Architectural Decision Guidance across Projects

프로젝트 전반에 걸친 아키텍처 의사 결정 지침

Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge

문제 공간 모델링, 의사 결정 백로그 관리 및 클라우드 컴퓨팅 지식

Olaf Zimmermann, Lukas Wegmann Institute for Software. Hochschule fur Technik (HSR FHO) Rapperswil, Switzerland {firstname.lastname} @hsr.ch

올라프 짐머만, 루카스 웨그만 소프트웨어 연구소. Hochschule für Technik (HSR FHO) 스위스 라퍼스빌 {firstname.lastname} @hsr.ch

Heiko Koziolek, Thomas Goldschmidt Research Area Software. ABB Corporate Research Ladenburg, Germany {firstname.lastname} @de.abb.com

Heiko Koziolek, Thomas Goldschmidt 연구 영역 소프트웨어. ABB 기업 연구 독일 라덴부르크 {firstname.lastname} @de.abb.com

Abstract- Architectural Knowledge Management (AKM) has been a major topic in software architecture research since 2004. Open AKM problems include an effective, seamless transition from reusable knowledge found in patterns books and technology blogs to project-specific decision guidance and an efficient, practical approach to knowledge application and maintenance. We extended our previous work with concepts for problem space modeling, focusing on reusable knowledge, as well as solution space management, focusing on project-level decisions. We implemented these concepts in ADMentor, an extension of Sparx Enterprise Architect. ADMentor features rapid problem space modeling, UML model linkage, question-option-criteria diagram support, meta-information for model tailoring, as well as decision backlog management. We validated ADMentor by modeling and applying 85 cloud application design decisions and 75 workflow management decisions, creating one problem and three sample solution spaces covering control system architectures, and obtaining user feedback on tool and model content.

초록 - AKM(Architectural Knowledge Management)은 2004년부터 소프트웨어 아키텍처 연구의 주요 주제였습니다. 개방형 AKM 문제에는 패턴 서적과 기술 블로그에서 찾을 수 있는 재사용 가능한 지식에서 프로젝트별 의사 결정 지침, 지식 적용 및 유지 관리에 대한 효율적이고 실용적인 접근 방식으로의 효과적이고 원활한 전환이 포함됩니다. 우리는 재사용 가능한 지식에 초점을 맞춘 문제 공간 모델링 개념과 프로젝트 수준 의사 결정에 초점을 맞춘 솔루션 공간 관리로 이전 작업을 확장했습니다. 우리는 Sparx Enterprise Architect의 확장인 ADMentor에서 이러한 개념을 구현했습니다. ADMentor는 신속한 문제 공간 모델링, UML 모델 연결, 질문-옵션-기준 다이어그램 지원, 모델 조정을 위한 메타 정보 및 의사 결정 백로그 관리를 제공합니다. 85개의 클라우드 애플리케이션 설계 결정과 75개의 워크플로 관리 결정을 모델링 및 적용하고, 제어 시스템 아키텍처를 다루는 1개의 문제와 3개의 샘플 솔루션 공간을 생성하고, 도구 및 모델 콘텐츠에 대한 사용자 피드백을 얻어 ADMentor를 검증했습니다.

Keywords-agile practices; architectural synthesis; architectural decisions; cloud computing; knowledge management; patterns; UML;

키워드 - 애자일 관행; 건축 합성; 건축 결정; 클라우드 컴퓨팅; 지식 관리; 패턴; UML입니다.

1. INTRODUCTION

1. 소개

Lindvall and Rus stated that "the major problem with intellectual capital is that it has legs and walks home every day' [10]. In response to this threat, codification and personalization strategies can be applied. The Architectural Knowledge Management (AKM) community has proposed support for these strategies in metamodels, methods, and tools since 2004 [1]. Most AKM work focusses on capturing architectural decisions after the fact; e.g., the ISO/IEC/IEEE standard 42010:2011 recommends the capturing of decision rationale in architecture descriptions [8]. Such rationale answers why-questions concerning architectural synthesis. However, little attention has been paid to making the gathered knowledge applicable to multiple projects across organizations and to give decisions a guiding, lasting role in design processes..

