Prioritizing Architectural Concerns

Lars Pareto
Department of Computer Science and Engineering,
Chalmers | University of Gothenburg
Gothenburg, Sweden
pareto@chalmers.se

Anna Sandberg, Peter Eriksson, Staffan Ehnebom
Ericsson AB
Gothenburg, Sweden
{anna.sandberg, peter.r.eriksson, staffan.ehnebom}@
ericsson.com

***Abstract*—Efficient architecture work involves balancing the degree of architectural documentation with attention to needs, costs, agility and other factors. This paper presents a method for prioritizing architectural concerns in the presence of heterogeneous stakeholder groups in large organizations that need to evolve existing architecture. The method involves enquiry, analysis, and deliberation using collaborative and analytical techniques. Method outcomes are action principles directed to managers and improvement advice directed to architects along with evidence for recommendations made. The method results from 3 years of action research at Ericsson AB with the purpose of adding missing views to architectural documentation and removing superfluous ones. It is illustrated on a case where 29 senior engineers and managers within Ericsson prioritized 37 architectural concerns areas to arrive at 8 action principles, 5 prioritized improvement areas, and 24 improvement suggestions. Feedback from the organization is that the method has been effective in prioritizing architectural concerns, that data collection and analysis is more extensive compared to traditional prioritization practices, but that extensive analysis seems inevitable in architecture improvement work.**

Keywords—software architecture design; architectural concern; concern prioritization; analytic-deliberative decision making.

I. INTRODUCTION

There is a delicate balance between documenting too much and documenting too little: over documenting leads to waste and missed windows of opportunity [1]; under documenting leads to chaos and project failure. High degrees of documentation have traditionally been associated with enterprise scale software development; low degrees of documentation with teamwork in the small.

Research in lean and agile software development has challenged this doctrine, but not yet got the balance right. Non-traditional team-formation, delegation, and coordination, e.g., streamline development [2], has reduced the need for documentation in large projects, but also invoked classical quality problems due to lack of architectural understanding and communication [3].

The purpose of this paper is to find methodology for prioritization of architectural documentation in enterprise scale projects with a need for architecture documentation evolution. By architectural documentation (AD), we mean a set of models, viewpoints, and views covering stakeholder concerns as defined by IEEE 1471/ISO 42010 [4]. By AD prioritization, we mean systematic selection of what stakeholder concerns AD should cover or not cover. By enterprise scale, we mean 300 designers or more under enterprise management. By evolution we mean corrective action in contrast to the planning of new projects.

Approaches to prioritization include methods with emphasis on *quantitative analytical techniques,* e.g., AHP [5], *qualitative analytical techniques,* e.g., grouping [6] and *collaboration* [7, 8]. Approaches differ with respect to granularity, ease of use, stakeholder involvement, scalability, duration and other factors [9]. Thus choice of prioritization method is a contingency problem: what is most appropriate in a specific case depends on the context.

We approach the problem of finding appropriate prioritization methodology through action research [10] in a large organization. Through iterative and incremental exploration of techniques plausible in the organization studied, and through reflection on our experiences, we build understanding on what characteristics a prioritization method should have to be effective in organizations of the studied kind.

Our result is a stakeholder-oriented method that combines analytical and collaborative techniques. Our method involves surveys to assess needs, graphical statistical techniques to present and validate needs, qualitative techniques to derive recommendations, and workshops for deliberation. Key analytical steps of the method are mapping architectural concerns to *priority zones*, identification of *improvements*, and *collaborative inference making*. By involving stakeholders in the activities where their engagement adds most value, and by attention to the specific frames of reference for each stakeholder group, a method that collects, abstracts, selects and conveys needs for improvement is obtained. The method is intended as decision support in architecture work *after identification* of architectural concerns and *before introduction or removal* of architectural views.

The paper exemplifies the method as it was applied in a pilot study at Ericsson: 29 senior designers and managers assessed the currents degree of documentation and their perceived need for change for 37 areas of concern identified in a previous study; responses were analyzed to obtain 5 areas with particular need for attention, 24 concrete suggestions of what to improve (directed to architects) and 8 action principles (directed to managers).

The remainder of the paper is organized as follows: we introduce concepts used in the study (Sec. II) and our research methods (Sec. III); we summarize research on which the method builds (Sec. IV), present the method itself (Sec. V), our results from applying it (Sec VI), and feedback from the unit (Sec VII). The paper ends with discussions of related work (Sec. VIII), future work (Sec. IX) and our conclusions (Sec. X).

978-0-7695-4351-2/11 $26.00 © 2011 IEEE
DOI 10.1109/WICSA.2011.13

II. Conceptual Framework

A conceptual framework identifying phenomena underlying our method is given in Fig. 1. Our primary entity of interest is the AD which is used by managers, architects, and designers, but for different purposes. The AD provides a set of views that may or may not cover architectural concerns important to its users. Architects have the technical power to define what views the AD should contain. Managers regulate the ways of working and have the economical and organizational power to evolve the AD and associated ways of working.

III. Research Methods

A. Research Strategy

Our choice of action research is due to the organizational context of our research. Engagement of the studied unit is conditioned on continuous alignment with local improvement goals and impact before theory building. The context is complex, and learning about it part of the research. The action research methodology is designed to support improvement under such conditions [10] and has proved itself efficient for software architecture improvement [11]. In the unit, action research is a preferred strategy with a good track record [12].

B. Research Site

The unit subject to improvement is Ericsson's research and development (R&D) organization. The unit hosts several projects that develop components for mobile communication networks, e.g., base stations, message routers, and various support nodes. (See e.g. Kaaranen for an overview [13].) Over the last decade, many of these projects have implemented model based software engineering and model driven development. Model based practices have gradually replaced document based practices which means that AD in the considered projects typically constitute blends of informal documentation, informal models, and formal models.

