What You Ask Is What You Get: Understanding Architecturally Significant Functional Requirements

Preethu Rose Anish[1], Maya Daneva[2], Jane Cleland-Huang[3], Roel J. Wieringa[2], Smita Ghaisas[1]
TATA Research Development and Design Centre (TRDDC) [1], TATA Consultancy Services Ltd., Pune, India
{preethu.rose, smita.ghaisas}@tcs.com
University of Twente[2], Enschede, the Netherlands
{m.daneva, r.j.wieriga}@utwente.nl
Systems and Requirements Engineering Center (SAREC) [3] DePaul University, Chicago USA
{jhuang}@cs.depaul.edu

***Abstract*—Software architects are responsible for designing an architectural solution that satisfies the functional and non-functional requirements of the system to the fullest extent possible. However, the details they need to make informed architectural decisions are often missing from the requirements specification. An earlier study we conducted indicated that architects intuitively recognize architecturally significant requirements in a project, and often seek out relevant stakeholders in order to ask Probing Questions (PQs) that help them acquire the information they need. This paper presents results from a qualitative interview study aimed at identifying architecturally significant functional requirements' categories from various business domains, exploring relevant PQs for each category, and then grouping PQs by type. Using interview data from 14 software architects in three countries, we identified 15 categories of architecturally significant functional requirements and 6 types of PQs. We found that the domain knowledge of the architect and her experience influence the choice of PQs significantly. A preliminary quantitative evaluation of the results against real-life software requirements specification documents indicated that software specifications in our sample largely do not contain the crucial architectural differentiators that may impact architectural choices and that PQs are a necessary mechanism to unearth them. Further, our findings provide the initial list of PQs which could be used to prompt business analysts to elicit architecturally significant functional requirements that the architects need.**

***Index Terms*—Architecturally significant requirements, large-scale project delivery, exploratory case study, empirical research method, qualitative interviews, quantitative study.**

I. Introduction

Requirements Engineering (RE) activities often involve capturing both functional and non-functional requirements, describing respectively what the system needs to do, and how it is to be achieved. In a typical project, software architects design an architecture that attempts to meet these requirements. While RE and software architecture (SA) textbooks [1, 5] indicate this to be a smooth process, experiences from real-life projects in the IT industry show that in large-scale contexts [2, 3], engineering functional and non-functional requirements is often disjoint, and executed by different teams at different points of time. This poses various problems to large-scale projects in which the synchronization of requirements documents is error-prone, cannot be accomplished by one person, and where reliance on tacit knowledge is not a viable option [18, 23].

Typically, functional requirements (FRs) are collected and documented by business analysts and domain experts [1]. Non-functional requirements (NFRs[1]), on the other hand, are often dealt with by software architects and technology experts [2, 3]. The knowledge corpora for FRs and architectural solutions are often maintained separately [17]. These factors make it hard for architects to uncover requirements that may have a critical impact on their architectural decisions. For example, a FR statement, such as *Ability to receive notification when a transaction is made*, may carry a significant architectural impact. We refer to such requirements as Architecturally Significant Functional Requirements (ASFRs). Almost all FRs may be argued to have some degree of impact on SA; however, our focus is on those FRs that have *significant* impact. By this, we mean any FR that has a potentially broad impact across the system, is high-risk, possibly volatile, and if modified could involve expensive refactoring. The kind of impact an ASFR can have on the SA is usually not explicitly stated in the ASFR statement.

The business analysts generally do not have the requisite technical knowledge to differentiate ASFRs from other requirements, nor to ask the kinds of questions that are needed to extract the information architects need in order to make architectural decisions. This scenario is quite different from NFRs that often explicitly elicit architectural considerations. In fact, NFRs are well-known to have a tight relationship with SA [19, 20]. For example, a NFR such as *the system shall not be unavailable more than 1 hour per 1000 hours of operation* is clearly an availability requirement with explicit architectural impact. On the other hand, an ASFR such as *Ability to receive notification whenever a transaction is made* has an implicit architectural impact. Furthermore, the design solution would change according to the required *notification type*. As *type of notification* is not explicitly stated in the FR statement, an architect might make an assumption which could lead to incorrect architectural decisions.

Our preliminary study shows that practicing architects deal with these problems by means of Probing Questions (PQs) which are designed to seek out architecturally relevant information [24]. We define Probing Questions (PQs) as a

[1] As there is no commonly agreed umbrella term for such requirements, we have opted to use the traditional term of NFR, which is broadly understood in both industry and academia

mechanism through which business analysts can extract from their client the unspecified architectural differentiators hidden in the ASFR statements.

In the RE discipline, much of the published literature [9, 16, 19] on architecturally significant requirements has focused on quality concerns (or NFRs) related to security, reliability, performance, and other such attributes. Criteria to identify ASFRs from a set of FRs, and mechanisms to decipher their exact impact on SA have not been studied so far. The engineering activities for NFRs and those for FRs are often disjoint. In this paper, we focus on FRs which do not deal with quality concerns and are not stated in reference to NFRs, but which nevertheless are likely to significantly impact architectural decisions. More specifically, we take into consideration the perspective of software architects. We present an empirical study based on semi-structured interviews in which we used open-ended questions to explore the current practices and insights of architects from organizations in India, the United States and the Netherlands. We interviewed architects from the domains of Banking, Financial Services and Insurance (BFSI), Healthcare, Retail, Government, Telecommunications, and Airlines Management.

The main objective of our interview-based study was to qualitatively explore whether the architects have encountered hidden FRs of architectural significance in their projects; and if so, how they identify and interpret those FRs. Having learned from our preliminary study [24] that PQs are used by architects to seek additional architecturally relevant information, we set out to answer the following research questions (RQs):

RQ 1: *What categories of ASFRs in the covered domains contain hidden / implicit architectural impact?*

RQ 2: F*rom a software architects' perspective, what kind of information is discoverable using PQs?*

RQ 3: *How do architects use PQs to infer and explicate hidden architectural impact?*

The remainder of the paper is organized as follows: Section II provides background and related work. Section III details the exploratory study research plan and execution, section IV presents our results, and section V discusses them. Section VI presents a quantitative illustrative study, section VII is about validity threats and Section VIII concludes the paper with implications for research, practice and teaching.

