

Assessing a Multi-Site Development Organization for Architectural Compliance

Viktor Clerc, Patricia Lago, Hans van Vliet
Department of Computer Science
Vrije Universiteit, Amsterdam, the Netherlands
{viktor, patricia, hans}@cs.vu.nl

Abstract

Multi-site development organizations require coordination and communication efforts between different sites to ensure successful distributed development. These efforts need to be guided by a set of principles and statements on the software architecture that must be complied with throughout the organization: architectural rules. It is of paramount importance that multi-site development organizations incorporate measures in the architecting process to secure compliance with these rules throughout the organization. We describe a method to assess the degree to which compliance measures are secured in multi-site development organizations. We share our experience in applying this method in a large development organization in the consumer electronics domain.

1. Introduction

An increasing number of organizations applies offshore software development in order to deliver solutions to its customers. This kind of multi-site software development faces additional problems compared to same-site development, such as differences in culture and spreaded software development activities. This spread requires additional communication and coordination efforts. These efforts aim to guarantee successful integration of subsystems into a working software product that meets the requirements.

Aforementioned communication and coordination efforts can be addressed by issuing software architectural rules. *Architectural rules* are the principles and statements on the software architecture that must hold at all times, and thus must be complied with. Architectural rules are not limited to software artifacts only, but include e.g. organizational processes that manipulate or create the artifacts as well. For example, a rule on branching might read: *a subsystem is the unit of branching, i.e. a subsystem branch is a branch of all assets in the subsystem.*

When an organization has a set of architectural rules

installed, these rules act as (additional) non-functional requirements to the software architecture. Lack of compliance with architectural rules often results in architectural mismatch [5], causing problems when reusing or integrating certain software elements from different sites. Examples of these problems include an error-prone construction process or excessive code size.

In order to let an organization comply with architectural rules, the rules need to be enforced. Besides enforcing the architectural rules themselves, achieving compliance requires guidelines for how to disseminate and secure the rules within the different development sites.

Differences in culture and a spread across different locations hamper compliance with architectural rules at multi-site organizations. This puts pressure on the guidelines targeted at disseminating and securing the rules. In order to bring the use and effectiveness of these guidelines to the surface, an assessment of the way these multi-site organizations secure their architectural rules may prove valuable.

This paper describes a method for assessing multi-site development organizations to determine deficiencies in securing architectural compliance. We report a case study in which we applied the method in a large multi-site software development organization. The case study revealed that, while initially problems were expected with regard to the *structure* of the architectural rules, the real problems pertained to the *process* by which the architectural rules are captured, disseminated, and secured within the organization.

This paper is structured as follows. Section 2 describes related work on architectural rules and methods to evaluate software architectures. Next, Section 3 introduces the organization at which the case study was performed: a large multi-site development organization in the consumer electronics domain. Section 4 describes the method used and the experiences in applying the method in the organization. Next, Section 5 describes the results of the assessment and the subsequent steps the organization took. Finally, Section 6 lists our conclusions and describes future work.

2. Related work

Recent work regards software architecture as a set of architectural design decisions [7]. Capturing architectural design decisions enables more architectural knowledge to be represented explicitly in the software architecture and thus prevents vaporization of architectural knowledge. Architectural knowledge consists of the set of architectural design decisions and the resulting design [9]. A specific subset of the architectural knowledge is formed by architectural rules. The need for architectural rules to guide and constrain the amount of artistry in software architecture was initially identified in [1].

Architectural rules are those architectural design decisions that need to be complied with throughout the organization. Architectural rules can be defined using a decision view on software architecture as described in [12].

Over the past few years, methods for assessing and evaluating software architectures [3, 10] have been developed. The steps in the method we followed fit the general guidelines on how to conduct architectural reviews as described in the SARA report [10].

Herbsleb et al. [6] report on a large case study in which multi-site development was compared to same-site development. Their contribution shows that multi-site development lowers the degree of informal communication and consequently the level of insight a site has into the software developed at another site. Multi-site development lacks a certain amount of trust provided by informal communication and needs a more formal communication process. Furthermore, according to [11], coordination of multi-site development can be organized according to standardized processes and written specifications. Our work regards software architecture as a coordination mechanism since the architecture divides a system into components that can be developed relatively autonomously. The processes and specifications that guide this development form what we regard as *architectural rules*. Compliance with these rules is of paramount importance for multi-site development organizations. However, achieving compliance requires additional effort and guidelines. We report on our experience in assessing an multi-site development organization for these efforts and guidelines.

