StarGro: Building i* Metrics for Agile Methodologies

Colomer, Daniel[1] and Franch, Xavier[2]

[1] Collins GmbH, Hamburg, Germany
dncolomer32@gmail.com
[2] Universitat Politècnica de Catalunya (UPC) c/Jordi Girona, 1-3, E-08034 Barcelona, Spain
franch@essi.upc.edu

Abstract. Requirements management is one of the cornerstone activities in software development. Agile methodologies use dedicated methods, techniques and artifacts in order to implement this activity. Remarkably, Backlog Grooming is the activity of managing and welcoming changing requirements in SCRUM. However, current industrial practices in agile development still tend to render this process in the shape of a list of statements, features and bug fixes that often leads to a blurred view of the goals of the project, the underestimation of client's needs and the decrease of the ability to respond to changes. In this paper we outline an approach that uses goal and agent oriented modelling techniques in order to fill in this "intentional" gap that current industrial approaches lack.

1 Introduction

Agile methodologies are nowadays adopted by most of the software development industrial sectors. These methodologies usually define an artifact and/or methodology that helps organisations in managing their project requirements. Nevertheless, industrial practices tend to render this process in the shape of a list of statements, features and bug fixes that often leads to a blurred view of the project goals, the underestimation of the client's needs and might severely affect the capability of a team to react to changes.

The root of these problems can often be traced down to a misinterpretation of the Agile Principle of "Individuals and Interactions over processes and tools". Based on that principle, organizations tend to focus on development and reduce the activity of requirements management to the point in which the goals of the project get blurred, lost and only visible by the Product Owner but never reaching the development team.

In this paper we present an approach that uses i* as a way to capture the intentionality that is lost during the activity of requirements management. Our current work has the following objectives:

– To define a set of guidelines that will compose a base approach for building and using metrics in requirements engineering activities focused on agile methodologies.

– To define a set of guidelines that will compose a base approach for implementing metric repositories.
– To implement a public repository of i* metrics to be used as an example and base for metrics driven requirements engineering in agile methodologies.
– To implement an industry ready, open source and metrics-aware i* modelling tool.

The paper is structured as follows: First, we take a detailed look at the previously stated problem using a real industry case extracted from the German web development market executed between 2012 and 2014. Then, we follow by presenting StarGro, a set of guidelines to help implement metric repositories and we give some insights on a sample set of metrics that is being build using this approach. Finally, we discuss the first observations after using a prototype implementation of the suite and we close the paper by presenting a brief discussion and future work.

2 Case Study: Press Distribution System

A press distribution organization based in Hamburg, Germany, started a project with the goal of moving its current management system to the web. The project was defined as a complete reimplementation of the system in the shape of a web platform.

Initial Settings The initial set of requirements was given in a single 88 pages long document describing the functionality of the system divided in 9 different modules. This documentation was complemented with a set of screenshots of the original application. The project was planned to run 6 months and the initial development team configuration was of 1 lead developer, 4 senior developers, 1 junior developer and 1 front-end designer.

Project Results & Retrospectives The Project was never fully finished by the original team and it was transferred to another organization. The deadline was delayed more than 2 years.

During the development new resources were introduced such as a running copy of the original system which gave the developers some insights on how the original solution looked like. Customer collaboration was active in a daily basis throughout the whole project timespan.

Lessons Learned The following lessons were derived by the original team after assessing the retrospectives of the failed project:

– The need of tracking stakeholders and their goals and making this knowledge project-wide available.
– The need of systematically refining and prioritizing requirements.
– The need of tracking the project evolution and progress in order to predict, welcome and minimize changes better.

The Need of Metrics Metrics as a progress tracking and prediction tool were highlighted as potential solution to the need of welcoming changes and tracking the evolution of requirements.

In order to cope with all these problems, we explore in this paper the use of i*. This adoption can lead to keeping track and making the domain knowledge and intentionality project-wide available. Once we have a model that represents our domain from an intentional point of view and a set of metrics to evaluate it we can then evaluate different prioritization approaches.

