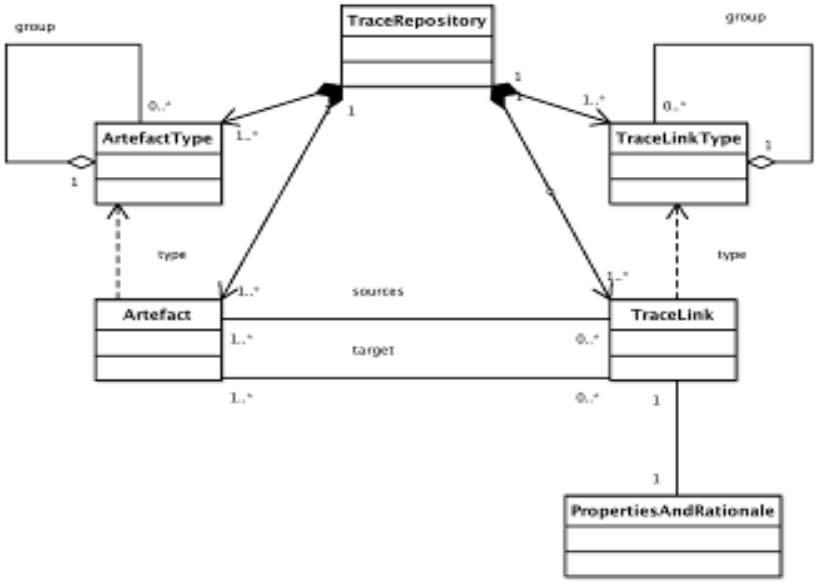

Figure 2. *Le méta-modèle de traçabilité du projet AMPLE*

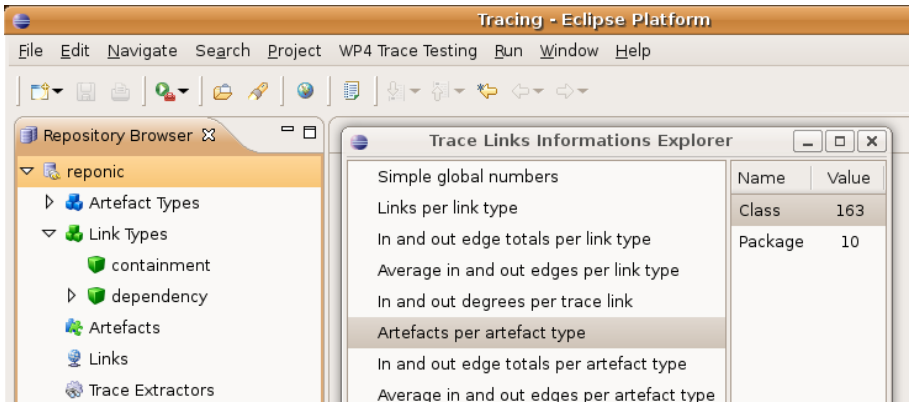

Figure 3. Vue d'écran d'un plugin de Eclipse présentant des informations sur les liens de traçabilité et les artefacts

5. Traçabilité en présence d'incertitude

Un autre problème considéré dans le projet AMPLE est celui de l'incertitude sur les informations disponibles au moment de la prise de certaines décisions durant le développement. Dans l'idéal, lors de la prise d'une décision (de projet, d'architecture, ...), l'information disponible serait complète ce qui permettrait de prendre des décision garantissant la qualité des produits. Bien sûr, ce scénario n'est pas réaliste et en pratique on doit d'attendre à n'avoir que des informations incomplètes ou imparfaites.

Si ces incertitudes sur les informations est un problème pour le développement de logiciel traditionnel, ce l'est encore plus pour les lignes de produits qui servent de base pour produire plusieurs applications, et pour lesquelles des décisions doivent être prises qui influenceront des applications encore non envisagées (ce qui déjà un cas d'incertitude).

5.1. Traitement de l'incertitude

Nous traitons l'incertitude dans le cadre de la traçabilité pour conserver une trace des décisions prises et des informations disponibles au moment ou elles l'ont été. Typiquement, pour chaque décision, plusieurs solutions sont envisagées. Ces alternatives sont évaluées suivant des critères de qualités définis fournissant les justifications pour préférer telle ou telle solution. Mais, comme mentionné plus haut, les informations disponibles pour l'évaluation de l'adéquation de chaque solution au critères choisis est incomplète ou imparfaite. Il devient donc difficile de prendre une décision adéquate dans ces conditions, et il devient important de garder une trace de toutes les alternatives, des informations disponibles et de leur évaluation au moment de la prise de décision. De cette façon, la décision peut-être revue et réévaluée ultérieurement, par exemple quand de nouvelles informations sont disponibles (celles ci pouvant être la découverte de l'échec de la décision qui avait été prise).

