Alignment of Software Product Management and Software Architecture with Discussion Models

Garm Lucassen, Jan Martijn E. M. van der Werf and Sjaak Brinkkemper
Department of Information and Computing Sciences
Utrecht University
Princetonplein 5, 3584 CC Utrecht, The Netherlands
{g.lucassen, j.m.e.m.vanderwerf, s.brinkkemper}@uu.nl

***Abstract*—How to achieve alignment of software product management with software architecture and whether there is a business case for doing so is scientifically unknown. Yet, software architecture has large, direct impact on product success factors: creating a winning product and delivering value to customers. In this exploratory case study paper we identify the most critical processes for SPM and SA alignment: requirements gathering and refinement. These processes require effective communication supported by high level architectural views. Our respondents, however, rely on simple methods due to their negative experiences with formalized models. Based on these findings, we propose the Accurate Architectural Models Approach (AAMA) which prevents architectural model divergence.**

I. Introduction

Software product managers achieve product success by focusing on three goals: (1) creating a winning product and business case, (2) conquering markets and growing market share and (3) delivering value to customers [1]. An essential aspect to reaching these goals is requirements engineering (RE) [1]. Software architects have similar goals. They make architectural design decisions (ADDs) based on available requirements and previous ADDs [2] to satisfy stakeholders' functionality and quality attribute requirements [3]. Believing that an early software architecture (SA) understanding is as important to project success as requirements understanding, [4] adapted the *Twin Peaks Model* as introduced by Ward [5]. The Twin Peaks model supports the co-evolution of SA and RE by concurrently developing requirement details and architectural specification (Figure 1).

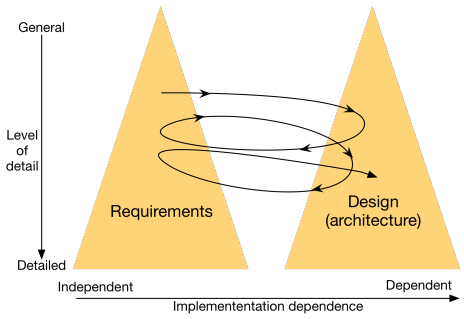

Fig. 1. The Twin Peaks of Requirements and Architecture [4]

The Twin Peaks model originates from research on software *projects*. However, the *product* software domain diverges from projects. As software producing organizations (SPOs) grow, they encounter large numbers and varieties of stakeholders that generate a continuous input of requirements [6]. Consequentially, decisions need to be made on which requirements to fulfill at what moment in time. To manage these inputs and decisions, SPOs require special processes [7], [8] that are typically the responsibility of *software product managers*. According to [9], product managers are crucial to product and company success and [1] even states that "companies win or fail depending on their product managers".

For SPOs, product managers are the primary interface and representative of stakeholders' requirements to the development team including the architect. Consequentially, we hypothesize that successful collaboration between software product management (SPM) and software architecture (SA) is essential for achieving both product success goal 1 and 3 and therefore product success. [10] supports this premise, noting that SPM alignment with SA is critical to software product line success. However, no publications to substantiate this claim are available. Before we can attempt to confirm the importance of SPM and SA interdependence, we first need to establish whether SPM and SA at contemporary SPOs collaborate in the first place. To this end, we pose the following research question: *"How do software product managers and software architects collaborate?"*.

This exploratory research discusses five case studies at Dutch SPOs to detail their SPM and SA processes and facilitating instrumentations. We identify two critical processes where product managers and architects exchange information: requirements gathering and requirements refinement. These activities require reciprocal information exchange on a high technical level to drive product success. Respondents, however, lack a structured approach that warrants SPM and SA alignment. We propose a solution that prevents architectural model divergence to improve SPM and SA alignment.

The subsequent sections II, III and IV discuss relevant SA and SPM literature. Next, we detail the research approach of this paper in section V. Section VI presents the case study results, followed by a discussion in VII and a proposed solution in VIII. We summarize our primary results and present future research opportunities in the concluding Section IX.

978-1-4799-6357-7/14/$31.00 © 2014 IEEE

II. Software Architect Processes

Software Architecture is "the set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both" [3]. Typical software architecture documentation consists of a collection of views, which are "a representation of a set of system elements and the relationships associated with them" [3]. An organization creates architecture documentation for multiple purposes. Organizations use it to design, analyze, communicate and/or educate a software system from multiple, varying stakeholder role perspectives. Considering that stakeholders can be as diverse as a customer and a developer, the seminal standard on Software Architecture (IEEE Standard 1471-2000) declares that architectural models are inherently multi-viewed [11].

A software architect has four primary tasks: developing project strategy, designing systems, communicating with stakeholders and being a leader [12]. Over time, the architect develops a software architecture by making architectural design decisions (ADDs) based on the project context which consists of stakeholder requirements and previous ADDs [2]. The ADDs in aggregate attempt to satisfy stakeholders' quality attribute requirements [3]. Quality attribute requirements specify system operation, contrary to functional requirements which specify system behavior. Their importance for product success is equal. After all, a system's ability to produce correct information is useless when the system is unable to deliver that information in time, securely or at all [3]. Achieving quality attribute criteria is central to the software architecture discipline. The ISO/IEC 25010 standard on system and software quality models posits that software product quality attributes comprise eight characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability [13]. Each characteristic contains a number of related sub-characteristics, which "are useful for specifying requirements, establishing measures, and performing quality evaluations" [13].

The ADDs a software architect makes are based on stakeholder requirements. Therefore, quality attributes are an indirect consequence of the project's set of requirements, creating a strong relationship between RE and SA. Drawing inspiration from Nuseibeh's Twin Peaks model for software development [4], several authors have formulated approaches to further relate software requirements and architecture [14]–[18]. However, none of these are widely adopted in business or academia. For example, the CBSP (Component-Bus-System-Property) approach by Grunbacher [15], which delivers a "proto-architecture" based on functional requirements to prescribe further architectural development, has been applied a limited number of times since its introduction [19], [20].