In order to combine i* and metrics we propose the usage of iMDF [1], a method for defining metrics in i* which is mainly focused on the analysis of the target domain and the metrics themselves. The iMDF Framework can be summarized in 4 steps:

1. **Domain Analysis**: The goal of this step is to gain understanding about the domain whilst establishing the mapping from concepts in that domain onto the i* framework.
2. **Domain Metrics Analysis**: This step aims at analysing the departing suite of domain metrics before its formalization is tackled.
3. **i* Metrics Formulation**: This step makes operative the metrics in terms of i* constructs.
4. **iMDF Update**: This step includes eventually updating statistics, the language of patterns and the metrics catalogue.

3 StarGro: Building Metrics for Agile Methodologies

The following lines describe StarGro. The aim of Stargro is to help individuals and organizations improve requirements management activities by means of using metrics designed and specially tailored for Agile Methodologies on top of i* models. This approach can be applied both to individual organizations as well as to a particular domain (e.g., web applications).

The approach can be summarized in the following guidelines:

1. The users of the approach need to apply the 4 steps defined by iMDF on their target domains during the set up of a project in order to identify and clarify the domain of action and possible needed metrics.
2. Metrics will be organized and available in repositories for future reuse. Teams can reuse existing metric, add new ones or tailor existing metrics.
3. For applicability purposes, repositories should be based on a single i* meta-model and all the metrics belonging to that repository must be compatible with that meta-model.
4. A team can use more than one repository at a time as long as they are all defined on compatible meta-models.
5. The technology and means used to implement or represent metrics are not restricted by this approach.
6. Metrics will be executed as the i* models used in the project change. The results of the execution can be observed and interpreted during backlog

grooming sessions or at any moment in order to track the evolution of the project, get better insights on how to proceed and being able to react faster to change.
7. Feedback after using certain metrics is relevant specially for the 4th step of iMDF.
8. There are no restrictions on how to use metrics. Suggestions on the usage of a metric should always be available to the end user. Metrics can be used for example as a prioritization tool, as an elicitation help, etc.

4 Applying StarGro

Following the guidelines presented in the previous section we started implementing a public repository of i* metrics to be used as an example and base for metrics driven requirements engineering in agile methodologies.

The formal specification and an XQuery [3] implementation of the suite are work in progress. The source code of the current state is available online [4] packed as an Eclipse Project ready to be executed [5]. The Project contains iStarML [2] files as well as openOME [6] (.ood) resources to use as examples.

The full specification of the metrics and the details of applying StarGro are available online. [7]

4.1 The Metrics

The following is a list of the metrics included in the suite accompanied with a brief description.

- **Change Proneness**: This metric is defined as the number of both direct and indirect outgoing dependencies an element has. This metric is an adaptation of the Behavioural Dependency Measurement (BDM) presented in [3]. By definition, two elements connected by a dependency -directly or indirectly- in i* are intentionally dependent. Changes in an intentional element may occur by modifications to the element itself or by changes propagated from other related elements. Changes applied directly to an intentional element are difficult to predict in i* models, therefore we will consider for our prototype specification the changes occurred by propagation. We will base the measurement on intentional dependencies.
- **Intentional Bottleneck**: This metric is defined as the amount of directly incoming dependencies of an element. This metric uses the amount of incoming dependencies both at an actor and intentional element level. We assume the more incoming dependencies an element has the more critical it is for the

[3] http://www.w3schools.com/XQuery/
[4] http://www.dncolomer.net/resources/distar.zip
[5] http://www.dncolomer.net/resources/distar-tutorial.pdf
[6] https://se.cs.toronto.edu/trac/ome/
[7] http://www.dncolomer.net/resources/applying-stargro.pdf

success of a project. The rational behind this assumption is that the more incoming dependencies an element has the more propagations a change is likely to cause.
- **Epicness**: This metric is defined as the proportion of unrefinable elements an element is decomposed into. This metric explores SR graphs' leaves in order to help determining whether an intentional element is refined enough. We assume that the bigger proportion of tasks and resources the leafs of an intentional element are decomposed into, the more refined this element is. The rational behind this assumption is the fact that only goals and softgoals represent an intentional desire. The exploration of leaves is done ignoring the SR-Link type.

