Evaluating Failure Propagation in the Application Landscape of a Large Bank

Josef Lankes[1], Florian Matthes[1], Tarmo Ploom

[1] Technische Universität München

1 Motivation and Initial Situation

Today, enterprises operate a large number of applications providing critical support to the business. These applications form, when taken together, the application landscape, which can be seen as an important asset, providing essential support to business processes, but sometimes also acting as a limiting factor.

Important quality attributes depend not only on architecture and implementation of specific applications. The support an application landscape can deliver to business also depends on how the applications are integrated in the landscape.

This article focuses on failure propagation in an application landscape, which affects the availability at which the applications offer their specific services.

We applied metrics we introduced in [LS1] for evaluating failure propagation aspects in an application landscape on two proposals stakeholders from a large bank created to limit failure propagation. The evaluation was targeted at the subset of the landscape application supporting private banking, specifically the one located on the mainframe. This subset of the application landscape consists of 255 applications, organized into 75 subdomains, which are themselves organized into 18 domains. Together, the applications amount to about 12 millions lines of PL/1 code.

1.1 Proposals for Limiting Failure Propagation

Figure 1 shows, how these proposals intend to limit failure propagation.

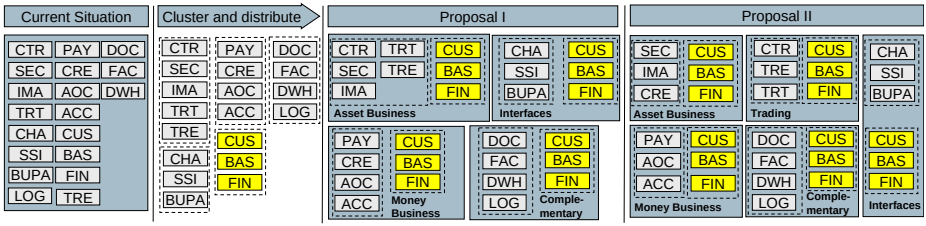

Fig. 1. Domains and their distribution in the proposals limiting failure propagation

Both proposals rely on making changes to how domains (small rectangles in Figure 1) are deployed and communicate. Thereby, the domains are organized

into so-called domain clusters (rectangles with slashhed lines) along functional concerns. In the as-is landscape, one platform (large rectangle with full line) hosts all domains. Contrastingly, the proposals distribute the domain clusters to different platforms, as illustrated on the right side of Figure 1: Proposal I introduces the platforms Money Business, Asset Business, Interfaces, which offers functionality to customers and suppliers, and Complementary, which contains non-banking functionality. Each platform hosts a specific domain cluster and a replication of the fundamentals-cluster, which provides basic data and services.

The platforms are independent, therefore, only asynchronous communication is allowed between them. The data used by the fundamentals-domains is replicated between their different deployments. Only one deployment of a fundamentals-domain is allowed to change its data, the other deployments are restricted to read-only access.

The two proposals differ in the number of platforms they create. Proposal II uses an additional platform, into which it puts an replication of the fundamentals-domains, and the domains CTR, TRT, and TRE.

2 Metrics for Assessing Failure Propagation on Application Landscapes

In order to evaluate the two proposals in respect to their ability to limit failure propagation in the application landscape, we used the metrics introduced in [LS1]. The metrics are calculated on detailed information about the targeted subset of the application landscape, which was structured as shown by Figure 2.

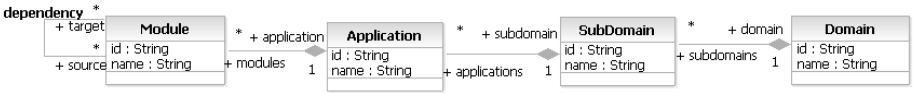

Fig. 2. Information model of the data available about the application landscape

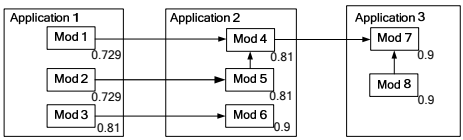

Fig. 3. Calculating failureProbability on information according to Figure 2

Figure 3 exemplifies, how the metrics were calculated on data structured as shown in Figure 2. This calculation procedure assumes, that a module fails, if

one of the modules it depends on, also transitively, is no longer able to render its services, as the respective Application has failed. For calculating the failureProbability of a module m, the algorithm starts at m, and derives the set of modules which are, also transitively, called by m. Then, it derives the set of Applications, in which these modules are located. These are the Applications which need to be operational for the module under consideration being able to render its services. n is the size of this set. Assuming, that Applications fail independently with availability[1] A , the failureProbability of m is $1 - A^n$.

