# Towards an i*-based Architecture Derivation Approach

Diego Dermeval[1], Monique Soares[1], Fernanda Alencar[2], Emanuel Santos[1], João Pimentel[1], Jaelson Castro[1], Márcia Lucena[3], Carla Silva[4], Cleice Souza[1]

[1]Universidade Federal de Pernambuco - UFPE, Centro de Informática, Recife, Brazil
{ddmcm,mcs4,ebs,jhcp,jbc,ctns}@cin.ufpe.br
[2]Universidade Federal de Pernambuco - UFPE, Departamento de Eletrônica e Sistemas, Recife, Brazil,
fernanda.ralencar@ufpe.br
[3]Universidade Federal do Rio Grande do Norte - UFRN, Departamento de Informática e Matemática Aplicada Natal, Brazil,
marciaj@dimap.ufrn.br
[4]Universidade Federal da Paraíba - UFPB, Centro de Ciências Aplicadas e Educação, Rio Tinto, Brazil
carla@dce.ufpb.br

**Abstract**. Goal orientation, in particular the i* (iStar) framework, offers expressive models that support requirements engineering. On the other hand, the understanding of how requirements models are related to architectural design is still somewhat limited. In the past years, we have been investigating how to derive architectural models from i* (iStar) models, focusing on modularity. As a result we proposed a Strategy for Transition between Requirements and Architectural Models – STREAM. In this paper, we summarize the current state-of-the-art of the STREAM approach, point out its challenging aspects and describe current ongoing research. Our challenge is to support a broader set of architectural decisions as well as to provide means for partially automating the models transformations.

**Keywords:** iStar, Requirements Engineering, Architectural design, Architecture Documentation, Architectural Decisions, Automation, Model Transformations

## 1 Introduction

Despite Requirements Engineering and Architectural Design being strongly related activities, there is a lack of techniques and methods handling the integration of these activities. Therefore, one of the major research challenges in software engineering is to provide systematic methods for designing software architecture from requirements models [2] [3]. The STREAM (Strategy for Transition between Requirements and Architectural Models) process [4] [14] presents a model-driven approach for generating initial architectures - in Acme [7] - from i* requirements models [13]. The STREAM approach consists of the following steps: (i) *Prepare Requirements Models*, (ii) *Generate Architectural Solutions*, (iii) *Choose an architectural solution*, and (iv) *Derive Architecture*. Horizontal and vertical model-transformation rules were proposed in order to perform the steps (i) and (ii), respectively. Non-Functional Requirements are used in the step (iii) to guide the selection of alternatives in the

architecture. Lastly, in the step (iv) the architecture is refined by using architectural refinement patterns.

Based on the generic STREAM process, some others extensions were proposed: STREAM-Adaptive [11] and F-STREAM [5]. The STREAM-Adaptive approach supports the generation of architectures for self-adaptive systems. This is achieved by enriching the i* models with information required to perform the reasoning related to adaptation, which is performed by pre-defined components. The F-STREAM [5] (Flexible STREAM) uses Software Product Lines principles aiming to make it easier to integrate the STREAM approach with other approaches that are able to handle some specific NFRs.

However, there are still some limitations. For example, only one of the possible architectural views [1] is supported. In addition, no support is given to document the different types of architectural decisions [9]. Finally, the model transformations are not yet automated. Hence, in this paper, we show how we intend to improve the family of STREAM approaches in order to face the last two shortcomings: supporting and documenting a broader set of architectural decisions and automating the model transformations required in the process.

This paper is organized as follows. Section 2 presents the goals of the research. Section 3 describes our proposals towards these goals. Section 4 presents the conclusions while Section 5 points out ongoing and future research.

## 2 Objectives of the Research

The general goal of this research is to enhance the STREAM approach, allowing it to be more complete and viable for industrial use. Therefore, we propose two specific objectives. Firstly, we derive an architectural specification that encompasses the documentation of a broader set of architectural decisions. Secondly, we intend to provide tool support to automate the transformations presented in the STREAM approach and its extensions. Thus, we aim to facilitate and promote the use of those approaches. As a side effect, we contribute to the improvement of the modularity and understandability of i* models.

## 3 Scientific Contributions

This section presents the proposed approaches to satisfy the research goals. Section 3.1 describes how we intend to include architectural decisions in STREAM, while Section 3.2 presents how we plan to automate its model transformations.

### 3.1 Architectural Decisions in the STREAM Process

Based on the classification scheme of architectural decisions proposed by [9], we noticed that the STREAM process only allows the decision-making of a subset of architectural decisions types. In this way, we are extending the STREAM approach in order to support two specific kinds of architectural decisions: the existential and technology decisions. So, to systematize the specification of architectural decisions in the extended STREAM process, we combined the step (iii) and (iv) into a single

activity named *Refine Model with Architectural Decisions* that encapsulates the design choices of the former steps in the classification scheme (existential and technological decisions) that we are using. Moreover, we renamed the steps (i) and (ii) to, respectively, *Requirements Refactoring* and *Generate Architectural Model* (Fig. 1).