II. Background and Related Work

This section summarizes relevant literature in RE and SA. To the best of our knowledge, there is no systematic empirical research that directly addresses our RQs. As a part of preparing this paper, we searched for empirical studies on characterizing ASFRs and deriving implicit architectural impacts that might be buried in them. We found seven publications [4, 6, 15, 16, 17, 21, 22] addressing these two topics. Although in the RE literature, there is much research on architecturally significant NFRs, we do not include it here as our focus is on ASFRs exclusively. Similarly, in the discipline of SA, a 2013 systematic review [25] on empirical studies indicated the prominent place of NFRs with respect to SA, but reported no study dealing explicitly with the topic of ASFRs. Moreover, the 2013 mapping study of Li et al. [26] on the use of knowledge-based approaches in SA indicated that the phenomenon of architectural knowledge recovery has been under-researched and "needs to be explored seriously" (p. 790). This contributed to the motivation for the present study.

We now summarize the seven studies relevant to our work. Chen et al. [4] presented an evidence-based framework to systematically characterize architecturally significant requirements (ASRs). Their framework consists of four sets of characteristics: Definition, Descriptive, Indicators, and Heuristics. In particular, their finding that ASRs tend to be described vaguely, is a motivation for this study. Ferrari and Madhavji [6] conducted an empirical study to understand the impact of requirements knowledge and experience (RKE) on SA activities. Their findings suggest that architects with RKE perform better than those without. The authors further discussed the possible implications of their findings on the areas of training, education and technology. Khan et al. [15] presented an analysis model that categorizes requirements dependencies that are architecturally significant in terms of change impact. They reported on an exploratory study based on the change history analysis of a real-life web-based information system in order to gather the most architecturally-significant requirements dependencies from their model. Niu et al. [16] introduced a practitioner-oriented approach to analyze and evaluate ASRs for enterprise systems. However, their focus is on NFRs exclusively. Salehin [20] investigated in a case study the extent to which requirements or requirements attributes' information is found to be missing during the SA process and the impact of that missing information on system architecting in terms of effort. The study suggests that architects actively search for strategies to reduce missing information in the requirements, so that the SA is designed with less effort. This, in turn, highlights the importance and the need for PQs as a vehicle for deriving ASRs from dispersed FR and NFR specifications. Ernst et al. [22] conducted an exploratory case study to understand the impact of technology constraints on SA. This study's focus was on COTS products. Last, Gross and Dörr [21] reported first results of explorative studies aimed at revealing software architects' information needs that should be fulfilled in software requirements specification. They indicated the relevance of certain requirements artifact types to SA, such as system responsibilities, data requirements, system functionalities, interactions, technical constraints, stakeholder goals and their suitable representation in terms of notations. Furthermore, these authors studied the variances among software architects in a group, regarding each one's information needs. In their observations, factors such as expertise and individual motivation might influence the information needs of software architects. The authors call for further research into the information needs of architects and the factors that precondition them. Our paper responds to this call. Drawing on the related work presented in this section, we chose to focus on PQs as a coping strategy to derive architecturally significant requirements from FRs. To the best

of our knowledge, this is the first time the concept of PQs is introduced.

III. The Exploratory Study Research Plan and Execution

We designed a qualitative interview-based research process by implementing R. Yin's guidelines for exploratory case study design [14]. A total of 14 practicing architects from India, the United States and the Netherlands, took part in our study. We used semi-structured open-ended in-depth interviews [13] that included (1) composing a questionnaire, (2) confirming the interview questionnaire with 3 experienced researchers, (3) implementing changes in the questionnaire based on the received feedback, (4) conducting a pilot interview to check the applicability and feasibility of the questionnaire to real-life context, (5) conducting the interviews with the practitioners by using the finalized questionnaire, and (6) following-up with those participants that demonstrated deeper knowledge or shared a more specific example of how the study's phenomenon happens in real-life. Interested readers can refer to the questionnaire at http://re.cs.depaul.edu/ASFR/QuestionnaireRE15.pdf.

The duration of each interview was between 60 and 90 minutes. All our participants were informed in advance of our research goals and the interview process. The interviews were on a one-to-one basis. Three researchers conducted the interviews (Anish met the 12 Indian participants, Daneva – the practitioner from the Netherlands, and Cleland-Huang – the architect from the United States). Table I presents details of the interview participants. At each meeting, each researcher and the respective interviewee used the questionnaire to guide the conversation. Our questionnaire included three sections designed to (i) collect information about each participant's experience and application domain (ii) collect perceptions and observations with respect to each practitioner's understanding of ASFRs, and (iii) collect practitioners understanding of PQs. As no substantial changes were made to the questionnaire after the pilot interview, we included the pilot interview also as part of the case study. Furthermore, during the interviews there were cases when other questions arose in addition to the ones included in the questionnaire. These questions were not previously anticipated, however the researcher conducting the study considered them interesting and pursued the interview in that direction.

As confidentiality agreements were a premise for the study, we cannot provide any information on the organizations employing the practitioners. However, as practicing architects, their professional profiles had the following common characteristics: (1) all of them work on large business information system projects, (2) all of them have worked as a developer in their initial professional years before taking roles of architects, (3) most of them are experienced in more than one domain, (4) except 2 participants (P4, P14), all others work on fixed price projects, and (5) except one (P14), all participants are architects from the vendor's side. For the qualitative data analysis, we deployed the guidelines of Charmaz' Constructive Grounded Theory building method [11], which is an approach used in social sciences to construct general propositions (called a 'theory' in this approach) from textual data. These general propositions summarize beliefs embedded in the textual data.