린드발과 루스는 "지적 자본의 주요 문제는 다리가 있고 매일 집으로 걸어간다는 것이다"라고 말했다[10]. 이러한 위협에 대응하여 성문화 및 개인화 전략을 적용할 수 있습니다. AKM(Architectural Knowledge Management) 커뮤니티는 2004년부터 메타모델, 방법 및 도구에서 이러한 전략에 대한 지원을 제안했습니다[1]. 대부분의 AKM 작업은 사후 아키텍처 결정을 캡처하는 데 중점을 둡니다. 예를 들어, ISO/IEC/IEEE 표준 42010:2011은 아키텍처 설명에서 의사 결정 근거를 포착할 것을 권장합니다[8]. 이러한 이론적 근거는 건축 합성에 관한 이유 질문에 답합니다. 그러나 수집된 지식을 조직 전체의 여러 프로젝트에 적용하고 의사 결정이 설계 프로세스에서 안내적이고 지속적인 역할을 하도록 하는 데는 거의 주의를 기울이지 않았습니다..

According to our personal experiences gained on industrial. research projects as well as many years in product development and professional services, the potential value of knowledge sharing is acknowledged by many architects (both junior and senior) - but still hard to realize in practice. Three reasons for these difficulties include a) the diversity in technology, b) the. long lifetime of decisions throughout software evolution, and c) the differences in architecture design practices from agile to "big design upfront'; this is particularly relevant in global.

산업에서 얻은 우리의 개인적인 경험에 따르면. 연구 프로젝트뿐만 아니라 제품 개발 및 전문 서비스 분야에서 다년간 지식 공유의 잠재적 가치는 많은 건축가(주니어 및 시니어 모두)에 의해 인정되고 있지만 실제로 실현하기는 여전히 어렵습니다. 이러한 어려움의 세 가지 이유는 a) 기술의 다양성, b) 입니다. 소프트웨어 진화 전반에 걸친 긴 의사 결정 수명, c) 애자일에서 "대규모 설계 선행"에 이르는 아키텍처 설계 관행의 차이; 이는 특히 글로벌과 관련이 있습니다.

firms with many geographically distributed, federated and loosely coupled development organizations. Project constraints such as frequent due dates, budget limitations and stakeholder pressure cause a lot of precious knowledge to remain tacit (i.e., in people's heads) [20]. In our previous work, we have presented solutions to make architectural decisions sustainable [24] in response to issue b) and to organize the knowledge into abstraction-refinement levels [27] in response to issue a). However, issue c) remains open; let us investigate it further.

지리적으로 분산되고 연합되고 느슨하게 결합된 개발 조직이 많은 회사입니다. 잦은 마감일, 예산 제한 및 이해관계자의 압력과 같은 프로젝트 제약으로 인해 많은 귀중한 지식이 암묵적으로(즉, 사람들의 머릿속에) 남아 있습니다[20]. 이전 작업에서 우리는 문제 b)에 대한 응답으로 아키텍처 결정을 지속 가능하게 만들고[24] 문제 a)에 대한 응답으로 지식을 추상화-정제 수준[27]으로 구성하기 위한 솔루션을 제시했습니다. 그러나 문제 c)는 여전히 열려 있습니다. 더 자세히 조사해 보겠습니다.

Most decision documentation tools suffer from the lack of incentives for mere decision capturing and other real-world. inhibitors. Capturing decision knowledge in text documents after-the-fact is a promising start, but bound to fail in the long. run as document-oriented decision logs are hard to process and maintain once they reach a certain size (say, 50 to 70 decisions captured). With such approach, sequential reading or full text. searches are the main processing options. Decision docu-. mentation models promise to improve the situation, but have to be created, organized, and maintained over time as well..