III. Software Product Processes

Software product management (SPM) is "the discipline governing a software product over its whole life cycle, from its inception to customer delivery, in order to generate the biggest possible value to the business" [22]. [23] defines a software product as "a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market". As software producing organizations (SPOs) grow, they encounter large numbers and varieties of stakeholders that generate a continuous input of requirements [6]. Due to limited resources, decisions need to be made on which requirements to fulfill at what moment in time.

To manage these inputs and decisions, SPOs require special processes [7], [8] that are typically the responsibility of *software product managers*. A product manager attempts to achieve product success by focusing on three goals: (1) creating a winning product and business case, (2) conquering markets and growing market share and (3) delivering value to customers [1]. In a case study examining what product managers should do to reach these goals, practitioners assert that "he does everything the product needs to be successful" [24]. Nevertheless, theory stipulates they manage requirements, produce release definitions and define products in an environment with many internal and external stakeholders [25], [26]. This wide range of responsibilities makes product managers crucial to product and company success [9], [27].

Although the number of publications on SPM is growing each year, the field is still young [28]. In 2011, [29] could find only 25 articles on the subject. Consequently, SPM scholars have difficulty to provide scientifically validated best practices to industry. However, several authors have developed SPM reference frameworks to exactly this end [21], [22], [25], [30]. Of these, the Software Product Management Competence Model (SPMCM) by [25] and extended by [21] is based on extensive empirical research. The SPMCM (Figure 2) presents the four main business functions within SPM: requirements management, release planning, product planning and portfolio management. Each business function consists of a number of focus areas that represent a strongly coherent group of capabilities. In theory, SPOs that hold these capabilities are more effective in their product management but this remains unproven to date.

[31] presents an approach to *agile* SPM, introducing product management sprints alternating with development sprints to ensure requirements' are ready for use when software development starts. During a product management sprint, the SPM team conducts *requirement refinement* for product backlog requirements from coarse-grained concepts to fine-grained instructions that software engineers use. Concretely, the product manager refines vision into specified requirements through three, progressively more fine-grained stages: themes, concepts and requirement definitions.

A. Requirements Management

Of the four main business functions in the SPM Competence Model, *requirements management* (RM) is the most frequent, lowest level function a product manager partakes in. Coincidentally, it is the most extensively studied domain in IS/IT research. [21] identifies three focus areas relevant in the product software context: (1) requirements gathering: "the acquisition of requirements from both internal and exter-

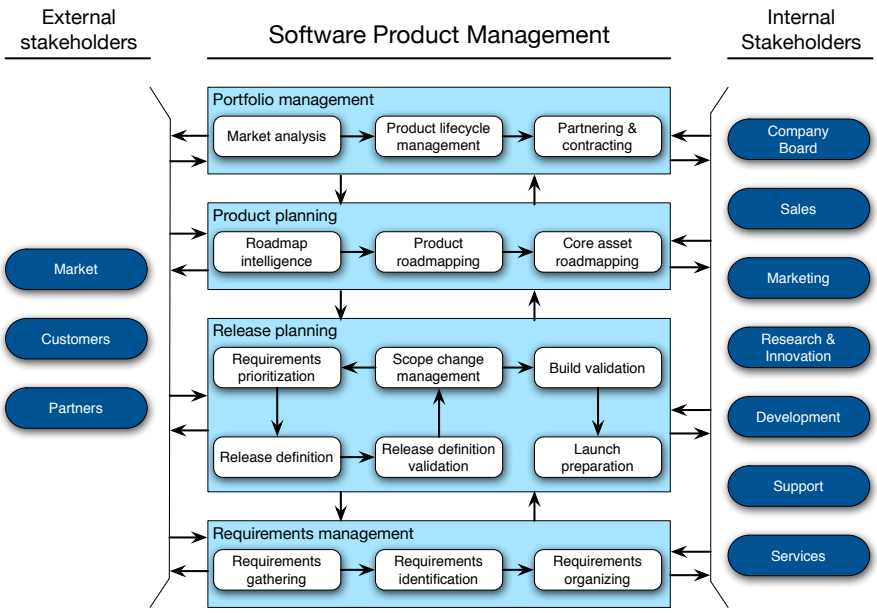

Fig. 2. Software Product Management Competence Model [21]

nal stakeholders", (2) requirements identification: "identifying the actual product requirements by rewriting the market requirements to understandable product requirements, and connecting requirements that describe similar functionality" and (3) requirements organizing: "structuring the requirements throughout their entire lifecycle based on shared aspects, and describing the dependencies between product requirements". These three activities combined with prioritizing requirements are traditionally called *requirements engineering* [32] and considered to be a core product manager responsibility [1], [26].

Requirements engineering typically occurs early in the lifetime of a project to prevent increased costs associated with requirements errors later on [33]. The product software context, however, presents specific challenges that necessitate a different approach to RE [34]–[36]. The large number of requirements makes RE a daily necessity over the lifespan of a software product. Furthermore, the large number of requirements frequently produces complications such as *information overload, combinatorial explosions* and *over-scoping* [7]. Several approaches to prioritize large numbers of requirements are available, both generic [37], [38], and context specific [39]. None of these are widely adopted by SPOs, despite providing concrete advantages over traditional prioritization techniques.

B. Release Planning, Product Planning, Portfolio Management

The emphasis of this research lies on requirements management, because it is the most frequent activity a product manager conducts. Therefore, this subsection details the remaining SPM activities only briefly. (1) *Release planning* comprises activities for creating and launching a release [21]. The broad business function comprises a variety of tasks ranging from prioritizing and selecting requirements to preparing the launch with internal and external stakeholders through negotiations and trainings. (2) A product manager conducts *product planning* to construct a roadmap, which is an action plan for a specific time period. The business function consists of three activities: roadmap intelligence, product roadmapping and core asset roadmapping. Information gathered in the first activity is input during actual roadmap construction. (3) *Portfolio management* consists of gathering strategic information and making product decisions across the entire portfolio [21]. Decision support information gathered during market analysis contributes to the business's decision making on product lifecycle management and partnering & contracting.