TABLE I. QUANTITATIVE STUDY RESULTS

Participant ID	Application domain	Country	Total experience as an architect (Years)
P1	BFSI , Healthcare	India	13
P2	Insurance	India	9
P3	BFSI, Healthcare	India	10
P4	Retail	India	5
P5	Banking	India	3
P6	BFSI, Retail, Government, Healthcare	India	7
P7	Telecommunication	India	6
P8	Banking, Medical	India	9
P9	Retail	India	12
P10	Insurance	India	14
P11	BFSI	India	15
P12	Insurance, Banking	India	11
P13	Banking, Airlines, Telecommunication	US	15
P14	Telecommunication	Netherlands	4

The method is exploratory and serves in situations where the researcher does not have pre-conceived ideas. It is driven by the desire to capture as much general information as possible from collected qualitative data and to allow the theory to emerge from the data. In this method, the propositions derived from one interview are compared to all previous interviews to check whether they are consistent with those interviews. Three researchers (Anish, Ghaisas and Daneva) first individually read the interview transcripts and attached a coding word to a portion of the text – a phrase or a paragraph. The coding words were selected to reflect the relevance of the respective portion of the interview text to a specific part of the studied phenomenon. The researchers then suspended their independent judgments regarding the concepts that were discovered individually and worked together to systematically reorganize their combined concepts into higher-level categories. This helped us understand the data better. We also shared and clarified assumptions to arrive at a consensus regarding how and why particular concepts were important.

IV. Results

This section presents our findings regarding our RQs. Figure 1 depicts some of the concepts and their relationships based on our findings. Therein, the dashed line is used to indicate that we enumerate a larger set of concepts than those depicted.

A. RQ1: What categories of ASFRs in the covered domains contain hidden / implicit architectural impact?

In the experiences of our participants, ASFRs were never sufficiently detailed in the FR specifications to a point that the architects could start designing the SA without eliciting further information from business analysts or domain experts. In the

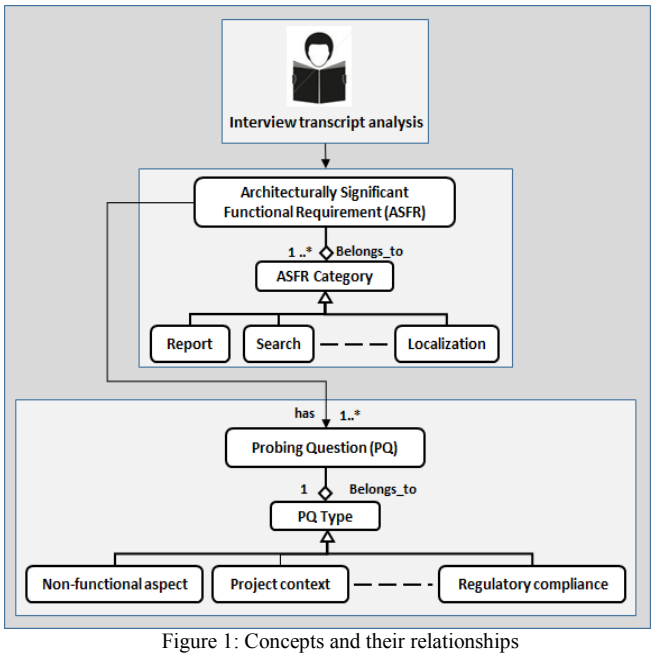

Figure 1: Concepts and their relationships

words of Participant P8: *The requirement is never elaborate enough to choose the correct architecture. There will always be a gap between our understanding and what the customer is conveying. Communication plays a key role"*. Our data analysis suggests that several hidden architectural implications are found in the ASFRs. Based on the kind of impact they have on architecture, we found that the following categories of ASFRs exist. Some of the examples are obfuscated for confidentiality reasons.

1. Audit Trail: Audit Trail facilitates auditing of system execution. Ten participants mentioned Audit Trail as an ASFR category. An example Audit Trail FR is: "*System must record every modification to customer records for audit purposes*." Participant P1 suggested: "*An important question is to understand the purpose of audit. Whether it is for regulatory compliance or for internal compliance. Depending on this, your audit architecture would change*".

2. Batch Processing: This includes requirements facilitating batch processing. Seven participants mentioned Batch Processing as an ASFR category. An example requirement is: "*The disbursement process should be a daily batch process for regular claim pay-outs"*. Participant P5 mentioned: "*A common thing to know is the frequency on which the batch process should run. One thing that we normally miss out is the privilege that it needs. What level of access specification you need for that batch to run?"*

3. Localization/Multilingual: Localization/Multilingual involves providing support for multiple languages. Six of the 14 participants mentioned Localization/Multilingual as an ASFR category. An example requirement is: "*During the term, the authority may require the contractor to deliver specific communications in other languages*". Participant P10 suggested: "*Localization has major impact on architecture. People usually go for upper end ERPs like Oracle or SAP as they have multi-language support*".

4. Business Process "State" Alerts: This class of ASFR is concerned with notifications, generated often as a part of executing a business process by the respective system. All 14 participants mentioned Business Process 'State' Alerts as an ASFR category. An example requirement is: "*Workflow is required to send notification to Underwriters*". Participant P5 mentioned: "*There are two types of notification. One is where we do not expect a response. Second one is where the transmitter wants the receiver to send back something. We had a scenario where we actually missed out that. Because we thought that it was just a notification we receive from the client; but the client wanted the notification to be acknowledged. That was one of the architectural changes we encountered"*.

5. Data-related Dialog: This category includes requirements specifying input data mechanisms. Participant P14 described the various ways to enter data into a billing system that processes discount vouchers: "*a voucher may be entered by means of QR code, by means of user's click on a link sent by email, or by user's manual data entry of voucher number*". For example, using a QR code assumes an external devise (a smartphone) be used as a QR code scanner, displaying the code and converting it to some useful form (such as a standard URL for a website). The smartphone would allow QR codes to send metadata to existing applications on the device and then hard-link it to an external URL.