대부분의 의사 결정 문서화 도구는 단순한 의사 결정 캡처 및 기타 실제 세계에 대한 인센티브가 부족합니다. 억제제. 사후 텍스트 문서에서 의사 결정 지식을 캡처하는 것은 유망한 시작이지만 장기적으로는 실패할 수밖에 없습니다. 문서 지향 의사 결정 로그는 특정 크기(예: 50-70개의 의사 결정 캡처)에 도달하면 처리 및 유지 관리가 어렵습니다. 이러한 접근 방식을 사용하면 순차적 읽기 또는 전체 텍스트가 가능합니다. 검색은 주요 처리 옵션입니다. 결정 다큐-. 멘테이션 모델은 상황을 개선할 것을 약속하지만 시간이 지남에 따라 생성, 구성 및 유지 관리되어야 합니다..

Both textual decision logs and documentation models are unable to steer the initial design and review work on a project. For instance, development processes using a milestone-based approach such as Stage Gate °R○ [19] may ask for a number of decisions to be made and documented before a milestone or gate can be passed (e.g., make-or-buy decisions and platform decisions that require product purchases). The existing decision capturing methods and tools do not make the need for such mandatory decisions explicit in the design process; they only allow decision makers to record (log) their decisions once they have identified and made them. As a consequence, the preparation for gate reviews or milestone approval meetings is time consuming and error prone, involving a lot of copy-paste and other manual content assembly work. To give an example: in cloud computing, architectural knowledge e.g. around workload exists in various forms from blog posts to white papers and patterns books [3]; this is precious, but passive knowledge. Related decisions are not made explicit; if the literature is not consulted, required decisions might be missed.

텍스트 의사 결정 로그와 문서화 모델 모두 프로젝트의 초기 설계 및 검토 작업을 조정할 수 없습니다. 예를 들어, 스테이지 게이트[19]와 같은 마일스톤 기반 접근 방식을 사용하는 개발 프로세스에서는 °R○ 마일스톤이나 게이트를 통과하기 전에 여러 가지 결정을 내리고 문서화해야 할 수 있습니다(예: 제품 구매가 필요한 메이크 또는 구매 결정 및 플랫폼 결정). 기존의 의사 결정 캡처 방법 및 도구는 설계 프로세스에서 이러한 필수 결정의 필요성을 명시적으로 만들지 않습니다. 의사 결정권자가 결정을 식별하고 내린 후에만 결정을 기록(기록)할 수 있습니다. 결과적으로 게이트 검토 또는 마일스톤 승인 회의를 준비하는 데는 시간이 많이 걸리고 오류가 발생하기 쉬우며 많은 복사-붙여넣기 및 기타 수동 콘텐츠 조립 작업이 필요합니다. 예를 들어, 클라우드 컴퓨팅에서 워크로드에 대한 아키텍처 지식은 블로그 게시물에서 백서 및 패턴 북에 이르기까지 다양한 형태로 존재합니다[3]; 이것은 소중하지만 수동적인 지식입니다. 관련 결정은 명시적으로 이루어지지 않습니다. 문헌을 참조하지 않으면 필요한 결정을 놓칠 수 있습니다.

Decision guidance modeling practices and tools are still. immature and confronted with a healthy amount of skepticism within the target audience, e.g., concerns about the organizatio-. nal feasibility of cross-product, cross-unit reuse of knowledge. as well as resulting maintenance efforts.

의사 결정 지침 모델링 관행 및 도구는 여전히 남아 있습니다. 미성숙하고 대상 청중 내에서 건전한 양의 회의론에 직면했습니다(예: 조직에 대한 우려). 제품 간, 단위 간 지식 재사용의 타당성. 뿐만 아니라 그에 따른 유지 보수 노력.

To overcome these problems, we separate decisions. required from decisions made. The resulting research questions (RQs) to be investigated in this paper are:.