**Fig. 1 Extended STREAM Process**

The aim of this new *Refine Model with Architectural Decisions* activity is to sharpen up the generated architectural model by considering existential and technology decisions. Moreover, through the documentation of these decisions using some documentation template, it is possible to capture the context, rationale and other relevant information about the decisions. A set of documented decisions are the output of each decision-making activity. In this paper, we do not have sufficient space to describe how we plan to record the architectural decisions, see [6] and [9].

Fig 2 illustrates the sub-process that presents the architecture refinement with decisions. In the *Make Existence Decisions* activity, the architect defines elements or artifacts that are required for the system's design or implementation. This kind of decision includes structural as well as behavioral decisions. For example, structural decisions lead to the creation of subsystems, layers, partitions, components, etc. Behavioral decisions are usually related to how the elements interact together to provide functionality or to satisfy a non-functional requirement [9]. For instance, the choice of a specific architecture pattern can be seen as an existence decision, so it is specified in this activity of the process.

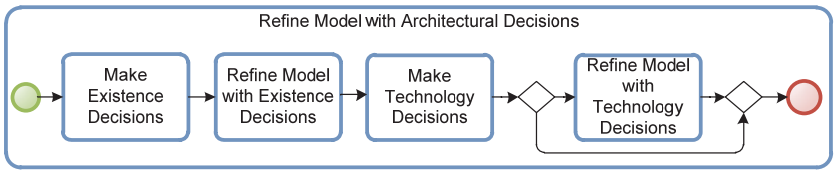


**Fig. 2 Refine Model with Architectural Decisions Sub-process**

The aim of the Refine Model with Existence Decisions activity is to refine the architectural models to reflect the existence decisions made during the earlier activity. Thus, the outputs of this activity include a refined ACME architectural model together with the list of existence decisions made.

The *executive decisions* are the decisions that do not relate directly to the design elements or their qualities, but are driven more by the business environment (financial), the development process (methodology), the people (education and training), the organization, and to a large extent the choices of the technologies and tools [9]. There are different kinds of executive decisions, but at this time we will focus only on technology decisions. So, the *Make Technology Decisions* activity involves decisions that should be part of an architectural specification, mainly, to guide the implementation of the architecture. Examples of technology decisions are the choice of a programming language and the choice of a specific framework.

There is a need to assess if the selected technology architectural decision affects or impacts the ACME architectural model. If this is the case, these decisions are considered in the *Refine Model with Technology Decisions* activity to further refine the ACME architectural model. For instance, selecting a specific API to be integrated with the architecture. Otherwise, if the decision does not affect the architectural model, the process is concluded. For example, the choice of a programming language.

In the next subsection we examine another challenge: the need to provide some degree of automation (tool support) for the approaches.

## 3.2 Automating Model Transformations

Some activities of the family of STREAM approaches can be time consuming. Hence, we should examine if some kind of tool support could be provided, at least to partially automate the processes. The (i) Prepare Requirement Models and (ii) Generate Architectural Solutions steps of STREAM are amenable to some degree of automation, since they rely on model transformations. The first activity relies on horizontal rules to refactor the i* requirement models prior to the architectural model generation. The second activity applies vertical rules to derive architectural models from the refactored i* models.

These transformation rules can be precisely defined using the QVT transformation language (Query/View /Transformation) [12], in conjunction with OCL (Object Constraint Language) [10] to represent the constraints. The transformation process requires the definition of transformation rules and metamodels for the source and target languages. The horizontal rules that aims to refactor the i* models have the i* language both as source and target language. On the other hand, recall that the vertical rules are used to generate architectural models (in ACME) from modularized requirements models (in i*). Hence, our vertical rules have i* as the source language and ACME as target language. Once defined and specified using QVT and OCL, the transformation rules could be incorporated in a tool, such as the iStarTool [8].

Note that the iStarTool already has internal representation of the i* metamodel and could be extended to allow the implementation of the new transformation rules. In doing so, the *Prepare Requirement Models* activity could become semi-automatic. The user would still need to select the candidate sub-set of elements to be factored out. After this selection, all the other steps of the activity could be automated. As a result, the refactored i* model could now be obtained with the press of a button.

The *Generate Architectural Solutions* activity generates candidate Acme models from the modularized i* models. The alternative solutions are derived from the inherent variability of i* models (e.g., due to the Means-Ends relationships). The choice of the candidate solution can be influenced on softgoal or quality attributes present in requirements models. Hence, we envisage including in the iStarTool the ability to generate all possible set of candidate architectures. Moreover, the tool could indicate the degree of satisfaction of a given set of softgoals for each architecture. Furthermore, the generated Acme models are used in subsequent steps (iii) and (iv) of the STREAM Approach. It remains to be studied how these steps could be partially automated.