IV. Software Product Manager and Software Architect Interdependence

Of architects' primary tasks, product managers are involved in two: *developing project (product) strategy* and *communicating with stakeholders*. While SPM has a strong functional focus, SA is primarily responsible for balancing the appropriate quality attributes (QAs). While architects are able to formulate adequate technical solutions for any functional requirement, their proposed solution might not be optimal considering product context [12]. For example, the QA *performance efficiency* might be imperative for generating customer reports, while *reliability* has priority for the support interface. It is the product manager's responsibility to accurately convey context to enable the architect to make the right architectural decisions. Depending on the product manager's background (30-50% marketing or engineering [1]) and position (R&D, marketing or P&L [27]), the architect can discuss QA specifics with the product manager. Besides technical stakeholders, [12] notes that architects are in frequent contact with customers and users. For SPOs, however, the large quantity of diverse customers and users poses a communication challenge. Instead, the product manager acts as a representative on both sides. He communicates customer requirements to the architect, receives feedback and reports results to the customer.

Of the three product success goals introduced in section III, the architect has direct impact on (1) creating a winning

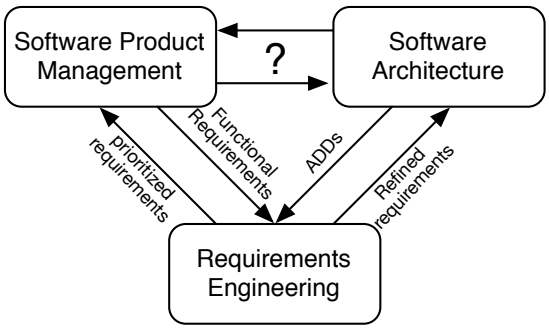

Fig. 3. SPM, RE and SA relationships

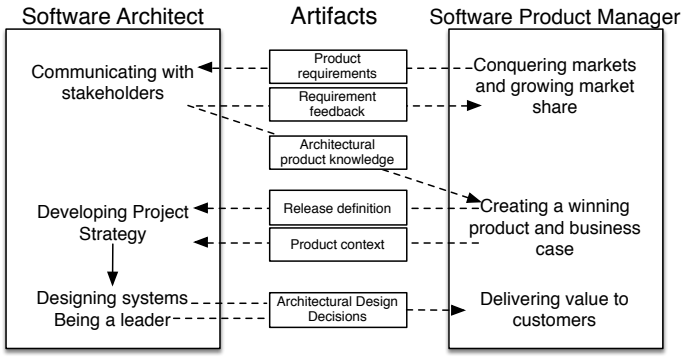

Fig. 4. Reciprocal Contributions to SPM and SA goals

product (and business case) and (3) delivering value to customers. Initially, we speculated that the product manager assumes the left peak of the Twin Peaks model. He exchanges functional details to refine requirements concurrently with the architect's specifications. In a previous case study however, the product manager's requirement refinement scope is limited: "The detailed definition of requirements is performed in three steps, of which only the first one is performed by the SPM team(s) [...] The software development teams then elaborate these [high-level requirement definitions] into requirements containing a detailed description of some desired functionality, described in sufficient detail to work with" [31]. The SPMCM by [21] does not recognize requirement refinement as an SPM activity to begin with. Instead, we believe there is a conceptual relationship between three activities: SPM, RE and SA. SPM is primarily responsible for high-level strategic issues such as market analysis, product roadmapping and fostering high-level requirements. RE refines the high-level requirements into sufficient detail to enable SA to realize the actual end-product. The information exchange between SA and SPM and their respective impact on one another's activities, however, is unknown. Figure 3 displays the conceptual relationship between SPM, RE and SA and illustrates this paper's central question: how do SPM and SA collaborate?

Requirements' central role for both stakeholders is a recurring theme in this paper. Within the SPMCM, product managers involve SA during requirements gathering, identification, organizing, and prioritization. Architects require SPM input to refine requirements, formulate correct architectural approaches and communicate with external stakeholders. Both the product manager and architect have a common goal: satisfy customer requirements to deliver value and create product success. Reaching this goal demands selecting the right requirements for a release and enabling developers to implement requirements in the right manner. While organizing, identifying requirements and formulating architectural approaches are essential, we believe these activities are out of the stakeholder's respective scope. Based on these literature findings, we posit that product managers' and architects' contribute to each other's goals in four activities: (1) requirements gathering, (2) prioritization, (3) refinement and (4) transferring product context. In terms of the conceptual relationship in Figure 3, the product manager provides product context and requirement details to the architect in exchange for architectural expertise in order to enhance one another's decision making.

We elaborate on the conceptual relationship by taking the perspective of SPM and SA their primary goals as introduced in sections II and III (Figure 4). The goals are connected with artifact flows we believe product managers and architects to exchange. Note that you should not read these flows as a linear route, but as an indefinite, iterative process. The result displays the mutual contributions to their respective goals, which we detail in this paragraph. As the primary interface for internal and external stakeholders towards the architect, the product manager receives a continuous stream of market requirements and customer requests. He organizes and selects specific *product requirements* to present to the architect. The architect responds with *requirement feedback*, which the product manager communicates to the appropriate stakeholders with the intention to conquer markets and grow market share. On the second level, the architect transfers *architectural product knowledge* to the product manager to enable collaborative requirements gathering, identification, organization, prioritization and refinement. The result is a mature *release definition*, which the architect combines with *product context* to *develop a project strategy*. Next, the architect leads the development team to design the system, making *architectural design decisions* based on the project strategy containing all provided inputs. These architectural design decisions lead to a finished product release and in turn deliver value to the customer.

To support reciprocal information exchange, product managers and architects need architecture views that relate requirements and architecture by displaying information and data on the intersection of their respective responsibilities. Unfortunately, the approaches relating requirements and architecture mentioned in Section II do not satisfy this need. Their goal is to facilitate software architecture design by developing functional requirements models. Identifying a similar disparity, [40] introduced the Functional Architecture Framework (FAF): an architectural view instrumentation that aims to support requirements management for SPOs with high requirement quantities. By linking requirements to specific

architectural application modules, requirement responsibility can immediately be assigned to the correct individual or team. To use the FAF, the SPM must first describe the product's functional architecture with a graphical architectural description language called the *Functional Architecture Model* [41]. It consists of a product context, a product scope that includes all high granularity application modules and typically two to three layers of sub-models. Next, the product manager assigns all existing requirements to the appropriate submodules and allocates newly incoming requirements in a similar manner. Finally, when a functional requirement enters development, the architect links his implementation solution enabling the product manager to analyze which functional requirements share a common technical component.