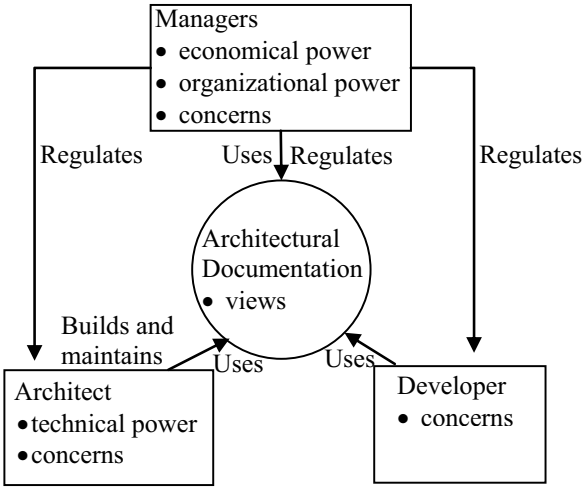

Figure 1. Roles and phenomena important in concern prioritization

In 2008, one project experienced consistency problems due to lack of architectural views at the subsystem level. Although UML-based architecture- and implementation models were used, these were not sufficiently connected but posed an abstraction gap left to designers to bridge in ad hoc ways; this resulted in consistency problems and unpredictability. A parallel project with longstanding documentation challenges [14] reported on similar experiences. The shared experience of these two projects led to research on *concern coverage* [15, 16].

In 2009, a third project implemented streamline development [2]—an in-house developed process for lean and agile software development in the large. Despite major modernizations to design documentation practices, this project experienced that "lack of a clear architecture" added significant lead time to projects, in particular for legacy systems. Methods for prioritizing architectural concerns—with the objective of removing superfluous ones—was seen as a promising complement to streamline development.

C. Action Research Cycles

So far, four action research cycles concerned with concern prioritization, preceded by previous research on concern inventorying, have been carried out. (See Fig. 2.) Each cycle followed the canoncial steps of action research described by Dickens [17]: diagnosing, planning acting, and reflection. Two cycles ($Cycle_2$ and $Cycle_3$) occurred in parallel, as two researchers focussed on different questions during the period. The cycles are outlined in Table I.

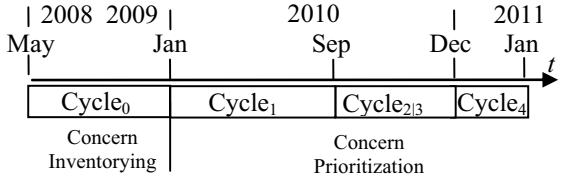

Figure 2. Action research cycles

TABLE I. Action research steps in each cycle

Step	**$Cycle_0$ – Concern Inventorying**
Diagnosing	Need to add missing views to AD.
Planning	Planned interview study to compare "mental views" among key stakeholders with the architectural views provided by current AD.
Acting	Interviewed 16 stakeholders (architects, designers, testers) about missing views. Identified 142 concerns important to stakeholders. Structured these in a taxonomy with 8 main areas and 37 sub areas. Assessed coverage of current AD, and found it to be 26% of the concerns identified.
Reflecting	The set of concerns important to designers is vast and many faceted. What concerns AD should cover is controversial. Some roles suffer from under-documented concerns, while other produce documentation not used. Systematic viewpoint engineering (to identify, prioritize, and realize views for concerns) is needed. What business drivers that should drive prioritization are unclear: is it quality, cost, efficiency, automation, maintainability or some weighted combination of indicators?
Step	**$Cycle_1$ – Concern Prioritization**
Diagnosing	Need to understand what concerns to increase and decrease documentation for, to improve speed.

Planning	Planned and piloted a survey study to assess current AD-coverage of stakeholder concerns and whether concerns are perceived as under- or over- documented, given that the success indicator is speed. The pilot study revealed several challenges in survey design: in limiting concern descriptions to "a few pages"; in keeping questions few and simple; in probing for concrete suggestions.
Acting	Executed survey. Invited 50 senior R&D engineers (architects, designers, line managers, project managers, and process engineers), out of which 29 responded: 68% from a >1000 p. base station software project; 12% and 4% from >100 p. telecom supports node projects, 12% and 4% from corporate concern and other functions. The survey was announced as difficult, but rewarding; respondents reported spending ½ a day on it. Analyzed survey answers using graphical statistical techniques. Studied change forces across projects only to find occasional differences. Inferred 37 hypothetical inferences from statistics. Presented statistics and inferences in a neutral way to respondents for feedback in a ½ day workshop. Documented responses in the form of reflections and observations. In the workshop, 35 out of 37 inferences were approved (after minor changes).
Reflecting	The survey and the workshop worked well in triggering reflections on current ways of working, in eliciting improvement ideas, and in prioritizing areas in need for improvement. To the dissatisfaction of some informants, the survey did not work so well in recognizing cases of over documentation: in the statistics, cases of over-documentation drowned in a generally positive attitude towards increasing documentation for increased speed. We realized that our term "increase documentation" captured all kinds of changes to design documentation: improved documentation, different documentation, and even removal of redundancy from documentation. A surprising observation during data collection was hesitance among architects to take the survey: while designers contributed rich amounts of data, several architects found questions "difficult to make anything meaningful out of". In the end, also the hesitant architects expressed interests in participating in analysis and deliberation activities, thus partly acknowledged the approach. A possible explanation is that of Farenhorst et al. [18]: architects, in general, do not like the act of codification of architectural knowledge, while codified knowledge ranks #1 on their wish lists.
Step	**$Cycle_2$ – Concern Prioritization**
Diagnosing	Need to better understand what kinds of increases/decreases and other changes the informants had in mind. Need for a theory to analyze, compare, and discuss different kinds of changes to AD.
Planning	Planned a qualitative analysis of the answers to the open survey questions. These had been included to enquiry concrete improvement suggestions, but answers now stood out as a rich data source for theory building.
Acting	Coded and categorized survey answers to arrive at 144 concrete suggestions on how AD should be changed to increase speed. Structured suggested changes in a taxonomy recognizing 13 kinds of increases and 9 kinds of decreases. Observed that increase categories and decrease categories were different, and that taking averages of increase and decrease forces is misleading. Updated statistics to distinguish increases from decreases accordingly.
Reflecting	Increase- and decrease-forces among informants are of different nature, and should not be set against each other. Concern prioritization is an act without firm boundaries: one may bring out concerns in most (and least) need of attention, but to define precise thresholds of what to attend or not attend to is beyond what can be feasibly modeled in this context. The techniques employed are intuitive, do not require special training in statistics, and should be amenable to most managers.
Step	**$Cycle_3$ – Concern Prioritization**
Diagnosing	Need to cluster concerns in groups to make results communicable to upper managers.
Planning	Planned a qualitative study of commonalities of concern areas subject to high (or low) change forces.
Acting	Explored several notions of zones for grouping concern areas in statistical plots. Developed a zone model recognizing both high and low forces to increase or decrease documentation. Defined a valuation function in two variables, and used it to define zones. Mapped concern areas to zones. Iteratively validated zone mappings with insiders. Adjusted zones until a suitably large list of prioritized areas was found.
Reflecting	For results to lead to action, desired changes to AD need to be articulated at "executive level": options should be few, concise, free from technical detail, and factual.
Step	**$Cycle_4$ – Concern Prioritization**
Diagnosing	Need to explain prioritization results to upper managers at a higher level of abstraction to make impact.
Planning	Identify variables that managers have intuition for; show relationships between these and desired improvements.
Acting	Hypothesized 5 variables plausibly correlated to prioritized areas. Assessed variables using a survey sent to two key informants. Studied correlation between these variables and forces to increase or decrease documentation; found 3 out of 5 variables to be correlated. Formulated action principles on the basis of findings.
Reflecting	To be effective, all prioritization activities and prioritization results need to be stakeholder oriented. To *engage* designers, architects, and managers in improvement work, we need to express ourselves using different forms: designers are comfortable with domain vocabulary and statistical plots; architects expect and appreciate highly structured, refined, and traceable knowledge; upper managers are more likely to react to knowledge presented in the form of action principles. Engagement of all roles is needed: designers to give input on the needs; architects to select and realize or remove views; managers to make (the right) economic decisions and to approve change projects.