3. Case study description

We performed a case study at a large software development organization that has multiple sites where software is being developed for a consumer electronics product. Software development within this organization is structured in projects. Each project delivers a new type of the product. As such, each project develops software to satisfy the requirements that hold for that product type.

To efficiently deliver the software for these types, the organization has developed a product-line architecture. The product-line architecture distinguishes several subsystems, each focusing on specific functionality, such as signal processing or the menu structure of the device. Subsystems are organized into layers. This is but a logical organization of the subsystems into manageable chunks. The relation between projects, layers and subsystems is depicted in Figure 1.

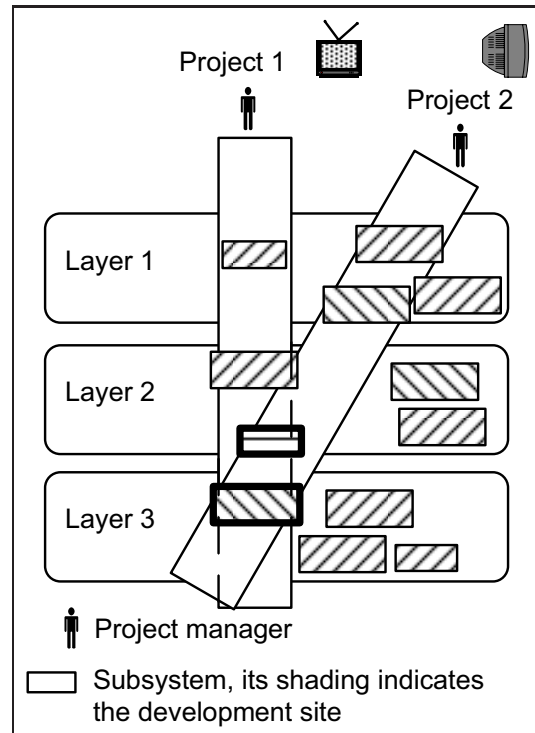


Figure 1. Projects versus subsystems. Subsystems in bold deliver software for use in more than one type.

Subsystems are developed by dedicated subsystem development teams, headed by a project manager and consisting of an architect, a configuration manager, and one or more software engineers. Each subsystem development team is located at a single site. Together, this organization into subsystems and the location of a subsystem development team at a specific site result in a certain amount of autonomy of the subsystem development teams. In order not to let this autonomy result in problems with integration of software from multiple subsystems, an architecture team is responsible for maintaining the product-line architecture. The architecture team consists of the chief architect, representatives of the major projects that are running, and some

main subsystem architects. Most members of the architecture team are located at one particular development site. To maintain the architecture, the architecture team issues architectural rules in small, text-based documents called architectural notes, or *archnotes* for short. Each archnote addresses a coherent set of architectural rules, e.g. all rules that have to do with naming conventions are captured in a single archnote. In total, 53 archnotes exist containing some hundreds of architectural rules.

Archnotes are mainly targeted at engineers, subsystem architects, and configuration managers. These employees are included in discussions on architectural decisions. Only decisions that concern all subsystems become rules, and are included in archnotes. A dedicated member of the architecture team documents the architectural rules.

Archnotes are placed on a central intranet website. New archnotes are communicated to the subsystem architects using a notification from a software configuration management (CM) system. The CM system provides a link to the archnote and some additional information. The subsystem architect uses the additional information to inform the employees in the subsystem development team of the new archnote. Each subsystem development team itself is responsible for complying with the architectural rules in the archnotes.

The organization experiences some problems in disseminating architectural rules and securing compliance with these rules within the organization:

- Engineers believe the formulation of architectural rules is too abstract. Therefore, they do not always read them. The extent to which architectural rules are read is presumed to differ across different development sites.
- Engineers do not always understand the architectural rules.
- Architectural rules do not cover all relevant decision topics.

Based on these problems, we hypothesized that a problem for not understanding and reading the archnotes was in the way the archnotes are structured. Improving this structure would then help in understanding the architectural rules. Following that, dissemination of the archnotes would be improved. Better dissemination of the knowledge in the architectural rules would then result in better compliance with the architectural rules.

We performed an assessment of the way architectural rules are secured to verify the aforementioned hypothesis. We used the assessment results to identify improvement points for the structure and use of the archnotes.

4. Performing the assessment

This section describes the phases of the assessment method as well as the results of applying the method in the case study. We used an assessment method that helps to reveal problems that exist in securing architectural compliance.