V. Case Study Research Approach

The previous sections explicated why we believe that collaboration between SPM and SA is crucial for product success. However, no empirical research that substantiates this claim by speaking to practitioners is available. As a first step, we answer the research question *"How do software product managers and software architects collaborate?"* by conducting five case studies with product managers and architects from five SPOs. Each SPO originates from the Netherlands, but their age, size and target market(s) differentiate. One SPO is an autonomous subsidiary of a bank with several small technical teams, while another is a 30 year old multinational with 2000 employees buildings software supporting SMEs. Each case study consists of an interview of approximately 1,5 hours with both a (senior) product manager and (senior) architect.

Due to the exploratory nature of this research, a formally structured interview or standardized questionnaire was unsuitable [42], [43]. Instead, we held semi-structured interviews to allow respondents to speak freely and the interviewer to pose follow-up questions, while maintaining the ability to compare responses. For each interview, the end goal was to establish how the product manager and architect at the case study company collaborate. To this end, the interviewer posed 7 questions such as: *"What is the role of SPM in the organization?"* and *"Does the organization utilize any instrumentations to facilitate their collaboration?"*.

After each interview, the interviewer put together a case study report, describing the product manager and architect's role in the organization, how both roles collaborate, whether either experienced any problems in this collaboration and what kind of instrumentations they utilize. Based on the case study report, the interviewer records whether the respondents collaborate in the four collaborative activities.

Before we commenced our case studies, we examined [44]'s guidelines for case study research. Our approach complies with the three conditions to determine when a case study is appropriate for your research: (1) we pose a 'how' research question, (2) we do not have control of behavioral events and (3) we focus on contemporary events. Furthermore, the research approach adheres to the criteria for research design quality as follows: we use multiple sources of evidence to test construct validity, create a rich theoretical framework from previous literature to ensure both internal and external validity and build case study protocols and databases for reliability.

VI. Results - SPM and SA Collaboration Patterns

All case companies have similar approaches to SPM and SA. To begin, all claim to practice agile development, which prescribes a distinct (management) approach. The architects in particular have loosely defined roles, which they grew into as the company and their seniority grew. They are the technical lead of a development team with the additional responsibility to ensure sound architectural decisions. As [12] puts it, the architect is considered "first among equals", leading the development team and communicating with relevant stakeholders such as the product manager. Due to the organizations' agile approach, architects only make ADDs before the start of a project for architecturally relevant requirements while ADDs for regular requirements are made on an ad-hoc basis. Identifying which requirements are architecturally relevant is done during requirements refinement, which each case companies conducts to some extent. Subsection VI-B further details this collaborative effort.

A. Generic Findings

None of the respondents declared that transferring product context is an important part of SPM and SA collaboration. Yet, our case study architects possess deep contextual knowledge due to two circumstances. As the product grows, the architect's contextual knowledge organically grows along. When an architect works on the same product for a long time, he is bound to establish an accurate contextual model. Furthermore, the architects at 3 companies were part of the product's initial development team and thus had intimate knowledge of the domain, while the product manager role was introduced only after product success. Respondents argue that these factors substitute the need for contextual clarification. Architects do rely on SPM to communicate with customers and users. Product managers function as intermediaries for external stakeholder requests that might concern the architect. Support and sales first report to the product manager, who decides whether to escalate the issue to the architect. Any resulting information or actions traverse the same paths.

Contrary to what previous literature prescribes [1], product managers at our case study companies typically also assume project management responsibilities. While each SPO gathers, identifies, organizes, and prioritizes requirements to create new release definitions, none had identical approaches. Despite this difference, all product managers independently identify and organize requirements. The responsibility and stakeholders involved during requirements gathering and prioritization diverge. Three case companies have formalized meetings where the product manager gathers requirements from the architect. The remaining architects transfer requirements on an incidental basis. One product manager conducts prioritization independently, only consulting the architect when necessary. At three other SPOs the product manager relies on the architect's

input to make informed requirement decisions. The extent of this reliance depends on the product manager's technical product knowledge and capabilities. The following paraphrase succinctly illustrates the typical difference between product managers with an engineering and marketing background:

> Alice and I are product managers at FinComp. Alice has an engineering background, I have a lot of experience in professional services. Before the start of a sprint, Alice works out a first level of architecture and validates it with the architect. I told the architect not to even come to me with technical questions because I cannot provide answers. Alice can challenge architectures he comes up with, I cannot and thus rely 100% on the architect to advice me on technical requirements.

One SPO has a special committee to collaboratively make decisions on requirements. Although the resulting broad stakeholder support is beneficial, the organization loses decisiveness and agility due to extra process overhead. The product manager and architect proposed a solution:

> "We'd like a portion of the release to fully be our responsibility so we can autonomously decide to implement some product requirements regardless of current customer requests"

B. Important Overlapping SPM & SA Processes

Product managers and architects exchange information during two activities: requirements *gathering* and *refinement*. Gathering requirements from a variety of different stakeholders is crucial as a product manager [21]. Of all the internal stakeholders that contribute to the product manager's requirements gathering process, the architect is unique. He is the only actor with the capability to identify technical debt and formulate requirements to solve critical deficiencies. One respondent that transitioned from engineering into SPM clarifies the importance of formally requesting the architect for input:

> "Gathering requirements from the architect is our most important collaborative process. Due to my functional focus I no longer notice where code quality is degrading and overdue maintenance is growing."

The quote above illustrates that even product managers with a (strong) engineering background are unable to maintain in-depth technical knowledge of the product(s) they manage. This characterizes the first interdependence of SPM and SA: the product manager is responsible for scheduling requirements including technical maintenance, while the architect has the domain knowledge to identify which product aspects need maintenance. To this end, three of our case companies have scheduled meetings to discuss technical maintenance. The architect presents which product sections require attention and why, the product manager shares his position from a business perspective and together they decide what issues to invest in. During these meetings information exchange is verbal, at times supported by simple informal models that happen to resemble the Functional Architecture Model by [41]. Depending on the technical competence of the product manager, both spend less or more energy on this communicative effort. At another case company, technical maintenance is a standard part of each sprint and the architect's responsibility. While this permits more effectiveness and autonomous decisiveness for the architect, the risk of focusing resources on the wrong issues increases.