IV. CONCERN INVENTORYING

Our method for concern inventorying is outlined in Fig. 3. The method is an abridgement of the actions in $Cycle_0$, with the windiness associated with exploratory action removed.

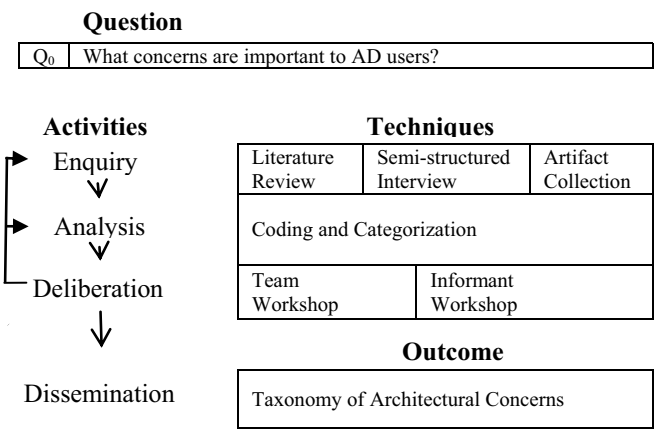

Figure 3. Concern inventorying method

Concern inventorying addresses the fundamental questions of what architectural concerns are important to the users of AD (Q_0); it proceeds by inquiry, analysis, deliberation and dissemination. What techniques to use for inventorying (literature review, etc.) to which extent, depends on qualities of the available taxonomies.

A. Literature Review

The inventorying effort may be reduced by adaptation of existing taxonomy [16, 19-21]. Qualities to consider when selecting taxonomy are *domain* (business domain, product type), *organization assumptions* (software processes, tools used), *granularity* (level of details in the breakdown, level of detail in definitions), and *scope*—does it cover areas outside the area immediately "under the technology lamppost" [22]. The more representative, refined, and concrete, the more value the taxonomy will bring.

B. Semistructured Interviews

Existing concern taxonomies are useful starting points in concern inventorying, but not effective theories: concerns are situational, and important concerns are not necessarily covered by available taxonomies. Semi-structured interviews (i.e., inquiry about needs around open-ended questions) help discovering and articulating new concerns.

C. Coding and Categorization

Concern taxonomy is beneficially constructed with a qualitative data analysis tool, a requirements engineering tool, or some other tool that supports text analysis, conceptualization, categorization, drill down into data sources, and report generation. Inductive coding of interview transcripts is used to reveal concerns that may otherwise be overlooked. Deductive coding is used to see whether concerns of existing taxonomies are present or not. Categorization is used to group concern into areas of concern, and major areas of concern.

The taxonomy resulting from our concern inventorying [15, 16, 23] is outlined Fig. 4: the outer circle lists the areas of concern; the inner circle the major areas.

D. Informant and Team Workshops

The importance of recurrently feeding analysis results back to the organization cannot be overstated. It reveals new areas and data sources previously overlooked, it aligns terminology with vocabulary meaningful inside the organization, and validates (or refutes) interpretations. As action research teams involve insiders, team workshops around selected themes become effective means of deliberation. Even more effective are workshops with informants, but at higher costs (and increasing impatience among informant), thus should be used sparingly.

V. Concern Prioritization

Fig. 5 outlines our prioritization method, which is an abridgment of our action research cycles 1–4. Prioritization involves focus on four questions ($Q_{1\text{-}4}$) that each reflects a diagnosis step of $Cycle_{1\text{-}4}$; questions are addressed by iteration over four activities (enquiry, analysis, interpretation, and deliberation), and by employment of various techniques (concern coverage survey, etc.) depending on the question addressed and the nature of the activity.

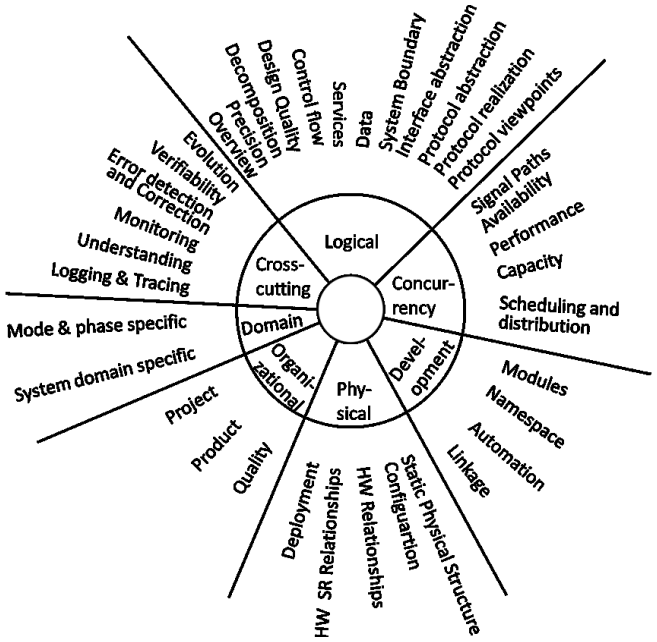

Figure 4. Taxonomy resulting from concern inventorying

Questions

Q_1	How strong are the forces among AD users to increase/decrease documentation in each area of concern?
Q_2	What areas should we recommend managers to improve?
Q_3	What improvements should we recommend for these areas?
Q_4	How should we communicate recommendations to managers?