In terms of the SARA report [10], the objective of our assessment is to identify opportunities for improvement in architectural compliance. This objective corresponds with one of the concerns of the chief architect. The preparation of the assessment focused on getting clear the goal of the assessment (reveal problems that exist in securing architectural compliance), identifying the scope (the architectural rules), and selecting participants for the assessment (global architects and important roles in the subsystem development team, such as subsystem architect, engineer, and configuration manager). The assessment activities themselves focused not so much on the architecture itself, but rather on the guidelines and rules in place for the architecture. In observing the guidelines and rules, we specifically regarded both the structure of the architectural rules and the use of the architectural rules. The assessment was concluded by presentations of the assessment results (conclusions and improvement suggestions) to the management of the organization and several subsystem development team representatives.

We provide a general overview of the method. The next subsections describe each phase of the assessment method in more detail. In the next subsections, results from the case study are in *italic*.

Figure 2 provides an overview of the assessment activities:

1. Based on the objective of the assessment, our analysis starts with determining and analyzing the audience (population) of the architectural rules. This results in insight into the usage of the architectural rules at different sites.
2. Based on insight into the audience of the architectural rules, we construct a questionnaire. The questionnaire contains specific questions on the following topics: the relevance of architectural rules, the perception of participants of the role of architectural knowledge within the organization, and the participants' opinion of the architecture team. These questions aim to provide information on the hypotheses formulated in Section 3. Next, we send out the questionnaire to a selection of employees which includes at a minimum all major roles involved at the major development sites. We analyze the answers to the questionnaire to find e.g. conflicting statements and to find specific suggestions as to where to dig deeper.

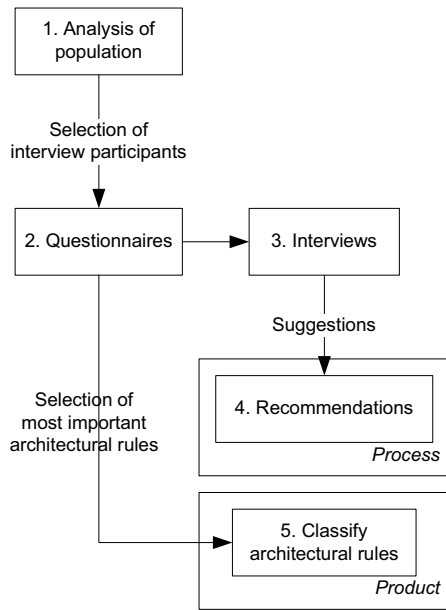


Figure 2. Overview of assessment activities.

3. Using the same selection criteria as with the questionnaire, we select a subset from the questionnaire participants to conduct interviews with. We use the interviews to foster specific suggestions for removing deficiencies in securing architectural compliance. The questionnaires and interviews may reveal potential issues in securing architectural compliance. For that reason, we send out the questionnaires to participants *individually* and conduct *individual* interviews [10].
4. The suggestions collected in the interviews in turn are used to provide recommendations to the management of the organization.
5. Insight into what architectural rules are most important for the participants' daily work enables us to study these rules. We can determine the structure that seems most appropriate to disseminate architectural rules. This may result in an improved classification of the architectural rules. Finally, the architecture team uses the list of most important architectural rules to see if these rules have a higher relevance and require a different way of securing compliance with them.

4.1. Analysis of population

The potential population that has to comply with the architectural rules consists of all employees that are engaged in the architecture to which the rules pertain.

The actual usage of the architectural rules across development sites provides information on that part of the population which actually reads them. We distinguish between different sites to see whether differences exist between those sites.

The organization at which we performed the case study develops its software using four main development sites. Each site has a main configuration manager role assigned. The configuration manager grants employees access to the source code tree as one of its tasks. We contacted the main configuration manager to obtain the number of employees involved in software development that reside at that site.

We used the log files of the intranet-site web server of the archnotes to get insight into the actual usage of archnotes. The analysis spans the period from January, 2005 until April, 2006. Figure 3 shows the usage of archnotes at the four main development sites. The total number of employees involved is 515. Of these potential users of archnotes, 288 have accessed at least one archnote.

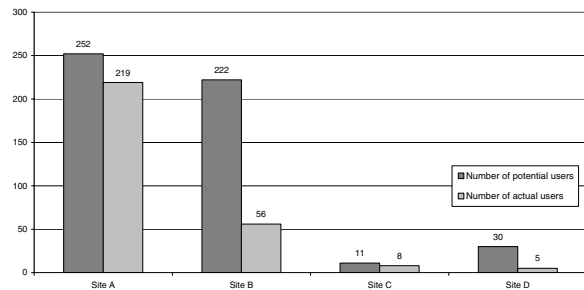


Figure 3. Usage of archnotes per site.