6. Payment: This includes requirements facilitating financial transactions. Six participants mentioned Payment as an ASFR category. Participant P12 suggested: "*Payment is always a problem area. You need to know whether the payment is to be done through card or through net banking. Whether you need to link it with one bank or with multiple banks. All this will impact architecture*".

7. Print: This includes requirements facilitating support for printing documents. Seven participants mentioned Print as an ASFR category. An example requirement is: "*The commission statement should be printed using the package-supplied format"*. Participant P5 suggested: "*The problem is that they (clients) would always want a web application for printing. Now, if you use web application for printing, you need ActiveX-a component that deals with the printer from the website. It's an important component without which printer would not work. ActiveX has a security compromise. This is always a matter of negotiation between client and technical team because client would initially say to go ahead with printing from website. When the technical team is ready with printer and during deployment they have to reduce the security of the browser. Now client would say that it is not a good design to reduce security of the browser. So printing from browser is always a confusion. So this is one architectural decision*".

8. Report: This category includes FRs that facilitate Report generation. Ten participants mentioned Report as an ASFR category. An example requirement is: "*System should generate reports on complaint register on monthly basis*". Participant P3 suggested: "*Whether reporting requirement is for operational reports or for analytical reports? Whether it*

includes structured report or includes unstructured data as well? If somebody is giving some document and things like that and if you need to extract that information and try to produce some report, then it is more complex from architectural perspective than structured data".

9. Search: This category includes FRs that provide support for Search functionality. Ten participants mentioned Search as an ASFR category. An example requirement is: "*Claims Assessor should be able to search for claim records to be processed*". Participants P5 suggested: "*Search functionality is very big. This will need detailed level of analysis, whether we are looking for completely web-based search where we search for all text within web or we are looking for specific things, maybe I can have a drop down of all credit, I need to check all transactions of a particular client. You can have an advanced level search or a very generic search that will search through web. So for search you would need a detailed level of requirements gathering to come up with right architecture".*

10. Third Party Interaction: This category includes requirements that facilitate interaction with third party components. Six participants indicated Third Party Interaction as an ASFR category. An example requirement is: "*Once an application has been applied for, it will be exported to ABC back office where it may be processed both automatically by ABC back office system and manually by an underwriter*." Participant P4 mentioned: "*yes, third party interaction is critical. We have to understand where the third party tool is going to be installed. Is it inside client architecture or outside the architecture? What sort of patterns are acceptable by third party? All this would decide your architecture*."

11. Workflow: This includes requirements that provide support to move work items, facilitate reviews and approvals. Ten participants mentioned Workflow as an ASFR category. Participant P2 and P12 suggested: "*Workflow kind of requirements have a very big impact on architecture. After deployment, customer would say he wants a workflow included. The entire architecture has to be reworked in this case".* Participant P10 suggested: "*Workflow requirements are the key things to decide on your technical architecture, because workflows in organizations are not straightforward".*

12. Online Help: This category includes requirements pertinent to providing online help facility. Two participants mentioned Online Help as an ASFR category. An example Online Help FR is: "*An online help facility should be available for claim intimation process*". Participant P3 mentioned: "*Someone trying to save some form type of online help is different from someone trying to just navigate for some information. Is it only information consumption type of online help or is it about how you navigate to get to the right information? Second one could be that you are the contributor of information, you need help on how to populate the information. So consumer verses provider could be one sub-category, and internal verses external could probably be another".*

13. Licensing: This includes facilitating services for acquiring, installing and monitoring license usage. Three participants mentioned Licensing as an ASFR category. An example requirement is: "*There should be facility for installing and monitoring license usage*". Participant P3 suggested: "*Yes this is also very important from architectural perspective. For licensing there are different models available in market for different systems. Some could be user based, some could be CPU or system based, and some could be perpetual licenses. So, many variants are available. Then, based on the probing questions you could choose the right model".*

14. User Behavior Analysis*:* This includes FRs implied by the business goals of the client organization to collect, analyze and aggregate data on users' behavior. Examples of such ASFRs came from architects' experiences in web-based systems with consumer-oriented front end, for example, e-commerce applications and retail systems. In these system's contexts, architecting for User Behavior Analysis means implementing web monitoring features. As Participant P1 says, features for tracking the *"number of hits, number of page views, number of conversion, and number of orders processed".*

15. Storage Mechanisms: This type of ASFR includes features needed for automatic handling of documents, converting paper to electronic files and integrating the files with client organization's legacy business and possibly customer relationship management systems. Participant P3 indicated: *"The customer said that he wants to store all documents for x number of years. This looks very simple but from an architectural perspective, how would you store it, how would you maintain it is important".*

B. RQ2: From a software architects' perspective, what kind of information is discoverable using PQs?

We found that all our participants in fact were using PQs as a routine approach to ensure that the ASFRs included sufficient information to support architectural decisions in their projects. The PQs essentially unearthed the unspecified architectural differentiators hidden in the ASFR statements. Participant P8 suggested: "*The scope of requirement again affects architectural design. Client might initially say that print is needed. When we drill down the requirement further, we see these types of complexities".* Based on the concerns in the PQs specified by the architects, we found that the following types of PQs exist to bring out the unspecified architectural differentiators in ASFR statements:

1) ***Business rules in client's organization:*** Our analysis suggests that most of the PQs are about the possible constraints on software system's behavior and/or the constraints that provide support to system's behavior in client's organization. For example, in the case of Audit Trail ASFR category, participant P1 suggested the following PQs which in fact point towards one or more business rules in client's organization:

1. How do you enable audit trail?
2. How long does the audit trail need to be kept?
3. What audit trails need to be maintained?
4. What information does business want to see in the audit trail?
5. How often will the audit trail be retrieved?

2) ***Strategic technology choices****:* PQs of this type are concerned with the architectural impact of the investment decisions of client organization in specific hardware and software technologies. For example, users' devices that should be accounted for as part of technology infrastructure on which the system (for which an architect is responsible) would run or specific statistics that need to be collected, tracked and aggregated at system's run time, for user's behavior analysis. Such PQs were put forward by participants working in e-commerce systems or retail web-based systems that assume the implementation of specific tools for web monitoring.