Each of the case companies conducts requirement refinement, though none have adapted [31]'s guidelines. Advanced concepts such as a product management backlog and alternating sprints with development have not been formally integrated into the organizations. Instead, SPM refines requirements on a case by case basis into high-level definitions. Next, during a formalized (bi-)weekly meeting, product management refines requirements with the development team and architect in an agile, iterative manner. The product manager attends these meetings to coordinate and support the development team. He answers questions development poses, validates their interpretation of his work and might even lead the meeting. Different topics come up, at times including requirement validation and dependency linking; sub-activities of requirements identification and organizing. Two product managers note that it is imperative to convey *why* they want to develop a feature. This enables developers to autonomously answer questions, preventing additional information requests during development and/or incorrectly implemented features. Requirements go through as many refinement iterations as necessary until the requirement definition has sufficient detail for development to start developing in the next sprint. Effectively, SPM and SA are going through the Twin Peaks stages until they hash out the lowest level of detail from the requirements perspective. Due to the frequency of this collaboration and importance to creating a winning product and delivering value to customers, we claim that requirement refinement is the most important process during which product managers and architects collaborate and exchange information.

C. Instrumentations Facilitating SA & SPM Information Exchange

All respondents recognize that they use a variety of tools, but note that few are applicable to this research context. For example, each SPO uses a tool to track features such as Jira, OnTime or Pivotal Tracker. These tools are central to their development activities, primarily benefiting short term project management instead of the more conceptual information exchange between an architect and product manager.

Just one of the case companies claims to use a tool specifically to facilitate information exchange between technical and less technical roles: Enterprise Architect (EA). Although EA has a variety of features, this company primarily used its modeling capabilities to create both technical and functional models of their product which are connected by a data model. The product manager discusses their EA experiences:

> "EA improves [people] alignment and [communication] clarity. Recording ideas in models well enables us to discourse efficiently; explaining how, why and what you want becomes comprehensible. In turn, translating functionality and business requirements into technical requirements is now significantly more logical. Furthermore, because all stakeholders operate around the different models, alignment occurs far earlier in the lifecycle."

Despite these advantages, respondents from the other four case companies explicitly stressed their aversion to static documentation tools. Although none object to modeling itself, they warn against meticulously creating models for company-wide usage. Because, despite their meticulous effort, modeling is a human activity that *will* produce errors. One respondent stated: "No documentation is better than outdated and thus wrong documentation", to which his co-interviewee, a product manager with limited technical background, added the following nuance: "If it's not a living document, it does not serve a purpose. If it's got a lifespan of X amount of time, don't invest in it. If you know it's a living document that you're going to refer to for the rest of your life, then nobody has an issue updating it.". Yet, neither could think of a document within their SPO with this kind of urgency.

Instead, these respondents create rough models, (functional, data, flow, etc) on a whiteboard when necessary. Aside from always being up to date, this approach permits hands-on collaboration on the whiteboard and communicative flexibility in what aspects of the model you draw, directing focus to only those parts that matter. Often, these models resemble the Functional Architecture Modeling technique [41], consisting of high-level elements that exchange high-level data objects. When participants deem a specific architectural approach is desirable, someone photographs the drawing and adds it to the requirement specification. The drawing then provides high-level guidance without becoming a fully thought-out implementation plan.

VII. Discussion

Case study analysis shows that all of the SPOs have some form of SPM and SA interdependence, typically addressed at a (bi-)weekly meeting. Although the formalization degree of these meetings varies, requirements are a common central topic. We found direct indications for three out of four collaborative activities presented in subsection IV: (1) requirements gathering, (2) prioritization and (3) refinement. Much to our surprise, respondents indicate that architects do not need (4) product context clarification from product managers to identify important QAs and formulate correct architectural approaches.

The case study SPOs avoid having to communicate product context to new architects by not hiring external recruits for architectural roles. Instead, they promote software engineers whose contextual knowledge already is close to on par with the product manager. Furthermore, the architect receives sufficient contemporary contextual information in an implicit manner during his involvement with the product. However, we believe explicit context exchange is beneficial for the architect's understanding in the long run. One architect clarifies: "We used to be our own customers. Nowadays we are out of touch with what it's like to work in the domain. Because we haven't done the work in 15 years and the field has evolved drastically, we need to test our assumptions with customers."

All other requirement related activities are part of the collaboration: product managers request SA input for requirements prioritization, the architect has a unique contribution for requirements gathering and together they refine requirements into sufficient detail for development. Additionally, during refinement different topics come up including requirement validation and dependency linking. Through these two sub-activities, they collaborate on requirements identification and organization indirectly as well. Finally, the product manager acts as an intermediary between the architect and customer requests and requirements.

Requirements gathering and refinement are particularly interesting due to the complex, reciprocal information exchange that takes place. When a product manager gathers requirements from internal stakeholders, the goal is to include the perspective of all relevant stakeholders [45] to gain insight into what issues are currently important within the organization. Unfortunately, there is no specific literature describing how product managers gather requirements, let alone how specific stakeholders take part in this activity. Without this information, formulating approaches or designing instrumentations to support SPM and SA alignment is impossible. Our case studies show that architects contribute a unique class of requirements because the product manager is unable to reliably identify technical debt himself. Thus the architect attempts to clarify the product's weak aspects in formalized meetings. However, he can only do so by conveying technical details that are conceptually out of scope for the product manager.

The extent to which our product management respondents participate in requirements refinement is out of scope in a similar manner. In theory, development conducts requirement refinement to add sufficient detail to underspecified requirements, enabling each team member to fulfill any requirement in a later stage. However, four of the product managers participate in requirements refinement sessions. In their experience, the independent, document-driven approach for refinement described in [31] generates too many requests for more details. Instead, architects refine requirements collaboratively with product managers to resolve impediments as early and effectively as possible.