Although the potential number of users is roughly the same for the two major development sites (sites A and B), significant differences can be observed in the actual number of users from those sites that access the archnotes. The site with the highest ratio of potential versus actual users (site A) is the site at which the architecture team resides.

The actual usage of the architectural rules provides information on the number of queries ('hits') on architectural rules from different sites. This helps in identifying what kinds of architectural rules are accessed most often. In the next step of the method, among other things, we determine if these architectural rules are indeed the architectural rules deemed most important for the participants' daily work (and thus, whether existing rules satisfy the knowledge need).

Using the same log files, we counted the number of hits on archnotes. To prevent a bias, we regard multiple queries of a single user to a single archnote file within a period of two minutes as one hit. This two-minute threshold was chosen after a discussion with several architects – they indicated that every user of an archnote should be able to find the information in it within two minutes.

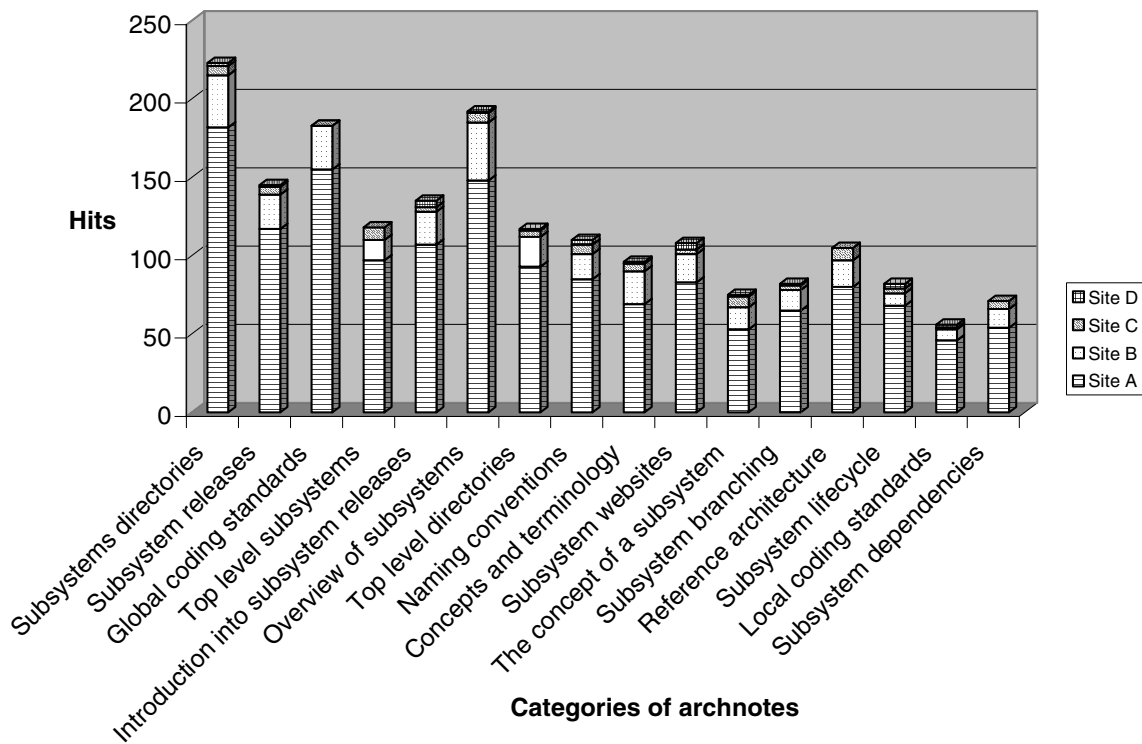


Figure 4. Number of hits per archnote per development site.

We wanted to know which archnotes were accessed more often than others. To this end, we plotted the number of hits per development site for the archnotes accessed most often – see Figure 4. The usage of archnotes provided valuable information as to what knowledge of the software architecture is accessed most often. Archnotes accessed most often pertain to software artifacts (such as subsystems¹, naming conventions, and branching of subsystems) and coding standards. These topics are typically covered in the development view or code view as covered by [8] and [13], respectively.

We used the analysis of the audience of the archnotes in two ways:

- To select participants for questionnaire-guided interviews to obtain insight into the actual status of securing architectural compliance and suggestions on how architectural compliance can be improved within the organization.
- To determine whether archnotes that are accessed

¹Especially the archnotes on subsystem lifecycle, describing possible states of a subsystem, and subsystem releases, describing the process and conventions used when making a release of a subsystem for integration purposes, are important in multi-site development organizations. Compliance with these archnotes enables efficient integration of subsystems.

more often are also regarded to be more important to be informed of architectural rules.