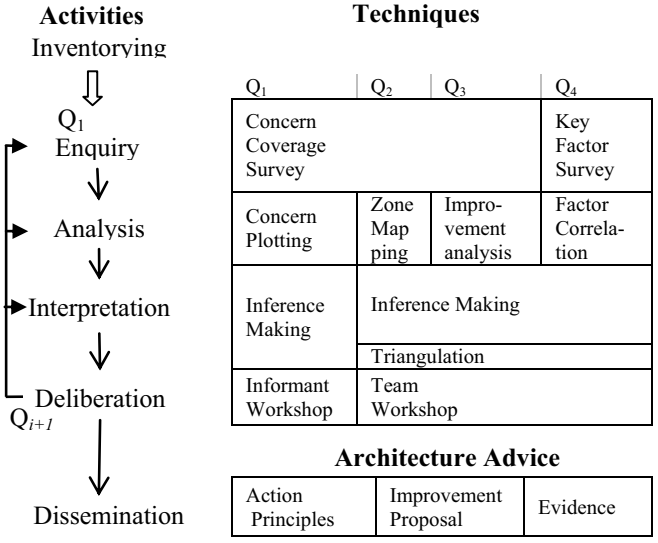

Figure 5. Our concern prioritization method: four questions (Q_i) are iteratively adressed using 5 generic activities (to the left) and 11 specific techniques (the upper boxes) to yield architecture advice (the lower boxes) directed towards to managers and architects

The method presupposes that concerns inventorying has been carried out (to answer Q_0), and ends with dissemination of architecture advice (*action principles*, *improvement proposal*, and *evidence*) among architects and managers.

The first iteration focuses on an initial question (Q_1), whereas following iterations shift focus to new questions (Q_2, Q_3, and Q_4) while refining answers to previous questions as knowledge grows. In addition to the focus questions, subsequent iterations also address *raised questions* (r_i) which are questions raised during deliberation. Sub-iterations for the raised questions, without change of focus, are introduced as needed if time so allows. Iteration continues until all questions have been answered to the degree of confidence required by the organization.

Each activity employs techniques suitable for the questions investigated. Techniques involve *surveys* (concern coverage survey, key factor survey), *statistical techniques* (concern plotting, zone mapping, factor correlation), *qualitative techniques* (improvement analysis, inference making, triangulation), and *collaborative techniques* (informant workshop, team workshop).

A. Concern Coverage Survey

Concern coverage survey assesses current degrees of documentation and change forces through surveys sent to designers. Surveys give sufficiently many samples to allow statistical techniques, and are inexpensive (compared to interviews). Limiting enquiry to designers is motivated by our experiences: in our survey, designers excelled on response rates, data quality, understanding of concerns, and concern coverage; architects and managers occasionally exhibited the same degree of concern awareness, but were on the large whole, less aware of needs; needs brought forward by managers and architects were also brought forward by designers, and collective forces to increase or decrease documentation did not vary across roles.

The questions to ask for each area A, for the overall improvement goal G, and the associated input instrument for each question are

1) *How would you classify the present support in area A?* (Not documented | weakly documented | scantily documented | well documented | strongly documented | don't understand)

2) *How should support change if G is to be improved?* (Essential to decrease | desirable to decrease | slightly advantageous to decrease | slightly advantageous to increase | desirable to increase | essential to increase | no intuition)

3) *What kind of increases/decreases of information would you like to see in area A?* (Free text field.)

To stimulate further reflection the survey should also ask:

- *Further comments of the major area M as a whole*? (Free text field.)
- *Additional reflections on concern coverage regarding what is done well, and urgent improvement needs*? (Free text fields.)

To allow differentiation among informant groups in the analysis, questions about *role*, *product family,* and *project experiences should* also be asked.

The survey should be accompanied by a *survey guide* that, in a concise way, explains the purpose of the survey, important concepts, and the concern taxonomy.

B. Concern Plotting

Concern plotting is a potentially automatable technique that maps survey answers to a diagram with the current situation along one axis, and the desired change along the other. (See Fig. 6.) Survey answers are weighted (1 for slightly advantageous, 2 for desirable, and 4 four essential to increase, and corresponding negative weights for decreases), and average *increase-forces* and *decrease-forces* computed from the weighted answers. Change forces are then plotted against the current situation. Fig. 6 shows such a plot for the major area of logical concerns.

Here, points to the left reflect decrease-forces and points to the right increase-forces. We can observe a strong force to increase *overview documentation* (the rightmost dot), and an opportunity for increasing documentation of *data*-related logical concerns (as the force is strong and the area is under-documented). Control flow and interface abstraction seem over-documented and likely to benefit of a decrease of documentation. Concern plots of this kind are produced for each major area of concern.

C. Inference Making (Q_1)

Inference making is a technique to turn *data displays* (concern plots and other statistical displays) and *data sources* (qualitative survey data and notes) into verbal hypotheses about the current situation. It proceeds by collaborative hypothesis generation around the data displays, followed by evidence building on the basis of the data displays and the data sources. Useful data displays include pie-charts, bar-charts, and histograms showing

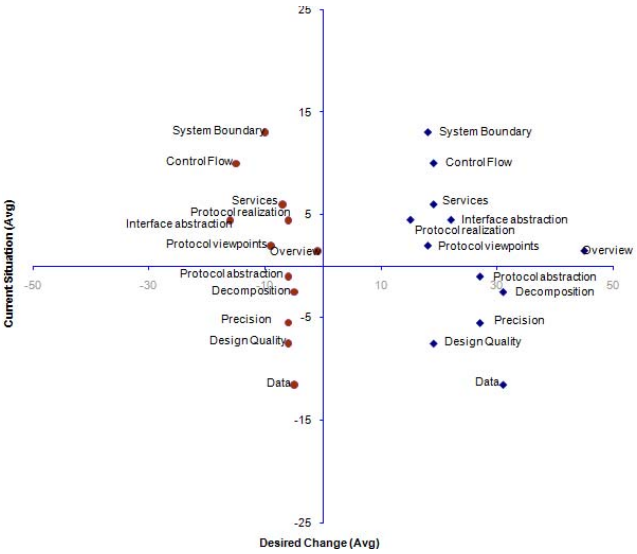

Figure 6. Concern plot for the major area of logical concerns

- understanding of concern areas among informants,
- non-understanding of concerns per role,
- desires for change across roles,
- distribution of answers for each area of concern,
- distribution of informant roles, and
- population of analysis categories.