These findings enable us to formulate an answer to our main research question. Software product managers and software architects collaborate in three ways: they exchange information to gather requirements, collectively prioritize requirements and iteratively refine requirements selected for a release. These three activities have large, direct influence on product success. After all, gathering, selecting and refining the *right* requirements in the *right* manner is a prerequisite to delivering the *right* value to customers.

Consequentially, aligning involved stakeholders in these tasks is imperative to creating a winning product. Figure 4 presented the reciprocal contributions between SPM and SA, demonstrating that the key to alignment is facilitating their different forms of information exchange by creating a shared understanding of the product. Models that both stakeholders are able to reason about and discuss have the potential to attain this goal. However, of our respondents, only one SPO utilizes formalized technical and data models structured in a company-wide standardized manner. Although their experience with the tool is positive, all other respondents note one critical drawback: human-made models are outdated, wrong or both and using them will lead to corresponding decisions.

These skeptical respondents rely solely on the most basic communicative tool, drawing on a whiteboard on an incidental basis. [46] uncovered similar results, noting that architects primarily utilize whiteboard sketches to communicate architecture models to secondary stakeholders. While they claim the benefits of these transient models are great, they forgo SA's analytical and educational merits. Moreover, we question the accuracy and timeliness of these high-level model sketches in practice. Especially considering that as product complexity grows and subsequently the number of people involved in development, accurately depicting an SA from memory becomes exponentially more difficult. This presents us with a conundrum: if neither formal nor incidentally drawn models are applicable for facilitating SPM and SA alignment, what approach should an SPO take?

VIII. Proposed Solution

SPOs face a dilemma in their efforts to align SPM and SA. Our case studies show that SPOs have two approaches to documenting a product: multiple, formalized models or no models at all. Over time, the former leads to outdated, inaccurate models that harm the organization, while the latter forgoes all benefits of models by relying on stakeholders' communicative competence to exchanging the right information. We require an approach that utilizes models understandable for both stakeholders and ensures their timeliness and accuracy.

Combining the literature review and results from our case studies, we identify four architectural model uses relevant for product software: (1) *incidental models* which stakeholders draw when the need presents itself during meetings, (2) *informal models* that record incidental models for later reference if necessary, (3) *discussion models* depict the product on a high (functional) level to enable technical stakeholders to exchange information with non-technical stakeholders and (4) *structured models* take a formalized approach at depicting architecture such as data models for documentation purposes, possibly captured in tooling. Of these, discussion models (DMs) are particularly relevant for our purposes. Anyone can create a simple DM by using requirements as guidance and DMs can have multiple degrees of detail for different stakeholders. These characteristics adhere to the requirements argued by [47] for aligning requirements with architecture: requirements are decomposed into fine grained specifications and organized in a similar manner to the system's architecture.

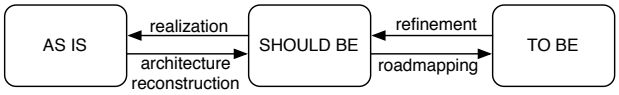

Fig. 5. Accurate Architectural Models Approach

Unfortunately however, DMs become outdated and inaccurate as well. While [40] prescribes continuously monitoring the architecture and adjusting the FAF when it is no longer accurate, the following scenario happens in practice: over time the discussion model (DM) starts to diverge from the implemented situation. The architect, however, is confident the DM is accurate enough for the task at hand and chooses not to update it or forgets to do so after implementation. Meanwhile, the product manager re-uses the inaccurate model for new requirements. Three iterations later the DM diverges from reality to such an extent that it is useless and both stakeholders complain when they have to create a new one. Concisely, human error and/or negligence causes DMs to become outdated, wrong or both as well.

While many different models exist, SPOs lack a structured approach to prevent model divergence from the implemented situation. To this end we propose the Accurate Architectural Models Approach (AAMA) that engages both the product manager and architect to ensure ongoing accuracy and timeliness. AAMA consists of three co-existing model instantiations: a current technical truth, a future design and a combination of the two. This division is similar to ones used in architecture compliance and auditing [48], [49]. To work with these models, stakeholders follow four distinct steps as shown in Figure 5.

- **AS IS** The product's current situation as seen from the architect's technical perspective.
- **SHOULD BE** The product's current design, consists of the current situation and all pending design changes based on new requirements.
- **TO BE** A future design of the product based on new requirements to be included in a release.

This paragraph presents a scenario introducing how SPOs can apply AAMA to improve SPM and SA alignment. As a product matures, the product manager and architect recognize the need to approach their collaboration in a more sophisticated manner and decide to apply AAMA to their discussion models. Together they create an *origin model* based on the product's current situation to facilitate their information exchange. On this originating moment, the AS IS and SHOULD BE are identical and both stakeholders agree to apply AAMA by adhering to the following four steps.

1) **Roadmapping** New requirements prompt the product manager to use the origin model to create a TO BE.
2) **Refinement** In the second step, both stakeholders refine the TO BE to create a SHOULD BE.

3) **Realization** The architect consults the SHOULD BE to realize the new requirements.
4) **Architecture Reconstruction** After realization, a new AS IS situation materializes which diverges from the SHOULD BE, prompting the architect to conduct architecture reconstruction to warrant the SHOULD BE's accuracy.

Finally, the cycle restarts when the product manager uses the newly updated and accurate SHOULD BE as input.

Applying AAMA has three consequences for the models position in the organization. (1) Frequent reviews and updates makes the DM a *living model*, preventing outdated models and human made errors. (2) By ensuring that architects as well as product managers examine and update the DM on a frequent basis, both have a deep understanding of the recent changes to the product. (3) Having a high architectural view available on demand enables all stakeholders to create incidental models of strictly those product sections relevant to the discussion. In turn, these consequences have three positive influences on the reciprocal contributions of SPM and SA:

1) The common view and mutual deep understanding assists the frequent communication on product requirements and requirements feedback.
2) The architect transfers the right architectural product knowledge more effectively and efficiently because he knows the extent of the product manager's architectural knowledge.
3) AAMA's iterative TO BE DM refinement aligns with the Twin Peaks model. The product manager expresses requirements in the TO BE DM, which the architect examines for implementation and requests further refinement details from the product manager. The resulting resulting release definition is more complete and unambiguous, contributing to the architect's technical product strategy.