Nous avons proposé un modèle qui vise à représenter et documenter les hypothèses formulées durant une prise de décision et les informations sur lesquelles elles se basent. Dans ce modèle, chaque hypothèse doit être évaluée selon une métrique, quel qu'elle soit. La valeur de la métrique est représentée avec un nombre flou (*fuzzy number*) qui permet de représenter l'incertitude sur cette hypothèse (dut au manque d'information). Chaque alternative d'une décision inclue plusieurs de ces hypothèses qui sont combinées entre elles par une formule à définir. Cette combinaison de nombre flous met en œuvre un calcul de probabiliste selon le degré d'incertitude de chaque hypothèse. Les alternatives sont finalement comparées entre elles selon ces valeurs finales. En conservant les valeurs individuelles de chaque hypothèse, on s'autorise à réévalué celles-ci dans le futur et à revoir la classification des alternatives.

Suit un exemple démontrant le mécanisme proposé: Imaginons une décision architecturale qui offre trois alternatives: architecture à objets (OO), architecture de composants (CO) et architecture par aspects (AO). Les trois architectures possibles vont être comparées selon leur coût, leur adaptabilité et leur tolérance aux fautes. Des poids différents peuvent être définis pour chaque critères, par exemple le coût pourrait être considéré comme un critère primordial et recevoir un poids plus important. Les estimations de chaque architecture pour chaque critère (hypothèses) sont données dans la Table 1. La dernière colonne de la table donne la valeur finale de chaque alternative par combinaison des valeurs pour chaque critère.

En cas de révision des estimations (obtention de nouvelles informations), les valeurs dans la table sont revues et l'on peut-être amené à reconsidérée la décision prise.

Table 1. *Cas hypothétique de comparaison de plusieurs alternatives selon différent critères*

	Coût	**Adaptabilité**	**Tolérance aux fautes**	**Valeur Finale**
Architecture OO	5	3	3	3,33
Architecture CO	3	5	5	2,83
Architecture AO	2	2	3	2,83

5.1. Traçabilité de l'incertitude

Notre objectif est de garder la trace des décisions prises en présence d'incertitude. A priori, la méthode de traitement de l'incertitude proposée ci-dessus peut s'adapter à n'importe quelle approche de traçabilité. Nous proposons ici un exemple pour illustrer son fonctionnement (section basée sur (Galvão, *et al.*, 2008)).

Nous commencerons par une présentation du modèle de donnée permettant de stocker les informations relative à une décision. Ce modèle tire certains éléments de (Kunz *et al.,* 1970), (MacLean *et al.*, 1991), (Tekinerdogan, 2006), and (Aksit, 2004). Il est illustré dans la Figure 4. Il y a une décision (*Decision*, en haut à gauche) pour chaque problème (*Problem*) et possible solution (Solution). Pour chaque décision il y a une raison (*Rationale*) associée qui justifie cette décision. Une raison contient les alternatives (*Alternative*) considérées. Une alternative est un solution candidate possible pour résoudre un problème. Les alternatives ont des arguments (*Argument*) en faveur ou contre eux et des suppositions (*Assumption*) sur leurs caractéristiques. Une supposition peut-être quantifiable ou non, quand c'est le cas nous parlons de caractéristique de qualité (*QualityFeature*) à laquelle sont associés des valeurs (*QualityValue*).