3) ***Non-functional aspects related to ASFRs:*** Many PQs referred to relevant non-functional aspects that should be considered while making an architectural decision. PQs of this nature unearthed NFRs, which if not probed, would be left unattended or assumed. For example, participant P2 suggested the following PQs related to scalability and security:

1. What is the volume of data?
2. How many master tables are needed for audit purposes?
3. What level of security is needed in case of sensitive data?

4) ***Regulatory compliance:*** PQs pertinent to regulatory compliance have major influence on architectural decisions. This was explicitly suggested by five of the interview participants. Participant P1 said in case of the Localization category of ASFR: "*Are there regulatory or legal requirements in that region that necessitate a particular kind of content delivery? Regulatory requirement is a big thing"*. Regarding the Audit Trail category of ASFR, Participant P3 mentioned: "*An important question is to know the purpose of audit. Is it for regulatory compliance or for internal compliance? Depending on this your audit architecture would change"*.

5) ***Project Context:*** These are PQs concerned with contextual project parameters that impact SA. This includes PQs pertaining to functional fit, business case, cost considerations, time frame, technology stack available with the client, skill set availability on both vendor's and client's side, organizational culture, mentality of the client's organization, and even strategic requirements imposed on the project. On cost considerations, participant P8 suggested: "*with SMS, you need to consider whether a service provider is needed. You need to see whether you need a free service or you have budget to pay for the service. Customer says you just send SMS, I don't want to pay for it. In this scenario you need to go for free providers. Thus, customer's budget and timeframe will affect architectural decision*". Furthermore participant P3 said: "*I think you need to understand why typically you should not get into the batch operation unless there is a valid reason for it. So, why you need a batch process would be a probing question. Is it because of the technology or is it because of some real functionality?*" On strategic considerations, participant P14 indicated: "*Sometimes it is hard to arrive at a consensus. So the requirements you are asking about are strategic requirements. Whose systems to keep? Whose systems to phase out?"* On business case, participant P14 suggested: "*The impact depends on whether we need a temporary solution till the period we are waiting for the adoption of a better platform. In this case, you can tolerate a less good architecture. We will drop the system as soon as we update the infrastructure or we make a long term investment. We have a portfolio of infrastructure projects that phase out old systems and get new ones. So, the impact depends on where in this spectrum your project is. In other words, the business case matters"*.

6) ***Compound architectural effect on two or more ASFRs***: A PQ of this type provides an answer to more than one category of ASFRs. Participant P6 used this PQ when asking for payment transaction processing details: *"What confirmation would you be receiving from third party gateway to indicate success of payment?"* This PQ jointly addresses the ASFRs of Business Process 'State' Alerts and the Third Party Integration categories.

C. RQ 3: How do architects use PQs to infer and explicate hidden architectural impact?

Our participants indicated that asking PQs seemed an intuitive and common sense approach to unearthing the unspecified architectural differentiators from the ASFRs. We found that their reasoning about what to include as PQs was traceable to one or more of the following: (1) domain knowledge, (2) the architects' knowledge about the nature of the customer's business and in the vendor's processes of delivering projects, and (3) maturity of the client organization. More specifically, in the perceptions of our practitioners, these concepts had the following roles in the architects' judgments of what PQs to ask:

1) Domain knowledge was deemed critical in operationalizing ASFRs in terms of detailed steps (for example, as illustrated in the Audit Trail category of ASFR in the previous section). Furthermore, domain knowledge is instrumental in bringing out the constraints on system behavior. Participant P13 stated: "*Yes, you can recognize ASFRs immediately if you have experience of a domain. In a new domains you may be suspicious about them, but with some digging [PQs] you could classify them. Experience plays a significant role"*.

2) The architects' knowledge about the nature of customer's business and about the vendor' processes of delivering projects was important for formulating the PQs concerned with the clients' infrastructure and strategic technology choices. Participant P2 mentioned: "*The application is needed on desktop, laptop and tablet. Customer will not explicitly say that he wants it on the tablet. After implementation, they would say that they want it on tablet. Generally if you are not a seasoned architect, you would not take care of it"*. Also in the experience of P2: "*You do not want to recommend an open source solution to a health care customer where data availability is a very critical requirement and you do not want to depend on community supported systems for such critical requirements. You may want a solution suitable for mission critical systems"*. Example of knowledge about vendor's processes are the PQs concerned with parts of the business case. For example, money considerations, resource

availability, business criticality and affordable or desired time frame for the project.

3) Maturity of the client organization in terms of process-oriented thinking was deemed important for judging the relevance of PQs to be asked, because it is reflected in the way the FRs are documented. In the experience of our participants, more mature client organizations usually write their FRs with more discipline, which ensures breadth and depth of the documents provided to vendors. As Participant P6 suggested: "*The maturity with which FRs are documented varies considerably from one client to another. Sometimes we get FRs which are as good as technical specifications. But sometimes we get a FR which is at a very abstract level. Thus PQs will depend on the level of maturity and detail with which the FRs are mentioned"*.

We were also interested in discovering how the architects acquired answers to the PQs. We found that this depended on (i) the project organization's setup, and (ii) organizational hierarchy of the architect. We found that most of the time, the architects escalate the PQs to the respective RE-specialist (the business analyst in their projects), who then forwarded the PQs to the business managers at the client's organization. However, sometimes, the architect was a person with skills from both business and architecture side, in which case the architect posed the PQs directly to the clients.

Further, we wanted to know what the architects do with the answers to the PQs. Twelve out of our fourteen participants, amended the FR document by adding the newly-discovered ASFR details. In one case, the architect used the repository of his company's Architecture Office to document the ASFRs pertaining to his particular project.

V. Discussion

This section compares our finding to those in previously published studies relevant to our work. It also provides possible explanations about why we observed what we observed. Our discussion follows our research questions.