이러한 문제를 극복하기 위해 우리는 결정을 분리합니다. 내린 결정에서 필요합니다. 이 논문에서 조사할 결과 연구 질문(RQ)은 다음과 같습니다.

RQ 1: How to model decisions required so that a) they are applicable to diverse projects, b) do not age fast e.g. due to technology evolution, and c) are simple to maintain over time?

RQ 1: a) 다양한 프로젝트에 적용할 수 있고, b) 기술 발전으로 인해 빠르게 노화되지 않으며, c) 시간이 지남에 따라 유지 관리가 간단하도록 필요한 결정을 모델링하는 방법은 무엇입니까?

To answer RQ1, we supersede previous metamodels for decision capturing and sharing with lean knowledge quadruples that give decisions a guiding role that works effectively and efficiently both in traditional and in agile settings.

RQ1에 답하기 위해 우리는 의사 결정에 기존 환경과 애자일 환경 모두에서 효과적이고 효율적으로 작동하는 안내 역할을 제공하는 린 지식 4배로 의사 결정 캡처 및 공유를 위한 이전 메타 모델을 대체합니다.

RQ 2: How to integrate decision modeling concepts into architecture design practices and tools commonly used by architects to evolve their designs and record decisions made along the way, without creating more effort than gains?

RQ 2: 의사 결정 모델링 개념을 건축가가 일반적으로 사용하는 아키텍처 설계 관행 및 도구에 통합하여 이득보다 더 많은 노력을 들이지 않고 설계를 발전시키고 그 과정에서 내린 결정을 기록하는 방법은 무엇입니까?

To respond to RQ 2, we annotate the decision knowledge. with meta-information, leveraging already existing organizing. principles such as viewpoints, refinement levels, and project stages. Decision capturing is streamlined by leveraging lean documentation templates (from practitioner literature) flexibly.

RQ 2에 응답하기 위해 의사결정 지식에 주석을 추가합니다. 메타 정보를 사용하여 기존 조직을 활용합니다. 관점, 개선 수준 및 프로젝트 단계와 같은 원칙. 의사 결정 캡처는 린 문서 템플릿(실무자 문헌에서)을 유연하게 활용하여 간소화됩니다.

We make our solutions (answers) to RQ 1 and RQ 2 available in a new add-in to the modeling tool platform Sparx Enterprise Architect, and demonstrate their value by creating decision knowledge for cloud computing and other domains.

우리는 모델링 도구 플랫폼 Sparx Enterprise Architect의 새로운 추가 기능에서 RQ 1 및 RQ 2에 대한 솔루션(답변)을 제공하고 클라우드 컴퓨팅 및 기타 도메인에 대한 의사 결정 지식을 생성하여 그 가치를 입증합니다.

The remainder of this paper is structured in the following way: Section II discusses the state of the art and the practice. with respect to our research questions and derives requirements. from it. Our novel research contributions towards completing the vision for an active, guiding role of architectural decisions to accelerate design work are introduced in Section III. Section IV presents an implementation of these concepts, Section V our further validation activities that include action research and experiments with architects from industry and their knowledge. Section VI then discusses the validation results and practicality. of our approach. Section VII concludes and highlights future work and other opportunities.

이 논문의 나머지 부분은 다음과 같은 방식으로 구성되어 있습니다. 섹션 II에서는 최신 기술과 관행에 대해 논의합니다. 우리의 연구 질문과 관련하여 요구 사항을 도출합니다. 그것에서. 설계 작업을 가속화하기 위한 건축 결정의 적극적이고 안내적인 역할에 대한 비전을 완성하기 위한 우리의 새로운 연구 기여가 섹션 III에 소개됩니다. 섹션 IV는 이러한 개념의 구현을 제시하고, 섹션 V는 업계의 건축가 및 지식에 대한 행동 연구 및 실험을 포함하는 추가 검증 활동을 제시합니다. 그런 다음 섹션 VI에서는 검증 결과와 실용성에 대해 논의합니다. 우리의 접근 방식의. 섹션 VII는 미래의 작업 및 기타 기회를 결론짓고 강조합니다.