An example hypothesis with evidence is given in Fig. 7. Examples of other hypotheses, inferred using this technique, and later validated by the informants, are given in Table II.

D. Informant Workshops

The informant workshop is a collaborative deliberation technique used to validate the statistical displays, validate inferences made from these, and to generate further hypotheses. An informant workshop has two sessions. In the *interpretation session*, the statistical displays (distributed to informants in advance) are presented in an unbiased way, and discussed by the informants, while analysts take notes; analysts should be careful, during this session, not to inject opinion. In the *validation session*, analysts present their own hypotheses and their evidence while informants collectively comment, criticize, approve, or refute the hypotheses or the evidence. Notes-taking, preferably by several analysts, is used to document the response. By encouraging informants to make their own interpretations before presenting hypotheses, misinterpretations are more likely to be discovered.

Examples of hypotheses confirmed in our informant workshop are given in Table II; examples of questioned hypotheses are:

~~H_3~~: In nearly all logical concern areas we want to increase documentation.

~~H_{18}~~: We are very pleased with the documentation of namespace concerns.

~~H_{34}~~: To increase speed, we should increase documentation in virtually all concerns areas.

One hypothesis (~~H_{18}~~) was found overly strong with too sweeping argumentation; one (~~H_3~~) was found surprising, and one (~~H_{34}~~) to have unconvincing argumentation. In all, 33 hypotheses were validated (sometimes with minor corrections).

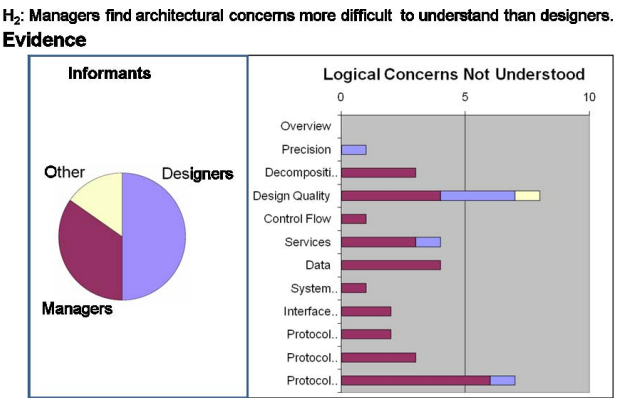

Figure 7. Example Hypothesis and its Evidence

TABLE II. HYPOTHESES RESULTING FROM INFERENCE MAKING

H_1: Some senior managers would benefit from education about architectural concerns important to designers and managers.
H_2: Managers find architectural concerns more difficult to understand than designers.
H_4: We are in great need of better overview documentation.
H_5: Data is weakly documented and needs increased documentation.
H_7: In some areas of logical concerns, we are very pleased with the level of documentation.
H_9: Design quality is the logical concern area least understood among informants.
H_{10}: Role does not impact views on needs.
H_{11}: Project home base does not significantly impact views on needs.
H_{32}: Ranking from best to worst degree of documentation is organizational-, physical-, logical-, development-, crosscutting-, domain, and concurrency concerns.
H_{33}: We are happy with what's well documented, whereas the moderately/weakly documented needs increase.
H_{35}: No weak links in documentation is needed to increase speed. ("In the sense of traceability across documents.")
H_{36}: Concerns without an established modeling language tend to be less documented than those we know how to document.
H_{37}: Knowledge transfers from designers to managers, or empowered designers, are needed to increase speed.

E. Zone Mapping

Zone mapping is a potentially automatable technique to identify concern areas in most need for improvement. We recognize three basic zones (see Fig. 8.): one for areas with an appropriate degree of documentation (OK zone); one for areas deserving increased documentation (INC Zone), and another for areas deserving decreased documentation (DEC Zone). The increase- and decrease zones further contain sub-zones recognizing priority (INC++ $\subseteq$ INC+ $\subseteq$ INC and DEC-- $\subseteq$ DEC- $\subseteq$ DEC).

Notice the S-shape of the OK zone, which is to compensate for a psychological effect observed in our study: concerns recognized as important by an organization tend to call for increases even though their documentation is actually already high and of good quality. By attending primarily to weakly or moderately documented concerns, we compensate for this effect, and direct resources to areas where they will have most effect.

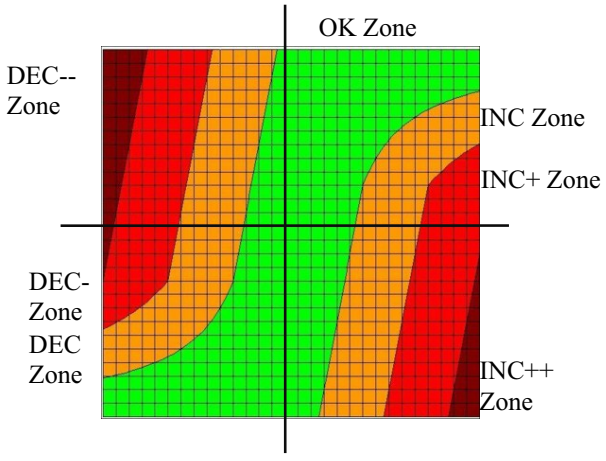

Figure 8. Example of a zone map

Also notice that zones are offset slightly to the right. This is to compensate for another observed effect: recognized opportunities to increase documentation tend to outnumber recognized opportunities to decrease documentation, which motivates a "handicap" for concerns in need for decrease.

Zones are defined by a function in two variables that is continuous, and monotonic, and that gives sensible weights to the two variables. By adjusting the surface of this function (which we have expressed using a matrix), zone boundaries may be moved to include additional or fewer concerns, in order to obtain a number suitable for recommendation.