4.2 Observations

We present in this section the first observations of experimenting with the prototype implementation of the suite. We applied the metrics to a real ongoing project called "Outfit Creator". The Outfit Creator is a Web Application that enables users to graphically create fashion outfits using images and resources from an e-commerce platform.

- **Change proneness**: We observed that the current implementation predicts change more successfully when requirements are more refined. This is due to the fact that the we currently use the number of i* dependencies to determine how strong the intentional relationship between two Actors is. The more refined a requirement is, the more dependencies we are able to identify. In order to achieve a more accurate change prediction we are currently studying the possibility to include the exploration of SR-Links and other i* constructs as well as giving different weights to direct and indirect dependencies. Furthermore we are also studying how can we explore forward when a dependee is an actor and not an intentional element. In the current prototype version the actor is considered a leaf of the explored graph.
- **Intentional Bottleneck**: In this case we observed a similar situation than in "change proneness". We observed that the current implementation performs better in stages where requirements are more refined. Nevertheless this only affects us in cases where we need to compare or priroritize bottlenecks and therefore we need better insights on the "strength" of these points. In order to improve the accuracy of this metric we are studying the possibility to include the exploration and study of both depender and dependee of each of the incoming dependencies.
- **Epicness**: This metric has proven to perform the worst. The main problem is the fact that simply studying the proportion of tasks and resources in the last SR-Link level is not enough to determine with precision whether a requirement is refined enough or not. In order to achieve a more accurate measurement we are studying the possibility to include the SR-Link type as a relevant factor. We are studying the possibility to consider not only the proportion of tasks and resources but also patterns of proportion evolution from level to level and the number of leaf elements as well as its evolution.

5 Discussion & Further Work

In this paper we sketched StarGro: a set of guidelines to help individuals and organisations build metrics for Agile Methodologies. We started building a suite of metrics to be used as a base and example.

We did not specify whether the approach presented in this paper should be applied at an industry level, therefore enabling a global repository of metrics, or at an organization level, therefore supporting the existence of many private repositories that belong exclusively to concrete groups, organizations or teams.

Nevertheless we encourage a mixed solution. The existence of repositories both public and private favour all levels. That is why we define later in this document a reduced public Dependency Based Prediction Metrics Suite that can be used as a base for any private or public entity and will be further extended in the future.

A natural next step is for us to study and define a repository containing the suite of metrics. We are also planning to add new guidelines to StarGro regarding the generic implementation of i* metrics repositories.

As a future work we are planning to further refine the suite of metrics and to publish a public and open source version. Parallel to that we are planning to add more metrics to the suite and to develop a web application that can handle i* modeling together with metric definition and execution.

We are currently exploring the possible usage of XQuery, iStarML and Metric Repositories in order to define model and dependency aware filters for i* models. The aim of such filters targets the automatic generation of i* diagram views that contain elements satisfying a provided filter criteria. The potential usage of filters is tightly related to the need of requirements management features in i* modeling tools. If we aim at enabling i* to be used as a requirements engineering framework in the industry we must provide a way for developers, business, and other potential industrial users to search and filter requirements in an i* diagram or set of diagrams for that is a really common use case in such tools

References

1. Xavier Franch: A Method for the Definition of Metrics over i* Models.CAiSE 2009: 201-215
2. Carlos Cares, Xavier Franch, Anna Perini, Angelo Susi: iStarML: An XML-based Model Interchange Format for i*. iStar 2008: 13-16
3. Ah-Rim Han, Sang-Uk Jeon, Doo-Hwan Bae, and Jang-Eui Hong: Behavioral Dependency Measurement for Change-proneness Prediction in UML 2.0 Design Models. COMPSAC 2008: 76-83