II. STATE OF THE ART AND PRACTICE (AND DERIVATION OF AKM TOOL REQUIREMENTS)

II. 최신 기술 및 실습(및 AKM 도구 요구 사항의 도출)

Since an inaugural workshop held in Groningen, NL in. 2004, the software architecture community has advanced the state of the art in AKM significantly [1]. Some success with. decision guidance modeling has been reported e.g. for. enterprise applications and service-oriented architectures [27]; however, it is not a widely adopted practice yet..

네덜란드 흐로닝언에서 열린 첫 워크숍 이후. 2004년, 소프트웨어 아키텍처 커뮤니티는 AKM의 최신 기술을 크게 발전시켰습니다[1]. 약간의 성공. 의사 결정 지침 모델링이 보고되었습니다. 엔터프라이즈 애플리케이션 및 서비스 지향 아키텍처[27]; 그러나 아직 널리 채택된 관행은 아닙니다..

To avoid unnecessary overlap with previous publications from the community, we structure this section into a comparison of seven templates that have been reported to be actively used in practice (Section II.A), briefly revisit research prototype tools from other researchers (II.B), and then summarize our previous work on the subject (II.C, II.D). Finally, we derive requirements for design and implementation of ADMentor from the state of the art and the practice.

커뮤니티의 이전 간행물과의 불필요한 중복을 피하기 위해 이 섹션을 실제로 활발히 사용되는 것으로 보고된 7개의 템플릿을 비교하고(섹션 II.A), 다른 연구자의 연구 프로토타입 도구를 간략하게 재검토하고(II.B), 해당 주제에 대한 이전 작업을 요약합니다(II.C, II.D). 마지막으로, 우리는 최신 기술과 실습에서 ADMentor의 설계 및 구현에 대한 요구 사항을 도출합니다.

A. ISO/IEC/IEEE 42010 and Practitioner Templates

A. ISO/IEC/IEEE 42010 및 실무자 템플릿

Table 1 compares the guidelines in IEEE 42010 (and its) companion template available under a Creative Commons License) with six other Architectural Decision (AD) capturing.

표 1은 IEEE 42010(및 크리에이티브 커먼즈 라이선스에 따라 사용할 수 있는 동반 템플릿)의 지침을 6개의 다른 AD(Architectural Decision) 캡처와 비교합니다.

templates used in industry, ordered by their age: IBM UMF [25], the Tyree/Akerman template [22], a key decisions template from Bredemeyer Consulting [6], M. Nygard's ADR blog post [14], an arc42 resource by Hruschka/Starke [7], and our own Y-statements [24]. We selected these seven templates due to their public accessibility and their know uses on industry. projects. The selection is not complete, but representative.'.

IBM UMF [25], Tyree/Akerman 템플릿 [22], Bredemeyer Consulting의 주요 의사 결정 템플릿 [6], M. Nygard의 ADR 블로그 게시물 [14], Hruschka/Starke의 arc42 리소스 [7] 및 자체 Y-문 [24]. 우리는 공개적인 접근성과 업계에서의 알려진 용도 때문에 이 7가지 템플릿을 선택했습니다. 프로젝트. 선택은 완전하지는 않지만 대표적입니다.'.

The comparison in the table shows that there are many. formats, with consensus about the core attributes/aspects (e.g., AD outcome and why-justification), but significant variability. regarding traceability links and other types of attributes/aspects (e.g., status and owner of decision). The number of template. attributes/aspects varies; hence, the effort to fill out varies too. IBM UMF and Tyree/Akerman are the most comprehensive. templates; Nygard's ADRs, arc42 and Y-statements appear at the other end of the spectrum. Note that not all of the attributes/aspects are needed when predicting future decisions in decision guidance models (e.g., outcome, status); a generalized, forward-looking subset is enough (e.g., decision drivers, options to be considered). We can conclude that tools for decision capturing and sharing should be flexible and configurable to accommodate use of these (or other) templates.