A. RQ1: What categories of ASFRs in the covered domains contain hidden / implicit architectural impact?

We found 15 categories of ASFRs. One of these categories, the Third Party Interaction, overlaps with the infrastructure-related ASFRs studied in COTS context in [22]. Our findings agree with the results of Ernst et al. [22] that components of the infrastructure – tools, platforms, converging technologies, may have significant architectural impact that are not obvious when the FRs are elicited and tools selected. However, we found 14 additional categories of ASFRs that have not been addressed in prior empirical research. This suggests that the world of ASFRs is richer and more diverse than what was previously documented.

The presence of these categories in our results can be explained as follows: All our participants design enterprise information systems, which in fact are reactive systems [35] with some 'information provisioning' function − a function to facilitate business processes to provide communication among actors involved in those processes. Knowing that the architects design enterprise systems, the categories of ASFRs identified so far, in fact pertain to user functionality of the following types: (a) providing memory (Audit Trail, Data-related Dialog, User Behavior Analysis, Storage Mechanisms), (b) providing information about memory for users (Print, Report, Search), (c) facilitating processes in the user environment of the system (Batch Processing, Business Process "State" Alerts, Payment, Workflow, Licensing), (d) facilitating or prompting communication among actors and/or systems (Third Party Interaction), and (e) supporting functionality (that is needed to provide services to the user, such as Localization and Online Help). Most ASFR categories were indicated by more than one participant from different domains. Therefore the ASFR categories we found seem to be applicable across the domains we studied. However, we expect that as we include more diverse domains such as safety critical systems or systems with real-time command and control requirements into our study, we would be in a much better position to classify ASFR categories as domain-agnostic versus domain-dependent. Therefore, more research into these aspects is necessary if we are to understand the range of possible mechanisms for narrowing the gap between RE and SA.

We found that architects are skillful in spotting missing ASFRs or ill-specified ASFRs. As participant P14 explained, in his practice he had the habit of red-flagging suspicious FRs that could potentially derail his SA design (for example, features that have very few users were routinely inspected for potential negative effects they might have on the SA): "*I also have other 'red-alert requirements'. That is how we call requirements that are problematic for architecture. For example features that do not have many users. These get ignored until later when someone figures out that an exotic feature of a VP has not been considered for implementation*". We therefore think it is important to explicitly augment the ASFRs with the architectural insights derived from PQ based conversations.

B. RQ 2: From a software architects' perspective, what kind of information is discoverable using PQs?

We found six types of PQs used by architects to discern ASFRs. We found that PQs embed architectural knowledge and help greatly in deriving non-functional details relevant to specific FRs. This is close to what other researchers refer to as 'scenarios' in [5] or what are defined as 'fit criteria' in [32]. Reflecting on the six types, one might think these types are structured as the layers of an onion: system - project- business-society. Considering the system, the architect wants to know if ASFRs interact with each other (such as in PQ type 6) or with NFRs (such as in PQ type 3). Concerning the project, there may be financial or regulatory standards that impact an ASFR (such as in PQ type 5). The business may impose, or experience, constraints as a result of an architectural choice (such as in PQ type 1) and it may have strategic considerations that impact an ASFR (as in PQ type 2). It is important to question our interpretation; we may have misunderstood the PQs. However the fact that the interpretation assumes a generic onion-like layered structure makes us think that it is possible to observe our findings in other contexts where such a structure is present. That is, our evidence suggests that in other projects, architects

would ask similar PQs to unearth the unspecified architectural differentiators from the ASFR statements. Additional case studies will be needed to validate this.

C. RQ 3: How do architects use PQs to infer and explicate hidden architectural impact?

We found that domain knowledge is central to an architect's judgment on what PQs to ask. This agrees with studies on the software architects' professional skillset and qualifications [27], which indicate the criticality of the architect's strong knowledge of the targeted problem domain. We also found that in some cases what was obvious to the client was not self-explanatory for the architects. This confirms the previous findings that stakeholders' tacit knowledge is often not well articulated [34]. For example, when a client requested an SMS notification, the architect working on this project assumed that no email notifications would be included. The client found it hard to relate to the architect's assumption and thought that sending email notifications was an "obvious requirement". There are several plausible explanations for this. For example, in a large project where the client's business representatives and the respective architects may be in two different countries, with different email usage levels and practices, a disconnect might arise regarding what is obvious and what is not. Domain knowledge alone seems insufficient in such cases, as it is also important to understand how the domain knowledge is situated in the practical context of the client and its organizational culture. This example also suggests that if business analysts' watch for "obvious requirements", they could be of great service to the architects and help prevent architectural rework.

We found that the architects' knowledge about the nature of customer's business and the vendor's processes for project delivery were instrumental for the generation of PQs. In fact, PQs were the architects' way to approach the elicitation of ASFRs as *a reflective conversation* between clients/business analysts and software architects. This finding is new in the sense that it has not been discussed so far in empirical studies on ASRs. However, we do not find it particularly surprising. It could well be explained by using Schön's model [27] of consulting practice according to which consulting is about assisting clients to solve their organizations' IT problems by the complementary use of consultant's expertise and clients' knowledge of their business in a process of framing and negotiating various perspectives. The act of asking PQs suggest that software architects act as experts who assist their clients in improving the precision of their FRs definitions. By asking PQs, software architects demonstrate their commitment to having their positions confronted and tested so that the risk of making suboptimal architectural design choices is reduced. Moreover, each of our 6 types of PQs frames the ASFRs in a particular way. Following Schön, a consulting process that uses alternative ways to frame a problem (in this case the ASFR) is grounded on two types of consulting expertise: knowing-in-action and reflection-in-action. Knowing-in-action is defined as the consultants' professional knowledge, their employer's appreciative system, and frames of a particular community of practice (in this case, the architect's community of practitioners), all of which serve as the source of coping strategies when solving problems in a consulting intervention. In addition, reflection-in-action [27] (p.125), means focusing on certain details of the problem while leaving others in the background, thus framing the problem in a particular way. If we view our software architects from the perspective of consultants embedded in communities of practitioners (i.e. architects employed at a vendor's organization and engaged in client's projects), the PQs clearly are a coping strategy for solving SA-related problems (in this case, discerning architectural differentiators from ASFRs).