Although AAMA prevents cumulative DM divergence from the current implementation, one major shortcoming remains in the product software context. During implementation, the product manager is already processing new, continuously incoming requirements into a new TO BE DM. The moment the SHOULD BE DM is updated, the product manager has already produced a new TO BE DM, possibly creating a divergence. This divergence will result in either the product manager having to alter his DM or the architect to work with an inaccurate specification, reducing the DM's utility.

AAMA is a giant leap forwards, but requires further improvements to achieve continuous model accuracy. Some respondents wish to automatically generate high-level architecture views from data supplied by instrumentations they already use. A sentiment echoed in literature: "the most useful forms of documentation are views of the software that can be automatically generated" [50]. Automatically generating a DM from source code and subsequently connecting all elements to requirements from the requirement database is an exciting concept. However, automatic architecture reconstruction to date has only a few incomplete successes for specific uses cases [51], [52]. The accuracy and relevance of an instrument that automatically generates architectural views is doubtful and the prospect of having the architect adjust the entire view manually is considerably less exciting.

Instead, future design science research will use this qualitative study's results to formulate an automated prototype. It will unite the three different DM instantiations into one shared, digital model to prevent divergence at any stage. After prototype improvement interviews with practitioners, we plan to conduct a survey to determine Dutch product managers' and software architects' usage intention towards AAMA.

IX. Conclusion

The start of this paper asks *"how do software product managers and software architects collaborate?"*. By means of five elaborate case studies we uncover two critical processes where product managers and architects exchange information: requirements gathering and requirements refinement. These activities require reciprocal information exchange on a high technical level to formulate and refine the right requirements in the right way. The successful execution of these processes is a critical driver to product success. Respondents, however, rely on primitive methods and lack a structured approach that warrants SPM and SA alignment. We believe SPOs can improve their SPM and SA alignment by adopting discussion models and propose the Accurate Architectural Models Approach (AAMA) to prevent architectural model divergence.

In future research, we aim to further elaborate and formalize the SPM and SA interplay in order to benefit their alignment and resolve discrepancies between current literature and the findings in this paper. For instance, we intend to determine whether requirements refinement is truly out of scope for product managers, internal promotions always fill architectural vacancies and validate AAMA's potential benefits compared to updating models on an incidental basis. Furthermore, designing and validating an instrumentation to facilitate information exchange further establishes the research domain. Finally, a longitudinal study measuring the effect of applying AAMA and/or facilitating instrumentations will validate the resulting artifacts and impact of our efforts.

References

[1] C. Ebert, "The Impacts of Software Product Management," *Journal of Systems and Software*, vol. 6, no. 80, pp. 850–861, 2007.
[2] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions," in *Proceedings of WICSA '05*, 2005, pp. 109–120.
[3] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little, *Documenting Software Architectures: Views and Beyond*. Pearson Education, 2002.
[4] B. Nuseibeh, "Weaving together requirements and architectures," *Computer*, vol. 34, no. 3, pp. 115–119, Mar. 2001.
[5] P. T. Ward and S. J. Mellor, *Structured Development for Real-Time Systems: Vol. I: Introduction and Tools*. Prentice Hall, 1986.
[6] M. Khurum and T. Gorschek, "A method for alignment evaluation of product strategies among stakeholders (mass) in software intensive product development," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 23, no. 7, pp. 494–516, 2011.
[7] B. Regnell, R. Berntsson Svensson, and T. Olsson, "Supporting roadmapping of quality requirements," *Software, IEEE*, vol. 25, no. 2, pp. 42–47, 2008.

[8] K. Wnuk, B. Regnell, and C. Schrewelius, "Architecting and coordinating thousands of requirements an industrial case study," in *Requirements Engineering: Foundation for Software Quality*, ser. LNCS. Springer, 2009, vol. 5512, pp. 118–123.

[9] D. Condon, *Software Product Management*. Boston, Massachusetts: Aspatore Books, 2002.

[10] A. Helferich, K. Schmid, and G. Herzwurm, "Product management for software product lines: An unsolved problem?" *Communications of the {ACM}*, vol. 49, no. 12, pp. 66–67, Dec. 2006.

[11] M. W. Maier, D. Emery, and R. Hilliard, "Software architecture: Introducing ieee standard 1471," *Computer*, vol. 34, no. 4, pp. 107–109, Apr 2001.

[12] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, 2010.

[13] J. ISO, "IEC 25010: 2011: Systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models," *International Organization for Standardization*, 2011.

[14] J. G. Hall, M. Jackson, R. C. Laney, B. Nuseibeh, and L. Rapanotti, "Relating software requirements and architectures using problem frames," in *IEEE International Requirements Engineering Conference (RE'02)*, 2002, pp. 137–144.

[15] P. Grünbacher, A. Egyed, and N. Medvidovic, "Reconciling software requirements and architectures with intermediate models," *Software and Systems Modeling*, vol. 3, no. 3, pp. 235–253, 2004.

[16] R. Chitchyan, M. Pinto, A. Rashid, and L. Fuentes, "Compass: Composition-centric mapping of aspectual requirements to architecture," in *Transactions on Aspect-Oriented Software Development IV*, ser. LNCS. Springer, 2007, vol. 4640, pp. 3–53.

[17] K. M. van Hee, J. J. A. Keiren, R. D. J. Post, N. Sidorova, and J. M. E. M. van der Werf, "Designing case handling systems," in *Transactions on Petri Nets and Other Models of Concurrency I*. Springer, Berlin, 2008, vol. 5100, pp. 119 – 133.

[18] P. Avgeriou, J. Grundy, J. G. Hall, P. Lago, and I. Mistrk, *Relating Software Requirements and Architectures*, 1st ed. Springer, 2011.

[19] H. Vogl, K. Lehner, P. Grunbacher, and A. Egyed, "Reconciling requirements and architectures with the cbsp approach in an iphone app project," in *Requirements Engineering Conference (RE), 2011 19th IEEE International*, Aug 2011, pp. 273–278.

[20] C. Chen, D. Shao, and D. Perry, "An exploratory case study using cbsp and archium," in *Second Workshop on Sharing and Reusing Architectural Knowledge - Architecture, Rationale and Design Intent*. IEEE, May 2007, pp. 3–3.