Analysis of the intranet website usage does not provide an undisputed view on the actual usage of architectural rules throughout the organization: other mechanisms besides accessing a repository can be used to share architectural knowledge, such as informal chats and team presentations. In the interviews we verify the importance of the intranet website in disseminating architectural rules.

4.2. Conduct questionnaires

The next phase of our assessment method starts with developing a questionnaire based on the results collected so far. We use a questionnaire to gain understanding of the underlying reasons for the usage of the architectural rules. Furthermore, we try to obtain evidence to support our hypotheses stated earlier.

The questionnaire consists of the following elements:

1. Demographic data – Obtain name, role, years of experience, and subsystem experience of the participant.
2. Knowledge sharing – Identify practices and supporting mechanisms used to share knowledge, architectural knowledge in particular.

3. Opinion on the architecture team – Identify the participant’s opinion on the role of the architecture team and the activities the architecture team undertakes to disseminate architectural knowledge and secure compliance.
4. Important architectural rules – Identify the importance of the architectural rules for the daily activities of the participant. In this step, we tailor the questionnaire to contain the actual list of architectural rules that are applicable to the organization.
5. Missing architectural rules – Identify what topics exist that require architectural rules, but do not have them.

The questionnaire is designed to obtain feedback from the assessment participants on each of the topics mentioned above. This feedback results in information on both the architectural rules themselves, as well as the activities that are in place within the organization to disseminate and secure the architectural rules. In addition, the questionnaire allows participants to bring forward suggestions for improving the current situation.

Based on the analysis of the audience of the archnotes we identified a representative subset of 20 employees. These employees include the most important roles from the four main development sites. We sent out the questionnaire to these employees and obtained the following results:

1. We included 15 participants in either the subsystem architect or engineer roles. The remainder of the participants were 2 configuration managers, an event manager, a quality assurance officer and a requirements manager. The participants covered 58 % of all subsystems and three of the four main development sites. We selected 9 participants from site A, 9 from site B, and 2 from site C.
2. Table 1 shows the use of knowledge sharing practices at the organization. We found that the most important mechanism to share knowledge is the formal mechanism of a change request. A change request is issued using the configuration management (CM) system. The importance of other mechanisms for knowledge sharing is significantly lower than the importance of a

Table 1. Practices for knowledge sharing

Practice	Number of votes
Change Request using CM	12
Informal chats	7
(Group) presentations	7
Teleconference	6
(Face-to-face) meeting	6

change request. These other mechanisms all are less formal than using a change request. Although these informal mechanisms are important in general, they are applied less often due to the nature of the multi-site development organization. This has also been reported in [11].

3. All participants regarded the role of the architecture team as pivotal for disseminating the architectural rules. Table 3 lists the opinion of the participants on the architecture team. The table shows the degree to which the participants agree with the statement. We observed the three statements with the lowest score (indicated in boldface in the table): 1) participants felt that they were less regularly updated on the subjects addressed by the architecture team, 2) participants indicated an improvement point in the response time of the architecture team on a specific request for the team issued directly or by delegation, and 3) participants felt that the architecture team can improve on the point of being in control of architectural activities. We investigated these reasons in order to foster suggestions for improvement during the follow-up interviews.
4. We asked the participants to indicate the ten most important archnotes for their daily activities. Table 2 shows the results. We see that most of the important archnotes are the ones that were accessed most often using the intranet website (see Figure 4). Consequently, we concluded that the intranet website serves as an adequate mechanism to disseminate the architectural rules.
5. As a final result of the questionnaire, we collected a list of missing architectural rules. Missing architectural rules concern architectural topics for which architectural rules are requested, but not present. The participants indicated to need architectural rules for the following topics:
 - Dynamic aspects (execution, memory consumption, inter-process calls) of the software architec-

Table 2. Most important archnotes

Archnote subject	Number of votes
Naming conventions	12
Subsystem branching	10
The concept of a subsystem	9
Overview of subsystems	9
Subsystem directories	8
Subsystem releases	7
Top level directories	6
Subsystems in layer 1	6

Table 3. Opinion on the architecture team

Opinion	% agreed
The goal of the architecture team is clear to me.	71.43%
I am regularly updated on the subjects addressed by the architecture team.	52.86%
I know where to find the architecture team's information.	80.00%
Decisions are properly made by the architecture team.	67.14%
Decisions are properly documented and deployed by the architecture team.	67.14%
The response time of the architecture team is OK.	56.92%
The architecture team is sufficiently in control of architectural and design activities.	60.00%
The representation in the architecture team is effective.	68.57%
I am aware of the existence of the reference architecture (see Figure 1 as an indication).	82.35%
The purpose and content of the reference architecture is clear to me.	78.57%
The content of the reference architecture is useful to me.	80.00%
I am aware of the architecture conventions (archnotes).	94.12%
The purpose and content of the archnotes is clear to me.	84.62%

ture.