VI. Quantitative Evaluation Study

In order to validate our theory against actual software requirement specification (SRS) documents, we conducted a preliminary evaluation using real-life SRS documents. The goal of this first evaluation was to validate if the SRS documents in fact lack the needed architectural differentiators to make informed architectural decisions. We took 450 FR statements from 30 real-life SRS documents (from the Insurance domain), using Castillo's stratified sampling technique [33]. This is a probability sampling technique wherein the researcher divides the entire population into different subgroups (or 'strata'), then randomly selects the final subjects proportionally from the different strata. In our case, the strata were the projects and the lines of businesses (such as life, property and causality, health, marine etc.) within the insurance projects. Two of the authors (Anish and Ghaisas) studied the 450 statements and identified 199 ASFR statements from them. These 199 ASFR statements were then manually classified into the 15 ASFR categories presented in Section IV A above. However, in the studied SRS documents, we found FRs belonging to 10 out of the 15 categories. For each of those 10 categories, we compared the number of architectural differentiators addressed in FRs statements ($AD_{PRESENT}$) and the number of architectural differentiators identified in the interview study (AD_{IDENT}). This was needed in order to see how many differentiators occur in the sample.

TABLE II. QUANTITATIVE STUDY RESULTS

ASFR Category	FRs	AD_{IDENT}	$AD_{PRESENT}$
Business process 'state' alert	60	3	1
Search	21	10	1
Print	24	4	0
Audit Trail	21	11	2
Batch Processing	10	14	3
Localization	5	6	1
Report	7	19	5
Workflow	23	14	2
Payment	21	3	1
Third party interaction	7	2	0

FRs: Number of functional requirements in the SRS
AD_{IDENT} :Number of architectural differentiators identified from interviews
$AD_{PRESENT}$:Number of architectural differentiators addressed in FR statements

Table II presents this quantitative comparison. We found that very few (0% to 33 %) architectural differentiators were actually present in the ASFR statements. This strengthens our

working hypothesis that ASFRs do not explicitly state the architectural differentiators. PQs are a necessary mechanism to unearth them.

VII. Validity Threats

There are three inherent weaknesses of interview techniques [13] that we acknowledge in this study. First is the extent to which the participant's answered our questions truthfully. We minimized this threat by involving volunteers, under the assumption that if a practitioner cannot be honest, he/she could decline his/her participation at any stage of the research. Secondly, it is possible that the researcher might instill her bias in the data collection process. To counter this threat, we applied Yin's recommendations [14] in this respect by (i) including participants working in diverse application domains, which allowed the same phenomenon to be evaluated from diverse perspectives (data triangulation [14]); (ii) sending the interview answers to each participant prior to data analysis to confirm the information he/she provided; and (iii) getting the data analysis report reviewed by some interviewees (some of whom provided feedback on clarifying the concepts, but this did not change the analysis). The third weakness is the possibility of asking leading questions — suggestive and implying that one answer might be better over others. We, however, made every attempt to leave our interviewees to respond on their own terms, expressing their own perspectives and values, and giving examples of what they do and how they cope in particular projects.

We used the checklist in [9, 35] to assess the possible threats to validity in this study. As it is exploratory, an important question regarding the validity of our results is: to what extent can our practitioners' observations be considered representative for other project organizations? While not all settings in which the software architects infer hidden meanings from ASFRs are similar to the case study companies, some of them are [10, 36]. For instance, other vendors of large scale solutions that deliver enterprise information systems that serve clients in the business domains in which our architects worked, and that must ensure compliance to business sector specific regulations, might experience phenomena similar to those in this case study. Next, our quantitative evaluation (Section VI) found that 10 out of the 15 ASFR categories were present in the studied FR documents. We think the other 5 categories were not present because of less relevance to the domains of the projects included in the quantitative study (we included FR documents only from the insurance domain as these were easily accessible to us). However, further research is needed to uncover the contexts in which the remaining 5 categories would be more relevant.

VIII. Conclusions and Implications

This exploratory study brings out how software architects cope with the process of inferring hidden meaning from ASFRs that are buried in large scale project documentation. We found 15 categories of ASFRs. We also found 6 types of PQs that are indispensable for revealing client-organization-specific and project-specific architectural differentiators that have impact on the solutions' architecture.

Our study has implication for RE research, practice, and teaching. We noticed that architects' PQs were grounded on their domain knowledge. An interesting line of research would be to automate the identification and classification of ASFRs and recommending relevant PQs to the business analysts and architectural solutions to the software architects. For example, we can use machine learning techniques such as association rule mining to link the detected ASFR patterns with the right set of PQs that are necessary to be asked. We plan to draw on our earlier experiences on automating the detection of artifacts from text [24, 29 and 30]. Moreover, the domain-dependent PQs can be linked to augment existing knowledge repositories in organizations [31]. In order to have an effective mechanism that is able to accurately identify ASFRs from FR documents, indicate impact possibilities comprehensively, and offer design solutions more pointedly, we need a collaborative mechanism wherein experts can contribute and curate requirements and architectural knowledge. We plan to harness our previous work for this purpose [31]. Further, it would be interesting to find out whether agile approaches would be comparatively more or less effective in revealing hidden ASFRs.

From RE practitioners' perspective, knowing which ASFR categories have architectural impact can help business analysts to elicit a more complete set of requirements. This provides the information that architects need to make informed decisions and can potentially reduce wasted effort caused by the need to rework the solution later in the project.

Finally, it has been the observation of the authors that RE textbooks often overlook the important synergies between RE and other software engineering activities such as SA. As more attention is paid to the RE-software architect relationship, it is worthwhile to revise RE teaching by including concepts that help RE produce requirements documents that actually help architects come up with suitable architecture that better fits the client's business. In particular, our study suggests that complementing RE training with education on business rules (PQ type 1 in section IV B) might be a worthwhile option.