Zone mapping is in itself straightforward. We give a concern area *high attention* if it falls either in an INC+(+) or DEC-(-) zone, and *moderate attention* if it falls either in and INC or DEC zone, and no attention if it falls within an OK zone. The result of our zone mapping for the logical concerns given in Table III.

The outcome is that high attention should be given to increasing documentation of overview and data concerns, moderate attention to increasing documentation of interface abstraction, protocol abstraction, decomposition, design quality, and protocol viewpoints and moderate attention to decreasing documentation of system boundary control flow, and interface abstraction concerns. No attention is needed for documentation of services, protocol realization and protocol viewpoints. That the documentation of an area is important to both increase and decrease may appear surprising at first, but plausible when proposed improvements are considered.

F. *Improvement Analysis*

Improvement analysis supplements zone mapping with information on what to improve in prioritized areas. It proceeds by coding and categorization of survey data (cf., concern inventorying), and results in suggested improvements for each area of concern along with improvement categories. Table IV lists the improvements proposed for prioritized areas among all major areas; Fig. 9 shows the categories of improvements identified in our case, and the number of proposed improvements in each category.

Comparing the kinds of suggested increases and decreases in Fig. 9, we observe little tension: decreases are concerned with less organizational formality, fewer details in views, removal of unused views; increases with additional, improved, or more widely used views, with more educational materials, and enhanced infrastructure properties.

TABLE III. ATTENTION TO LOGICAL CONCERNS GIVEN BY ZONE MAP

Logical Concern Areas	Dec	Inc	Attention
System Boundary	DEC	OK	DEC
Control Flow	DEC	OK	DEC
Services	OK	OK	OK
Interface abstraction	DEC	INC	DEC/INC
Protocol realization	OK	OK	OK
Protocol viewpoints	OK	OK	OK
Overview	OK	INC+	INC+
Protocol abstraction	OK	INC	INC
Decomposition	OK	INC	INC
Design quality	OK	INC	INC
Data	OK	INC+	INC+

TABLE IV. SUGGESTED IPROVEMENTS FOR PRIORITIZED AREAS

Area	Suggested Improvements
Overview (Logical concerns)	1. Architectural principle motivations. 2. More abstract views. 3. High level architecture overview. 4. Partitioning principles in different dimensions. 5. A Clear overview. 6. Architectural description HowTo's.
Data (Logical concerns)	1. Information on data. 2. Visibility of data propagation. 3. Data dependency diagrams.
Linkage (Development concerns)	1. Traceability. 2. Automated traceability. 3. Design rules for build process. 4. Breakdown of requirements and traceability between black-box and white-box perspectives. 5. Relation between realization of managed objects and resource objects. 6. Dependencies and connections between data and protocols. 7. Traceable interaction from managed objects, to resource objects, to other parts.
Mode and phase specific concerns (Domain concerns)	1. Guiding principles for mode and phase specific concerns. 2. Overview documentation for modes and phases covering multiple system levels. 3. Improved description of update & upgrade procedures. 4. Distinct separation of mode and phase specific concerns. 5. Informal documentation on mode and phase specific concerns when reasoning between disciplines. 6. Mode and phase specific overview information. 7. View of system restart. 8. Increased precision in description of mode and phase specific concerns.

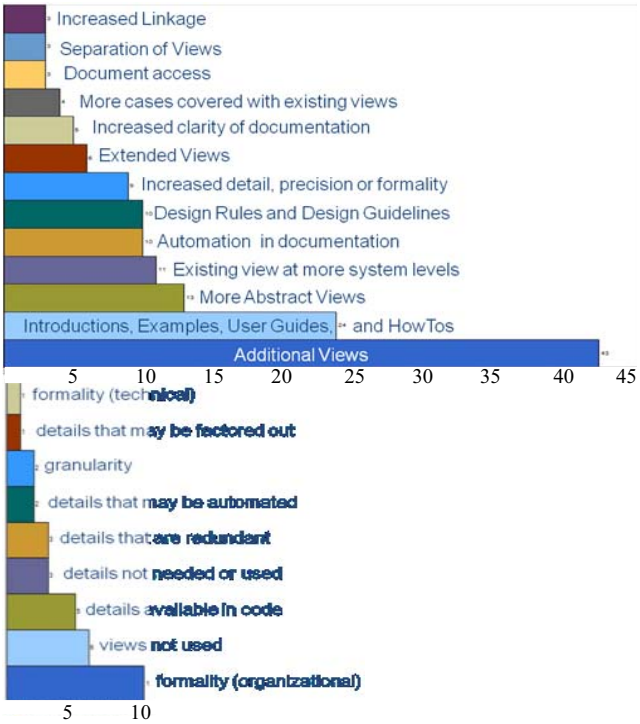

Figure 9. Improvement analysis outcome: kinds of proposed increases (upper graph) and decreases (lower graph) of architectural documentation

G. Triangulation

Triangulation, in our method, is an implicit technique, realized by analysts working with multiple data sources and methods: when inferences made during one iteration contradict those made in another, analysts are forced to revisit the supporting data to explain the contradiction.

Triangulation led to rejection of ~~H_3~~ and ~~H_{34}~~, as two data sources (informant feedback, and improvement analysis) spoke against their truth; triangulation also led to rejection of hypothetical action principles with too weak support in data.

H. Key Factor Survey

Key factor survey is a technique to identify and assess factors that managers can relate to, and that are empowered and motivated to handle. Without this activity, architecture advice—the outcome of our prioritization method—risks being perceived as "just another good idea" [24].

A key factor survey involves survey design and survey execution. Survey design proceeds by collaborative sessions around data displays and results in identification of key factors and scales for assessing these. Survey design should preferably involve software process improvement specialists and managers knowledgeable with improvement work within the organization. The focus question is how architecture improvement advice is best presented to management. Examples of factors found in this way are:

- MS: management support for documenting an area;
- HER: heritage of documentation practices;
- HW similarity: overlap with HW documentation practice;
- SIM: simplicity of the concern area, i.e., whether it is simple or difficult to deal with regardless of tool support;
- ToS: what UML and other standards support.

Reasoning with these factors is commonplace within the organization studied, thus expressing advice around these, improves our chances of impact. An example scale, used to assess the factor MS is

MS− : Most managers are probably not aware of this area, or its importance in SW development.
MS0 : Concerns in this area do not get much attention from management. They are known by management, to some degree, but not considered important enough to spend money or time on.
MS+: Some concerns in this area has (had) high attention and support from management, e.g., occurs in checklists, occurs in priority lists, or is addressed by internal improvement activities.