3 Evaluating Proposals for Limiting Failure Propagation

failureProbability, as described in Section 2, has been calculated for all modules. The probability of the complementary events 1-failureProbability(m), interpreted as an availability of the respective module, was averaged over the respective Domains. These values (called averageServiceAvailability) were calculated for both the as-is application landscape, Proposal I, and Proposal II.

Figure 4 shows these results, visualizing each one of the three scenarios as a line, with the domains on the x-axis, and the y-coordinate of the line showing the averageServiceAvailability of the respective domain. If a proposal has multiple deployments of a domain, the respective graph visualizes an average value.

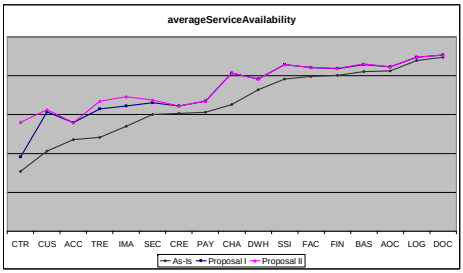

Fig. 4. Comparing the as-is landscape, Proposal I and Proposal II

In interpreting these results, the stakeholders went into two directions. On the one hand, the evaluation results gave them new insights into their proposals, showing especially, that the domains benefit largely differently from the proposals. Especially two domains, SEC and CRE, did not improve as expected.

On the other hand, the stakeholders discussed the assumptions underlying the evaluation. One point related to the assumption that dependencies crossing platform borders can be replaced by messaging, and then be considered having only minor impact on availability. This assumption can be disputed. If a dependency reads data, the need for this data does not disappear when messaging is

[1] Via the shared availability, the approach focuses on the landscape, and not on characteristics of specific Applications. However, it could be extended into this direction.

introduced. If the data is not returned within a specified time, this still constitutes a failure. However, as the dependency information does not contain the directions of the data flows, above evaluations are still used as an (possibly optimistic) approximation. This confirmed to the stakeholders, that projects need improved data about the application landscape. They limited this statement not only to above proposal comparison, but directed it at (IT-) projects in general.

Moreover, above evaluation did not consider an effect which can be exepected from the replication of basic functionality: Reduced likelihood of large failure events, which affect a high share of modules. In order to assess this effect, failure distributions were estimated[2]. These distributions indicate the probability of failures involving differently large shares of the modules in the landscape. They are shown for the as-is landscape in Figure 5, and for Proposal I in Figure 6, and clearly indicate, that Proposal I reduces the likelihood of large failures.

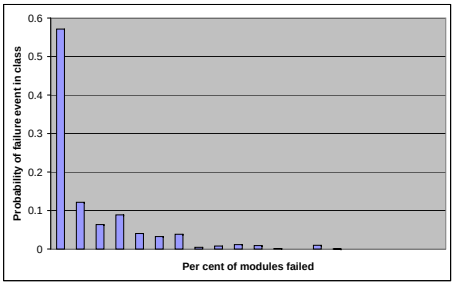

Fig. 5. As-Is

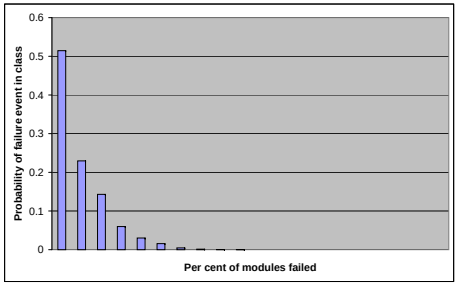

Fig. 6. Proposal 1

4 Resume and Outlook

Besides providing the stakeholders with information about their proposals, the metrics analyses helped them to refine their questions about the proposals. For example, reduced likelihood of large failures was not directly discussed before the metrics analyses. This resulted in a more systematic discussion of approaches for limiting failure propagation, based on explicit assumptions and information.

Thus, above described case showed, that metrics at the application landscape level are a suitable aid in the evolution of the respective architectures. Currently, we are researching into approaches for assessing quality attributes related to throughputs, latencies, but also changeability at application landscape level.

References

[LS1] Lankes, J., Schweda, C.: Using Metrics to Evaluate Failure Propagation in Application Landscapes. Multikonferenz Wirtschaftsinformatik, 1827–1838 (2008)

[2] via Monte Carlo simulations, exact calculation was too computation intensive