- *Deadlocks and preventing deadlocks in software.*
- *Dealing with state changes in an end product.*
- *Integration strategies of different subsystems.*

These architectural topics pertain to dynamic aspects of the software architecture. Addressing these aspects may be done by introducing a process view according to [8]. In the absence of architectural rules for these topics, the knowledge is currently obtained through informal communication. Although informal communication is important, it reduces traceability of compliance with these rules.

The questionnaire results show no difference in opinion at different development sites on important knowledge sharing mechanisms, the architecture team, or the most important archnotes. We already observed a low usage rate from site B, compared to the potential number of users of archnotes. We attributed this to the fact that the architecture team resides at site A. Employees at site B received notifications of new architectural rules in the form of a new or updated archnote. Consequently, site B might regard the archnotes as not-invented-here and has a lower urge to read them and comply with the architectural rules. We used follow-up interviews to validate this hypothesis.

4.3. Conduct follow-up interviews

The questionnaire results are collected and analyzed to think up additional questions for the interviews. Possible questions arise from contradictions in the answers of participants and contradictions between the questionnaire response and the analysis of the population (see Subsection 4.1). In order to obtain unbiased feedback from the

participants, the interviews are held with the participants individually. The interview results were studied and combined to determine overall improvement points.

We interviewed seventeen interviewees from the list of participants. We selected only one employee in the case that multiple employees work on the same subsystem in the same role. In conducting the interviews, we specifically focused on the role of the architecture team to find reasons for the scores listed in table 3. Studying the interview results led to insight into the reasons for the scores. As such, the interview results offered the following suggestions related to the entries in boldface in table 3:

- *Several participants experienced uncertainty on certain architectural topics. These participants see projects deviate from the architectural rules for the sake of the project. The architecture team often participates in discussions and is aware of the deviations. Yet, these deviations are not communicated throughout the development organization. This lack of communication results in the opinion that the architecture team is not in control and trailing on projects.*
- *The visibility of the architecture team is further bothered by a lack of resources in capturing the architectural rules in archnotes. At the moment, nineteen archnotes have the status 'planned' and have not yet been written. Although a list of missing architectural topics was identified, these are not expected to be addressed within a few months.*
- *The effectiveness of the architecture team's meetings can be improved. Members of the team that were interviewed indicated that both the current frequency of the meeting and the attendance should be higher. They felt that this would improve the visibility of the archi-*

ecture team to the rest of the software development community.

Aside from these specific suggestions to improve the role of the architecture team, we identified other problems experienced by the participants. The most important problems that prevent the organization from securing the architectural rules at different development sites are listed in table 4.

At this point, our initial hypothesis that the problem for securing the architectural compliance was in the structure of the archnotes seemed to be wrong; all problems identified pertain to the topics of archnotes, the deviations from archnotes, and the lack of verification of compliance with archnotes that prevent securing architectural rules within the organization. These problems concern the process by which the architectural rules are captured, disseminated, and secured within the organization.

4.4. Provide recommendations

We used the interviews to collect suggestions on how to improve architectural compliance. In analyzing the compliance topics in the questionnaire, we focused on those answers that indicated that room for improvement was possible according to the participant. Next, we asked the participant how, from the participant's perspective, this improvement would be most expedient. The collected responses were categorized. The five suggestions mentioned most often by the interview participants are listed below:

- Provide an e-mail notification of changed or new archnotes to subscribers.
- Let the architecture team prioritize the archnotes according to relevance.
- Let projects write down deviations from archnotes.
- Let the architecture team verify compliance with archnotes.
- Let subsystem architects periodically discuss archnotes in their team meetings.

These suggestions were presented to the organization's management to obtain support for improvement of securing architectural rules in the organization – see Section 5.

4.5. Classify architectural rules

At this point, valuable suggestions are collected from the previous steps of the assessment. Nevertheless, an analysis of the structure of the architectural rules can be performed in order to reveal additional suggestions. This analysis starts with those architectural rules that were regarded

as most important for the participants' daily work. Expliciting the structure of these rules and applying this structure to other architectural rules is expected to improve the readability of the architectural rules. This in turn would improve dissemination of the architectural rules within the multi-site development organization.