Survey execution simply means to ask key stakeholders to assess the identified variables using scales of this kind. In our study, assessment of the factor MS for the area of logical concerns resulted in assignment of MS− (i.e. unawareness) to the areas *design quality* and *data*, and assignment of MS0 (i.e., non-attention) to other areas; no logical concern area was assigned high attention (MS+), whereas 5 areas in other main areas were assigned high attention (3 organizational, 2 development, and 1 crosscutting concern areas).

I. Factor Correlation

Factor correlation is a technique to study relationships between key factors and zone mapping outcomes, i.e., how proposed improvements relate to the management domain. The purpose is to provide evidence for the action principles.

TABLE V. MANAGEMENT SUPPORT × IMPROVEMENT NEEDS

	DEC-	DEC	OK	INC	INC+
MS-	0	0	0	3	1
MS0	0	4	4	15	3
MS+	0	3	1	1	0

Factor correlation proceeds by cross comparison of a factor with desired changes. For instance, the relationship between management support (MS) and desired changes (DEC-. etc.) is given in Table V. Here, we see that areas important or desirable to increase are associated with non-attention (15+3) or lack of knowledge (3+1). We see that 3 out of 5 (60%) of the logical concerns areas that get high management attention are over-documented. Incidentally, a standard correlation test on these data showed a significant negative correlation ($\rho = -0.46$) between management support and the desire to increase documentation. This suggest that efforts in moving management's attention away from what the organization is focusing on today, to make the organization actively look for concerns overlooked, would be an effective strategy in improving AD.

J. Inference Making (Q_4)

Inference making is essentially the same process in all iterations, but the data sources and the level of abstraction at which claims are produced differ. In the inference making for the Q_4 iteration, primary data sources are the factor correlation matrices. One example of an inference made in this activity is the action principle *educate managers* (see Table VI), which is supported by the cell ⟨MS0, INC⟩ and its neighbours in Table V as well as hypothesis H_2 in Table II.

VI. CONCERN PRIORITIZATION OUTCOME

A. Action Principles

The outcome of our method resulted in the action principles in Table VI, which are inferred from our data sources, supported by traceable inference making, and validated in a team workshop.

TABLE VI. ACTION PRINCIPLES

Spend time on prioritizing concerns. Concerns are not equally important.
Educate managers about architectural concerns. Avoid unconscious prioritizations due to lack of knowledge.
Reach beyond managers' needs. Don't over-document in areas supporting managers' interests.
Challenge the heritage. The way you documented in the past is not the best way now.
Avoid the simplicity trap. Don't prioritize concern areas just because they are simple.
Move beyond out-of-the-box modeling. Don't be content with weak notations: utilize tools and extend.
Encourage designers to share knowledge. Designers know the needs: make sure to learn and share their views.
Encourage architects to learn and share. Architects must be willing to react on architecture improvement advice.

TABLE VII. PRIORITIZED AREAS OF IMPROVEMENTS

	Decrease	Increase
Important	*None*	Overview, data, linkage, mode and phase specific concerns
Desirable	System boundary, control flow, interface abstraction, namespace, static physical structure, H/W relationships, product, project.	Interface abstraction, protocol abstraction, decomposition, design quality, scheduling & distribution, capacity, availability, performance, automation, configuration, H/W S/W relationships, deployment, quality, system domain specific, error detection & correction, logging & tracing, monitoring, evolution, understanding, verifiability.

B. *Improvement Proposal*

An improvement proposal consists of a list of concrete proposals of what should be changed in each prioritized area. For instance, our advice to the unit was to increase and decrease documentation according to table VII.

C. *Evidence*

Evidence for the action principles and the improvement proposal is built from the *hypotheses* and *processed data* (statistical plots, taxonomies, and tables) which in turn are built from *raw data* (survey answers, workshop notes, and transcripts). The evidence data of our report amounts to roughly 100 pages (on electronic form), whereas raw data amounts to 300 pages.

VII. FEEDBACK FROM THE UNIT

The unit has been extensively involved during data collection and in all workshops. Feedback on the survey was that it was difficult, but that it lead to reflection on (and questioning of) current ways of working. Feedback during the informant workshop was that the concerns catalogue was a valuable checklist that would benefit from company-wide standardization, and that concern plots captured the needs well and were good for collective discussions of needs.

Feedback on the method, from a senior specialist in software process improvement, was that data collection and analysis is more extensive compared to traditional prioritization practices, but that extensive analysis seems inevitable in architecture improvement work.

Feedback on the action principles was that although it is clear that more insight in some concern areas is needed, it is also clear that managers should not have to understand concerns to the same degree as designers do.

VIII. RELATED WORK

Most works on prioritization in software engineering are concerned with product- or software requirements within a product development process [6, 25-27]. Concern prioritization is different to classical requirements prioritization, because AD is not engineered in the same way as products are: processes for AD evolution are typically implicit; there is no organized elicitation of stakeholder needs, and decision procedures and economical frames are not as explicit. Requirement prioritization techniques address some challenges of concern prioritization, but do not provide a whole solution.

Clements et al. presented a method with explicit focus on prioritization of views [19]; their method involves a catalogue of architectural views needed by stakeholders, stakeholder workshops, internalization of informant needs by the architect, decision making by the architect, and feedback on decision making from stakeholders. Our approach differs in its emphasis on analytical techniques, evidence building, surveys, and involvement of managers in architecture improvement work. Where Clements et al emphasize prioritization of concerns in the AD construction phases, our method emphasizes evolution of existing AD.

Fairbank [28] proposes a risk driven approach to architecture, where risk perceived by designers is used to prioritize architecture work. Our method differs in its focus on organization of prioritization work, and is goal agnostic.

Closest to our approach, is the distributed prioritization method of Regnell et al. [8], targeted at market driven product development. This method is iterative, driven by a small team that enquires disparate social worlds using survey's, analyses the feedback using graphical statistical techniques, involves informant feedback, and arrives at high level requirements. It does however rely on internalization of the prioritization problem by the deciding managers, whereas, in our method, a team of analysts provide managers with decision support.