표의 비교는 많은 것을 보여줍니다. 핵심 속성/측면(예: AD 결과 및 이유-정당화)에 대한 합의가 있지만 상당한 가변성이 있는 형식입니다. 추적성 링크 및 기타 유형의 속성/측면(예: 상태 및 결정 소유자)에 관하여. 템플릿의 수입니다. 속성/측면은 다양합니다. 따라서 작성하려는 노력도 다양합니다. IBM UMF와 Tyree/Akerman이 가장 포괄적입니다. 템플릿; Nygard의 ADR, arc42 및 Y-문은 스펙트럼의 반대쪽 끝에 나타납니다. 의사 결정 안내 모델에서 미래 결정을 예측할 때 모든 속성/측면이 필요한 것은 아닙니다(예: 결과, 상태). 일반화되고 미래 지향적인 하위 집합(예: 의사 결정 동인, 고려해야 할 옵션)이면 충분합니다. 의사 결정 캡처 및 공유를 위한 도구는 이러한 (또는 다른) 템플릿의 사용을 수용할 수 있도록 유연하고 구성 가능해야 한다는 결론을 내릴 수 있습니다.

B. Research Tools (Brief Recapitulation)

B. 연구 도구(간략한 요약)

AREL [21] is an add-in to Sparx Enterprise Architect, like. Decision Architect (see Section II.C below) and ADMentor (described in this paper). Like most AKM tools, AREL focuses. on decision rationale capturing. AREL defines a UML profile,. but does not mandate the decision capturing attributes. Hence,. our work can be seen as a continuation and extension of the research around AREL.

AREL [21]은 Sparx Enterprise Architect의 추가 기능입니다. Decision Architect(아래 섹션 II.C 참조) 및 ADMentor(이 백서에 설명됨). 대부분의 AKM 도구와 마찬가지로 AREL은 중점을 둡니다. 결정 근거 캡처에 대해. AREL은 UML 프로필을 정의합니다. 그러나 속성을 캡처하는 결정을 의무화하지는 않습니다. 따라서. 우리의 작업은 AREL을 둘러싼 연구의 연속이자 확장으로 볼 수 있습니다.

We refer the reader to Chapter 6 in [1] and two recent. conference publications, a requirements-based tool evaluation [2] and a systematic literature review [24] for information about other AKM/AD tools; recent additions to the portfolio. include ADvISE and SAW. Most of the existing tools focus on decision capturing, not on decision guidance; such tools do not. answer our two research questions from Section I satisfyingly..

우리는 독자에게 [1]의 6장과 최근의 두 장을 참조하도록 합니다. 다른 AKM/AD 도구에 대한 정보에 대한 컨퍼런스 간행물, 요구 사항 기반 도구 평가[2] 및 체계적인 문헌 검토[24]; 포트폴리오에 최근 추가되었습니다. ADvISE 및 SAW를 포함합니다. 기존 도구의 대부분은 의사 결정 지침이 아닌 의사 결정 캡처에 중점을 둡니다. 이러한 도구는 그렇지 않습니다. 섹션 I의 두 가지 연구 질문에 만족스럽게 답하십시오..

C. Decision Architect (Documentation Viewpoints)

C. 의사 결정 설계자(문서 관점)

The work reported in this paper is connected to the. formerly published Decision Architect [11], an add-in for Sparx Enterprise Architect implementing a conceptual framework for architecture decision documentation based on ISO/IEC/IEEE 42010. The framework consists of five decision viewpoints: relationship, chronology, stakeholder, forces and detailed viewpoint. With the implementation as an add-in to an. UML editor, it is possible to link architectural decisions to UML model elements thereby realizing traceability. In [11], we reported the application of Decision Architect by five software architects with good feedback..