In expliciting the structure of the architectural rules, the concept of architectural views is used. Architectural views prove valuable in structuring architectural knowledge, such as architectural rules [2].

During this step, the existing architectural rules are analyzed to determine whether they are structured using architectural views. If this is not the case, existing view models such as the ones described in [8, 13, 2] can be recommended.

Based on a study of the most important archnotes, we observe that they contain a variety of rules pertaining to the architecture. Furthermore, the architectural rules serve as an aid for multiple stakeholders. For example, an engineer would like to learn how the subsystem's code tree is organized to locate software assets. On the other hand, the configuration manager would like to learn how the code tree is packaged into a subsystem release to verify that a subsystem release is made correctly. Currently, the archnotes do not distinguish between stakeholders and their concerns. This hampers a stakeholder in finding the right information in the archnotes. Therefore, we need a classification of the architectural rules. A classification would improve finding the right architectural rules and consequently, complying with these rules. We proposed a classification in two ways:

- A specific architectural view. At the moment, the archnotes are not structured according to views. The architecture team could use the most important archnotes as perceived by the audience of the archnotes as a starting point. Next, the architecture team could reclassify the architectural knowledge in the archnotes and indicate what knowledge is of particular interest for what role from a compliance perspective. We already learned that the dynamic aspects of the software architecture are underexposed by the architectural rules. Specific attention could be directed towards closing this gap by reverting to the process view described in [8].
- A classification according to importance of compliance with these rules. The architecture team could indicate the relevance for compliance with architectural rules. Possible suggestions to discriminate in relevance are cost of non-compliance ("What will non-compliance with this rule cost me?") or interoperability ("What subsystems need to be changed when this architectural rule is issued?").

Table 4. Problems that prevent securing architectural rules

Category	Problem
Use of architectural rules	Projects deviate from terminology in architectural rules
	Projects deviate from architectural rules (way of working)
	Too many architectural rules exist
	Architectural rules are outdated or draft
Architectural topics	Too many architectural rules cover organizational aspects
	Too many architectural rules do not cover 'architecture'
	Some architectural rules contain too much text
Architecture team	The architecture team is trailing on the reality in projects
	Compliance with architectural rules is not verified

The organization regarded these suggestions as helpful to improve securing the archnotes. However, most of the suggestions of the participants done so far pertained to the process by which archnotes are created and disseminated. We concluded that the organization deems these process-related suggestions as more feasible to achieve improved compliance with the archnotes.

Introducing views in archnotes increases the structure since target stakeholders and concerns are explicitly addressed – this may improve the use of the archnotes. Moreover, formalizing the architectural rules in the archnotes enables compliance checking, which would be an incentive to use the archnotes as well. We also observed in this case study that formalized mechanisms such as Change Requests are successful in use. In conclusion, it could be hypothesized that a change in formalization of the archnotes could improve the process.

5. Results of the assessment

The case study led to several results. Firstly, the organization gained insight into the potential audience of architectural rules and their average usage of the rules. Secondly, questionnaire analysis and interviews revealed that the main problem with securing architectural rules within the organization was not so much in the rules themselves, or in the structure used to capture them. Rather, the real problems pertained to the process by which the rules are captured, disseminated, and compliance to the rules is secured within the organization.

In order to overcome these problems, we used the list of five specific improvement suggestions from the interviews. These suggestions were next presented to the management of the software development organization. After a period of reflection, the management decided to support four out of five improvement actions. Only the suggestion to send out change notifications of architectural rules was not supported, because management perceived this would result in an overloading of electronic communication. The supported

actions all aim to improve the process by which architectural rules are used within the organization and do not so much concern the way in which architectural rules are structured.

- **The architecture team prioritizes the archnotes according to relevance** – Priority is based on the architecture team's knowledge of the archnotes themselves and the insight gained from this case study. Difference in relevance is supported by the results of the interviews in that some archnotes do not discuss what the organization would call 'architecture', and that architectural rules on certain topics are missing. Furthermore, this prioritization offers the organization the possibility to tailor e.g. the dissemination process or compliance verification process to a specific archnote.
 - **Architects periodically discuss the archnotes in (subsystem development) team meetings** – Although the fact that archnotes exist is mentioned in subsystem development team meetings, the contents of archnotes is not discussed. This relatively little attention does not result in engineers paying attention to the rules. Addressing and explaining the rules instead is expected to increase the awareness in the rationale behind certain rules which is a prerequisite for compliance – it is expected to reduce the not-invented-here-syndrome. Explaining why a rule is the way it is [14] instead of just issuing the rules results in better understanding and compliance.
- In addition, we suggest that the architecture team uses techniques such as *Traveling Architects* [4] and visits other development sites. Visiting a development site and discussing archnotes helps to increase the number of employees from that site that read and understand the archnotes.
- **Projects write down deviations from architectural rules** – Several projects deviate from certain architectural rules for the sake of the project. Often, this fact

is known, but not communicated by either a change of the rule that was broken or a 'known deviation' to the architectural rule. Not communicating this deviation results in a lack of clarity about the authority of the rule. To prevent this, projects should provide a written statement that they deviate from an architectural rule, including the rationale for this deviation. Next, these written statements are sent to the architecture team to obtain approval.