IX. FUTURE WORK

At the time of writing, findings from the study have led to implementation of model-based support for interface specifications (interface abstraction) and new design rules (design quality). Decisions have partly been based on the architecture advice and partly on factors not present in the prioritization. Future work involves comparing advice fed to the organization with actual evolution and to identify factors with influence on the decisions.

Studies of the psychology behind preferences towards improvement by adding (rather than improvement by removal) and behind wanting more of what is there, would allow for more precise surfaces of zone functions. Another interesting phenomenon is why change sometimes does not happen even when it is important.

On the technical side, algebraic formulation of the zone function would allow automated ranking and interactive adjustment of zone boundaries. Templates for inference making in various situations would make the approach simpler to apply.

Extending the method to involve also external stakeholders in AD (such as regulatory/standard certification agents and integrators), brings further challenges that may require additional activities for enquiry and deliberation. It seems plausible that the method could be applied to prioritization problems in other domains that software architecture.

X. SUMMARY AND CONCLUSIONS

Prioritizing architecture documentation requires engagement at several levels of the organization: designers must articulate their concerns; architects must build and maintain views; managers must provide economical resources and appropriate organization. Core challenges in this are to elicit concerns, engage stakeholders, communicate needs at appropriate levels of abstraction, and to make inter-role communication effective.

We have presented a new method for concern prioritization for the purpose of architecture documentation evolution. The method combines collaborative and analytical techniques, and involves multiple stakeholder groups to produce grounded advice on which areas of architectural concerns to improve, and what to improve in these areas.

The method is the result of three years of action research in a division of Ericsson developing base station software. The method has been developed to meet needs as they arise in a real situation; the outcome of the method has been incrementally validated throughout its development.

ACKNOWLEDGMENT

Research supported by Ericsson Software Research through Ericsson's Software Architecture and Quality Centre (SAQC).

REFERENCES

[1] M. Poppendieck and T. Poppendieck, Lean Software Development: an Agile Toolkit. Boston: Addison-Wesley, 2003.

[2] P. Tomaszewski, P. Berander, and L. O. Damm, "From traditional to streamline development - opportunities and challenges," Software Process: Improvement and Practice, vol. 13, pp. 195-212, 2008.

[3] K. Petersen and C. Wohlin, "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case," The Journal of Systems and Software vol. 82 pp. 1479–1490, 2009.

[4] "ANSI/IEEE Std 1471::ISO/IEC 42010-Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems," 2007.

[5] T. L. Saaty, "How to make a decision: The analytic hierarchy process," European Journal of Operational Research, vol. 48, pp. 9-26, 1990.

[6] P. Berander and A. Andrews, "Requirements Prioritization," in Engineering and managing software requirements, A. Aurum and C. Wohlin, Eds., ed Heidelberg: Springer-Verlag, 2005.

[7] D. M. Karydas and J. F. Gifun, "A method for the efficient prioritization of infrastructure renewal projects," Reliability Engineering & System Safety, vol. 91, pp. 84-99, 2006.

[8] B. Regnell, M. Höst, J. N. och Dag, P. Beremark, and T. Hjelm, "An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software," Requirements Engineering, vol. 6, pp. 51-62, 2001.

[9] P. Berander, "Evolving prioritization for software product management," PhD Thesis, Department of Systems and Software Engineering School of Engineering, Blekinge Institute of Technology, Sweden, Blekinge, 2007.

[10] G. I. Susman and R. D. Evered, "An Assessment of the Scientific Merits of Action Research," Administrative Science Quarterly, vol. 23, pp. 582-603, 1978.

[11] H. B. Christensen, K. M. Hansen, and K. R. Schougaard, "Ready! Set! Go! An action research agenda for software architecture research," Seventh Working IEEE/IFIP Conference on Software Architecture, 2008.

[12] A. Borjesson and L. Mathiassen, "Successful process implementation," IEEE *Software,* vol. 21, pp. 36-44, 2004.

[13] H. Kaaranen, UMTS Networks: Architecture, Mobility, and Services. . New York: Wiley, 2005.

[14] L. Pareto and U. Boquist, "A quality model for design documentation in model-centric projects," Proceedings of he 3rd international workshop on Software quality assurance, 2006.

[15] L. Pareto, P. Eriksson, and S. Ehnebom, "Concern visibility in base station development: an empirical investigation," 12th Intl. Conference on Mode Driven Engineering Languages and Systems (MODELS), 2009.

[16] L. Pareto, P. Eriksson, and S. Ehnebom, "Concern coverage in base station development - an Empirical investigation," Software and Systems Modeling, Online First, 2011.

[17] L. Dickens and K. Watkins, "Action Research: Rethinking Lewin," vol. 30, ed, 1999, pp. 127-140.

[18] R. Farenhorst, J. F. Hoorn, P. Lago, and H. van Vliet, "The Lonesome Architect" Joint IEEE/IFIP Working Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA), 2009.

[19] P. Clements, *et al.*, Documenting Software Architecture - Views and Beyond, Boston, MA: Addison Wesley, 2010.

[20] P. Kruchten, "Architectural Blueprints — The "4+1" View Model of Software Architecture. ," *IEEE Software* vol. 12, pp. 42–50, 1995.

[21] N. Rozanski and E. Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. . Upper Saddle River, New Jersey: Addison-Wesley, 2005.

[22] M. Medvidovic, E. M. Dashofy, and T. R.N., "Moving architectural description from under the technology lamppost," *Information and Software Technology 49,* vol. 12, 2007.

[23] L. Pareto, "Architectural concerns in base station development," Research Reports in Software Engineering and Management, 2010:1, University of Gothenburg, 2010.

[24] P. G. Armour, "Don't bring me a good idea," *Communications of the ACM,* vol. 54, pp. 27-29, 2011.

[25] P. Berander, "Evolving prioritization for software product management," Doctoral dissertation, Blekinge Institute of Technology, 2007.

[26] P. Chatzipetrou, L. Angelis, P. Rovegård, and C. Wohlin, "Prioritization of issues and requirements by cumulative voting: a compositional data analysis framework," 36th EUROMICRO Conference on Software Engineering and Advanced Applications, 2010

[27] B. Regnell, P. Beremark, and O. Eklundh, "A market-driven requirements engineering process: Results from an industrial process improvement programme," *Requirements Engineering,* vol. 3, pp. 121-129, 1998.

[28] G. Fairbanks, Just Enough Software Architecture - a Risk Driven Approach. Boulder, CO: Marshal and Brainerd, 2010