Acknowledgment

The authors thank the participants for taking time out of their busy schedules to talk with the researchers. Cleland-Huang's participation was partially funded by United States National Science Foundation Grant # CCF-0810924.

References

[1] Lauessen, Software requirements – styles and techniques, Wiley, 2002

[2] D. Ameller, C.P. Ayala, J. Cabot, X. Franch, How do software architects consider non-functional requirements: An exploratory study. RE 2012: 41-50

[3] M. Daneva, L. Buglione, A. Herrmann. Software architects' experiences of quality requirements: What we know and what we do not know? REFSQ 2013: 1-17

[4] L. Chen, M. Ali Babar, B. Nuseibeh, Characterizing architecturally significant requirements, in IEEE Software, 30(2)2013: 38–45

[5] L. Bass, P. Clements, R. Kazman: Software Architecture in Practice, 2nd edition, Addison-Wesley, 2003

[6] R. Ferrari, N.H. Madhavji, The impact of requirements knowledge and experience on software architecting: An empirical study, WICSA 2007: pp. 16

[7] M. Brandozzi, D. E. Perry. From goal-oriented requirements to architectural prescriptions: The Preskriptor Process, STRAW 2003

[8] A. Egyed, P. Grunbacher, N. Medvidovic, Refinement and evolution issues in bridging requirements and architecture – The CBSP approach, STRAW 2001

[9] P. Runeson, M. Höst., Guidelines for conducting and reporting case study research in software engineering. ESE 2009, 14(2), pp 131-164

[10] P. Seddon, P. Scheepers, Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples, EJIS, 2011, pp. 1-13.

[11] C. Charmaz, Constructing Grounded Theory: A Practical Guide through Qualitative Analysis, Sage, Thousand Oaks, 2006.

[12] S. Ghaisas, P. Rose, M. Daneva, K. Sikkel, R.J. Wieringa, Generalizing by similarity: Lessons learnt from industrial case studies, CESI 2013, pp. 37-42

[13] N. King, C. Horrock, Interviews in qualitative research. Sage, 2010

[14] R.K. Yin, Case study research, Sage, 2014.

[15] S. S. Khan, P. Greenwood, A. Garcia, A. Rashid, On the impact of evolving requirements-architecture dependencies: An exploratory study, CAiSE 2008, pp 243-257.

[16] N. Niu, L.D. Xu,, J.C. Cheng, Z. Niu., Analysis of architecturally significant requirements for enterprise systems, IEEE Systems Journal, 8(3), pp. 850-857.

[17] T.M. Hesse., B. Peach, Supporting the collaborative development of requirements and architecture documentation, TwinPeaks 2013, pp. 22-26

[18] B. Paech, Av. Knethen, J. Dörr, J. Bayer, D. Kerkow, R. Kolb, A. Trendowicz, T. Punter, A.H. Dutoit, An experience-based approach for integrating architecture and requirements engineering. STRAW 2003, pp. 142-149

[19] A. Koziolek, Architecture-driven quality requirements prioritization, TwinPeaks 2012, pp. 15-19

[20] M.R. Salehin, Missing requirements information and its impact on software architectures: A case study, Master's thesis, The School of Graduate and Postdoctoral Studies, The University of Western Ontario, London, Ontario, Canada, 2013

[21] A. Gross, J. Dörr., What do software architects expect from requirements specifications? Results of initial explorative studies, TwinPeaks 2012, pp. 41-45

[22] N.A. Ernst, I. Ozkaya, R.L. Nord, J. Delange, S. Bellomo, I. Gorton, Understanding the role of constraints on architecturally significant requirements, TwinPeaks 2013, pp. 9-14

[23] P.R. Anish, B. Balasubramanium, A knowledge-assisted framework to bridge functional and architecturally significant requirements, TwinPeaks 2014, pp. 14-17

[24] P.R. Anish, B. Balasubramaniam, J. Cleland-Huang, R. Wieringa, M. Daneva, S. Ghaisas, , Identifying Architecturally Significant Functional Requirements, TwinPeaks 2015, Accepted

[25] N. A. Qureshi, M.B. Usman,.,N. Ikram, Evidence in software architecture, a systematic literature review, EASE 2013, pp. 97-106.

[26] Z. Li, P. Liang, P. Paris Avgeriou. Application of knowledge-based approaches in software architecture: a systematic mapping study, Information & Software Technology, 55, 2013, pp. 777-794

[27] P. Clements, R. Kazman, M. Klein, D. Devesh, S. Reddy, P. Verma, The duties, skills, and knowledge of software architects, WICSA 2007, pp. 20

[28] D. Schön, Educating the reflective practitioner, San Francisco: Jossey-Bass, 1987.

[29] J. Cleland-Huang, R. Settimi., X. Zou, P. Solc, Automated classification of non-functional requirements. Requirements Eng. 12(2): 103-120, 2007

[30] S. Ghaisas, M. Motwani, P.R. Anish, Detecting system use cases and validations from documents, ASE 2013, pp.568-573

[31] S. Ghaisas, N. Ajmeri, Knowledge assisted ontology-based requirements evolution, MaRK 2013, pp. 143-167

[32] S. Robertson, J. Robertson, Mastering the requirements process (2nd Edition), Addison-Wesley Professional, 2006

[33] J. J. Castillo., Stratified sampling method, https://explorable.com/stratified-sampling. Last accessed 27-2-15

[34] G.R. Gacitua, L. Ma, B. Nuseibeh, P. Piwek, A. De Roeck, M. Rouncefield, P. Sawyer, A. Willis, H. Yang, Making tacit requirements explicit. MaRK 2009, pp.40-44

[35] R.J. Wieringa, Design methods for reactive systems. Yourdon, Statemate and the UML. Morgan Kaufmann, San Fransisco, USA, 20

[36] R.J. Wieringa, M. Daneva, Six strategies for generalizing software engineering theories. Science of computer programming, Online First, 2015, pp. 100.