[21] W. Bekkers, I. van de Weerd, M. Spruit, and S. Brinkkemper, "A Framework for Process Improvement in Software Product Management," in *Proceedings of EuroSPI*, 2010a, pp. 1–12.

[22] C. Ebert, "Software Product Management," *Crosstalk*, vol. 22, no. 1, pp. 15–19, 2009.

[23] L. Xu and S. Brinkkemper, "Concepts of Product Software," in *European Journal of Information Systems*, vol. 16, no. 5, 2007, pp. 531–541.

[24] A. Maglyas, U. Nikula, and K. Smolander, "What do practitioners mean when they talk about product management?" in *20th International Requirements Engineering Conference (RE)*, Sept 2012, pp. 261–266.

[25] I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma, "On the creation of a reference framework for software product management: Validation and tool support," in *International Workshop on Software Product Management*, 2006, pp. 3–12.

[26] L. Gorchels, *The Product Manager's Handbook: The Complete Product Management Resource (2nd edition)*, 2nd ed. Columbus, OH, USA: McGraw-Hill, 2000.

[27] C. Ebert and S. Brinkkemper, "Software product management an industry evaluation," *Journal of Systems and Software*, no. 0, pp. –, 2014.

[28] S. Fricker, "Software product management," in *Software for People*, ser. Management for Professionals. Springer, 2012, pp. 53–81.

[29] A. Maglyas, U. Nikula, and K. Smolander, "What do we know about software product management? - a systematic mapping study," in *Software Product Management (IWSPM), 2011 Fifth International Workshop on*, Aug 2011, pp. 26–35.

[30] H.-B. Kittlaus and P. N. Clough, *Software Product Management and Pricing: Key Success Factors for Software Organizations*. Berlin, Germany: Springer, 2009.

[31] K. Vlaanderen, S. Jansen, S. Brinkkemper, and E. Jaspers, "The agile requirements refinery: Applying {SCRUM} principles to software product management," *Information and Software Technology*, vol. 53, no. 1, pp. 58 – 70, 2011.

[32] A. M. Davis, *Just Enough Requirements Management: Where Software Development Meets Marketing*. New York, NY, USA: Dorset House Publishing Co., Inc., 2005.

[33] B. Nuseibeh and S. Easterbrook, "Requirements engineering: A roadmap," in *The Future of Software Engineering*, ser. ICSE '00. New York, NY, USA: ACM, 2000, pp. 35–46.

[34] C. Potts, "Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software," in *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, 1995.

[35] L. Karlsson, sa G. Dahlstedt, B. Regnell, J. N. och Dag, and A. Persson, "Requirements engineering challenges in market-driven software development an interview study with practitioners," *Information and Software Technology*, vol. 49, no. 6, pp. 588 – 604, 2007.

[36] P. Sawyer, I. Sommerville, and G. Kotonya, "Improving market-driven re processes," in *Proceedings of the International Conference on Product Focused Software Process Improvement*, vol. 195, Oulu, Finland, 1999, pp. 222–236.

[37] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE Software*, vol. 14, no. 5, pp. 67–74, 1997.

[38] J. Natt och Dag, B. Regnell, V. Gervasi, and S. Brinkkemper, "A linguistic-engineering approach to large-scale requirements management," *Software, IEEE*, vol. 22, no. 1, pp. 32–39, 2005.

[39] J. Kabbedijk, S. Brinkkemper, S. Jansen, and B. van der Veldt, "Customer involvement in requirements management: Lessons from mass market software development," in *Requirements Engineering Conference, 2009. RE '09. 17th IEEE International*, 2009, pp. 281–286.

[40] T. Salfischberger, I. van de Weerd, and S. Brinkkemper, "The functional architecture framework for organizing high volume requirements management," in *Software Product Management (IWSPM), 2011 Fifth International Workshop on*, Aug 2011, pp. 17–25.

[41] S. Brinkkemper and S. Pachidi, "Functional architecture modeling for the software product industry," in *Software Architecture*, ser. LNCS. Springer, 2010, vol. 6285, pp. 198–213.

[42] P. Corbetta, *Social Research: Theory, Methods and Techniques*. London: Sage Publications, 2003.

[43] A. Kajornboon, "Using Interviews as Research Instruments," *E-Journal for Research Teachers*, vol. 2, no. 1, 2005.

[44] R. K. Yin, *Case Study Research - Design and Methods*. SAGE Publications, 2009.

[45] W. Bekkers, M. Spruit, I. van de Weerd, R. van Vliet, and A. Mahieu, "A situational assessment method for software product management," in *Proceedings of ECIS*, Pretoria, South-Africa, 2010b, pp. 22–34.

[46] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko, "Let's go to the whiteboard: How and why software developers use drawings," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 557–566.

[47] M. W. Whalen, A. Gacek, D. Cofer, A. Murugesan, M. P. E. Heimdahl, and S. Rayadurgam, "Your "what" is my "how": Iteration and hierarchy in system design," *IEEE Software*, vol. 30, no. 2, pp. 54–60, 2013.

[48] R. Koschke and D. Simon, "Hierarchical reflexion models," in *Proceedings of the 10th Working Conference on Reverse Engineering*, ser. WCRE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 36–.

[49] W. van der Aalst, K. van Hee, J. M. van der Werf, A. Kumar, and M. Verdonk, "Conceptual model for online auditing," *Decision Support Systems*, vol. 50, no. 3, pp. 636 – 647, 2011.

[50] M. Mirakhorli and J. Cleland-Huang, "Traversing the twin peaks," *IEEE Software*, vol. 30, no. 2, pp. 30–36, 2013.

[51] F. Schmidt, S. MacDonell, and A. Connor, "An automatic architecture reconstruction and refactoring framework," in *Software Engineering Research, Management and Applications 2011*, ser. SCI, R. Lee, Ed. Springer, 2012, vol. 377, pp. 95–111.

[52] F. A. Fontana and M. Zanoni, "A tool for design pattern detection and software architecture reconstruction," *Information Sciences*, vol. 181, no. 7, pp. 1306 – 1324, 2011.