- **Verify compliance with the architectural rules** – Compliance with architectural rules currently is not verified. Not verifying rules that are mandatory leads to a certain amount of indifference with engineers ("it probably is not that important then..."). This indifference is tackled by letting engineers include the architectural rules as explicit criteria in their regular review meetings. In addition, the quality assurance officer verifies that these reviews have been performed correctly. Verifying compliance leads to insight in the extent to which the software that is developed complies with the architectural rules set for it.

6. Conclusions and future work

Architectural rules are necessary in guiding, coordinating, and communicating software development efforts across multiple development sites. Architectural rules require additional guidelines on how to disseminate and secure the architectural rules within the different development sites to achieve compliance. This paper describes a method for assessing multi-site development organizations to determine deficiencies in securing architectural compliance. Application of the method on a case study showed an explicit role of the enforcing organizational unit and periodic communication of architectural rules across all sites is of paramount importance. Suggestions for lowering the not-invented-here-syndrome of the architectural rules at the development sites have been identified and are currently implemented.

In further research, we will determine the effect of the suggestions that were made. We intend to determine the current level of compliance with the archnotes. When the organization has implemented certain suggestions, we hope to experience an increased level of compliance with the archnotes.

Acknowledgement

This research has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering

Research via contract 638.001.406 GRIFFIN: a GRId For inFormatIoN about architectural knowledge.

References

- [1] M. Boasson. The artistry of software architecture. *IEEE Software*, 12(6):13–16, 1995.
- [2] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2003.
- [3] P. Clements, R. Kazman, and M. Klein. *Evaluating Software Architectures: Methods and Case Studies*. SEI Series in Software Engineering. Addison-Wesley Professional, Boston, 2001.
- [4] A. V. Corry, K. M. Hansen, and D. Svensson. Traveling architects - a new way of herding cats. In *Second International Conference on the Quality of Software Architectures (QoSA 2006)*, Stockholm, Sweden, 2006.
- [5] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch: why reuse is so hard. *IEEE Software*, 12(6):17–26, 1995.
- [6] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter. An empirical study of global software development: Distance and speed. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 81–90, Toronto, Ontario, Canada, 2001. IEEE Computer Society.
- [7] A. Jansen and J. Bosch. Software architecture as a set of architectural design decisions. In *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, pages 109–120, Pittsburgh, Pennsylvania, 2005.
- [8] P. Kruchten. Architectural blueprints - the 4+1 view model of software architecture. *IEEE Software*, 12(6):42–50, 1995.
- [9] P. Kruchten, P. Lago, and H. v. Vliet. Building up and reasoning about architectural knowledge. In *Second International Conference on the Quality of Software Architectures (QoSA 2006)*, Stockholm, Sweden, 2006.
- [10] H. Obbink, P. Kruchten, W. Kozaczynski, R. Hilliard, A. Ran, H. Postema, D. Lutz, R. Kazman, W. Tracz, and E. Kahane. Software Architecture Review and Assessment (SARA) Report. Technical report, 2002.
- [11] P. Ovaska, M. Rossi, and P. Marttiin. Architecture as a coordination tool in multi-site software development. *Software Process: Improvement and Practice*, 8(4):233–247, 2003. 10.1002/spip.186.
- [12] K. Smolander. Four metaphors of architecture in software organizations: Finding out the meaning of architecture in practice. In *2002 International Symposium on Empirical Software Engineering (ISESE'02)*, page 211, 2002.
- [13] D. Soni, R. L. Nord, and C. Hofmeister. Software architecture in industrial applications. In *ICSE '95: Proceedings of the 17th International Conference on Software Engineering*, Proceedings of the 17th International Conference on Software Engineering, pages 196–207, Seattle, Washington, United States, 1995. ACM press.
- [14] J. Tyree and A. Akerman. Architecture decisions: Demystifying architecture. *IEEE Software*, 22(2):19–27, 2005.