By automating the model transformations, several experiments can be performed to evaluate different architectural models without additional costs.

## 4 Discussion

In this paper, we have proposed two approaches aiming to improve the systematic process that generates architectural models from i* models. We have presented an approach to include support for recording architectural decisions in STREAM. Furthermore, we have indicated how the horizontal and vertical model transformations presented in the process could benefit from automation and tool support.

The first approach improves the family of the STREAM approaches, by allowing the rationale of the decisions made to be recorded. With this extension, it is possible to specify a more complete architecture by defining a broader set of architectural choices - for example, technology decisions. Moreover, by documenting the architectural decisions, the information that underlies the context of a decision can be recorded. However, such extra information may overload the refinement step of STREAM with documentation activities. Nonetheless, we believe that the benefits of documenting an architectural decision [6] far compensate the extra effort required for recording the rationale. We also need to investigate if we can anticipate specific kinds of decision-making that are common to these in earlier steps of the process.

The second improvement proposed in this paper minimizes the effort of applying the model transformation rules manually. Besides, it eliminates the possibility of making mistakes when manually applying these rules. Since the transformation process could be automatically supported, another positive aspect of this improvement is the increase of productivity, as it enables a simplification of the process and reduces the amount of manual activities.

## 5 Ongoing and Future Work

We offer a family of a systematic method that derives (with semi-automatic support) a candidate architectural design from i* models. With this in mind, we can describe specific ongoing and future work for each approach presented in this work.

On one hand, we are evolving the approach to include architectural decisions. We are defining how we will document the architectural choice. Our first attempt is to use a template as the proposed by Garlan et al. [6]. Hence, we need to evaluate how the i* models can guide or aid the documentation of the decisions. We are also investigating where does design decisions take place in STREAM. As future works, we will specify an extended STREAM approach that results in an architectural design that encompasses both the architectural decisions and the representation views. Furthermore, we need to further validate the approach with several case studies. We also intend to integrate the new process with a tool to manage the artifacts produced in the architectural design step.

On the other hand, we are extending the iStarTool [8] to support the horizontal mapping rules which modularize the i* models. The rules are specified in QVT and OCL. As future work, we plan to develop an iStarTool API to incorporate the vertical mapping rules, which generates the initial model Acme from modularized i* models.

Last but not least, experiments are required to validate the family of STREAM approaches as well as the new iStarTool functionalities.

## References


1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice (2nd Edition) Addison-Wesley Professional, 2003.
2. Berry, D. M., Kazman, R., Wieringa, R.: Second international workshop on from software requirements to architectures (straw'03). SIGSOFT Softw. Eng. Notes 29, 1–5, 2004.
3. Castro, J., Kramer, J.: From software requirements to architectures (STRAW'01). SIGSOFT Software Eng. Notes 26, 49–51, 2001.
4. Castro, J., Lucena, M., Silva, C., Alencar, F., Santos, E., and Pimentel, J.: Changing Attitudes Towards the Generation of Architectural Models. In: Journal of Systems and Software, 2011 (Accepted for Publication).
5. Castro, J., Pimentel, J., Lucena, M., Santos, E., Dermeval, D.: F-STREAM: A Flexible Process for Deriving Architectures from Requirements Models In: 9th International Workshop on System/Software Architectures (IWSSA'11), 2011 (Accepted for publication).
6. Garlan, D., Bachmann, F., Ivers, J., Stafford, J., Bass, L., Clements, P., Merson, P.: Documenting Software Architectures: Views and Beyond, 2nd ed. Addison-Wesley Professional, 2010.
7. Garlan, D., Monroe, R., Wile, D.: Acme: An Architecture Description Interchange Language. In: Proc.CASCON'97, 1997. Toronto, Canada.
8. IStarTool Project: A Model Driven Tool for Modeling i* models. Available at http://portal.cin.ufpe.br/ler/Projects/IstarTool.aspx, June (2011)
9. Kruchten, P., Lago, P., van Vliet, H.: Building up and reasoning about architectural knowledge. In QoSA, pp. 43–58, 2006.
10. OCL 2.0. Available in < http://www.omg.org/spec/OCL/>. Last access in 2011, June.
11. Pimentel, J., Lucena, M., Castro, J., Silva, C., Santos, E., Alencar, F.: Deriving Software Architectural Models from Requirements Models for Adaptive Systems: The STREAM-A approach. In: Requirements Engineering Journal, 2011 (Accepted for Publication).
12. QVT 1.0 - Query View Transformation. Available in < http://www.omg.org/spec/QVT/1.0/ >. Last access in 2011, June.
13. Yu, E.: Modeling Strategic Relationships for Process Reengineering. Ph.D. thesis. Department of Computer Science, University of Toronto, Canada, 1995.
14. Lucena, M.: STREAM: A Systematic Process to Derive Architectural Models from Requirements Models. Ph.D. Thesis, CIn, Federal University of Pernambuco, Recife, 2010.