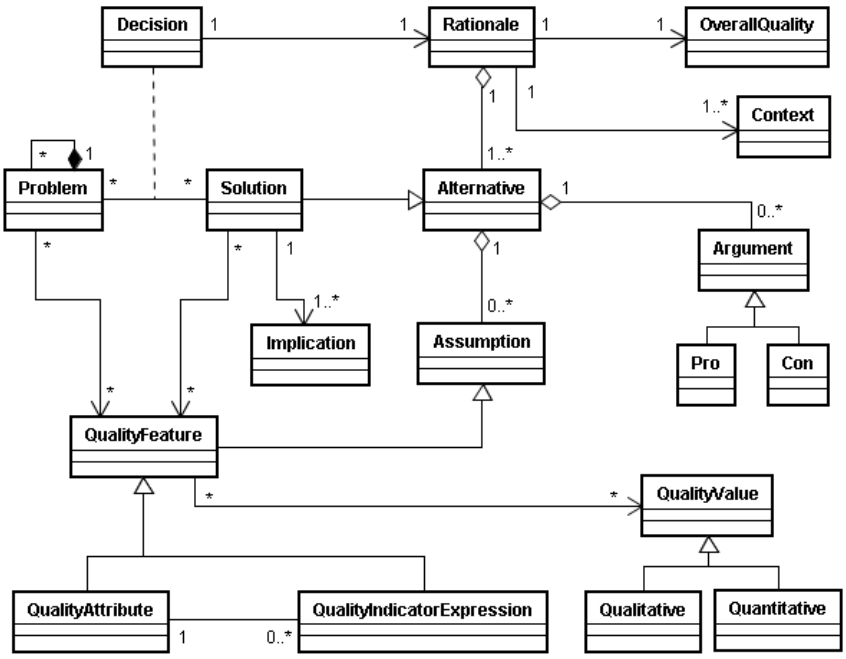

Figure 4. *Modèle de donnée des décisions et de leur raisons*

Garder les raisons d'une décision est déjà de la traçabilité, mais il faut aussi relier (tracer) les décisions à leur conséquences pratiques dans le projet et le code (Ramesh, 2001). Les éléments *Decision*, *Rationale*, *Alternative*, *Solution*, *Problem*, etc. de la Figure 4 sont des artefacts du développement (*Artefact* dans la Figure 3) qui peuvent donc être tracés comme tout autre artefact et avec tout autre artefact. Ainsi toute décision peut être très naturellement tracée jusqu'à ses raisons, problème, solution, etc., mais aussi jusqu'à d'autres artefacts de développement plus traditionnels comme des modèles, composants, programme, etc.

6. Conclusion

La traçabilité est une nécessité importante pour le développement et la maintenance de logiciel. Si ceci est vrai pour le développement traditionnel, ce l'est encore plus pour le développement avec des lignes de produits logiciels où il est fondamental de garder une trace entre les applications et les modèles du domaine créés pour pouvoir propager toute modification des application vers la famille d'application ou à l'inverse de la famille d'application vers ses applications.

Le problème est aussi plus complexe pour les lignes de produits logiciels du fait du plus grand nombre de produits possible, des deux processus de développement (ingénierie du domaine et ingénierie des applications) et de nouveaux types de liens (variabilité).

Nous avons proposé une taxonomie de traçabilité qui compte trois dimensions plus une. Les trois dimensions sont l'intra-traçabilité, entre des produits à un même niveau d'abstraction; l'inter-traçabilité, entre des produits à différent niveaux d'abstraction (selon le processus de développement); et la traçabilité de variabilité, entre les applications et la famille d'applications ou entre les points de variation et variantes. La dimension additionnelle est l'évolution quand l'on souhaite garder trace des différentes versions des produits du processus de développement.

Nous avons aussi présenter un méta-modèle de traçabilité qui nous sert de cadre pour le développement d'outils de gestion de la traçabilité.

Finalement nous avons discuté la gestion de l'incertitude dans le développement de logiciel. Toute décision durant le développement implique une certaine incertitude au niveau des informations disponibles. Il est important de garder trace des raisons qui ont amenées à une décision pour pouvoir revenir sur cette décision quand de nouvelles informations deviennent disponibles. Pour cela nous avons présenté un modèle de gestion de l'incertitude qui nous permet de garder les hypothèses qui ont mené à une décision et le niveau d'incertitude associé à ces raisons.

Références

Galvão I., Noppen J., van den Broek P., Aksit M., Managing Design Decision Rational in the Presence of Uncertainty, *soumis à publication*, 2008.

Geraci A., IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries. Institute of Electrical and Electronics Engineers Inc., 1991.

Gotel O.C.Z., Finkelstein A.C.W., “An analysis of the requirements traceability problem”. In: Proceedings International Conference on Requirements Engineering (RE). pp. 94–101, 1994.

Kunz W., Rittel H., Issues as Elements of Information Systems, Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California, 1970.

Grady J.O., System Requirements Analysis. Academic Press, Inc., Orlando, FL, USA, 2006.

MacLean A., Young R.M., Bellotti V.M.E., Moran T.P., Questions, Options, and Criteria: Elements of Design Space Analysis, *Human Computer Interaction*, vol. 6 , pp. 201–250., 1191

Pfleeger S.L., Software Engineering: Theory and Practice. Prentice-Hall, Inc., 3rd edition, 2005.

Shakil Khan S., Greenwood P., Garcia A., Rashid A., On the interplay of requirements dependencies and architecture evolution: An exploratory study. In Proc. of the 20th Int'l Conf. on Advanced Information Systems Engineering, CAiSE 2008. Springer Verlag, jun. pp. 16-20, 2008.

Tekinerdogan B., Aksit M., Introducing the Concept of Synthesis in the Software Architecture Design Process, *Journal of Integrated Design and Process Science*, 10(1), pp. 45–56, 2006.

Aksit M., The 7 C's for Creating Living Software: A Research Perspective for Quality-Oriented Software Engineering, *Turkish Journal of Electrical Engineering & Computer Sciences*, 12(2), pp. 61–95, 2004.