이 논문에 보고된 작업은 다음과 관련이 있습니다. 이전에 ISO/IEC/IEEE 42010을 기반으로 아키텍처 의사 결정 문서를 위한 개념적 프레임워크를 구현하는 Sparx Enterprise Architect용 추가 기능인 Decision Architect [11]를 게시했습니다. 프레임워크는 관계, 연대기, 이해관계자, 힘 및 세부 관점의 다섯 가지 결정 관점으로 구성됩니다. UML 편집기에 대한 추가 기능으로 구현을 사용하면 아키텍처 결정을 UML 모델 요소에 연결하여 추적성을 실현할 수 있습니다. [11]에서는 5명의 소프트웨어 아키텍트가 Decision Architect를 적용한 것을 좋은 피드백으로 보고했습니다.

For the latest version 0.5, released as open source in September 2014, the Decision Architect features re-designed forces and detailed viewpoints and numerous bug fixes. In contrast to ADMentor as introduced in this paper, Decision Architect is meant mainly for documentation purposes, but not for architectural guidance: there is no separation between problem and solution spaces, and it is not possible to import generic problem domain models. However, Decision Architect and ADMentor can be connected within Enterprise Architect to allow both detailed documentation and architectural guidance.

2014년 9월에 오픈 소스로 출시된 최신 버전 0.5의 경우 Decision Architect는 재설계된 힘과 상세한 관점 및 수많은 버그 수정을 제공합니다. 이 백서에서 소개한 ADMentor와 달리 Decision Architect는 주로 문서화 목적을 위한 것이지만 아키텍처 지침은 아닙니다. 그러나 Decision Architect와 ADMentor는 Enterprise Architect 내에서 연결하여 자세한 문서와 아키텍처 지침을 모두 허용할 수 있습니다.

AD aspect (attribute)	IEEE 42010 (Template V2.2)	IBM UMF ADTable	Tyree/ Akerman	Bredemeyer Key Decisions	Nygard ADRs	arc42 Hruschka/Starke	Y-Statements (ABB, [24])
ID	Unique Identifier	ID	(in D-Header)	/	(part of name)	(Section #)	(Id)
Outcome	Statement of the decision	Decision (Made)	Decision	Approach	Decision	Decision	we decided for
Requirements trace (FRs, NFRs)	Correspondence or linkage to concerns	(Derived requirements)	Related requirements	Business drivers, technical	/		/
Accountability (Role,Person)	Owner of the decision			drivers /			
Software architecture	Correspondence or linkage to		Related artifacts				In the context of
viewpointtrace Why-answers	elements Rationale (linked entity)	Justification	Argument	Conclusion		(Question under Decision)	(optional "because"half
Decisiondrivers	Forces, constraints		(Constraints)	Benefits, Drawbacks	Context	Constraints	sentence) facing
Assumptions	Assumptions	Assumptions	Assumptions	/	/	Assumptions	
Options	Considered Alternatives	Alternatives	Positions	/	/	Considered Alternatives	and neglected
Problem	/	Issue or Problem	Issue	/	/	Problem	/
Decision dependencies	(not in template, but in standard)	Related decisions	Related decisions	/	/	/	/
Categorization, classification	/	Subject Area, Topic	Group(ing)	/	/	(Decision Topic)	/
Name	/	Name	(in D-Header)	<>	Title	(Section heading)	/
StateofAD making	not in template, but in standard	(not in published example)	Status	/	Status		
Impact	not in template, but in standard	Implications	Implications	Issues/ Considerations	Consequences	/	to achieve, accepting that

Otherentries	Timestamps, Citations	Motivation	Notes, Related principles	Notes, Drivers realized	/	/	
Element count	9 (template), 11 (standard)	13	14	9 (plus 1-2 in header)	5	5(with 14 questions)	6
Scoping help (whichADs to capture?)	Yes	(not in table template, not published)	(anecdotal In article)	2001white paper	/	(ASRs mentioned)	
Size (page or word limit) or other hints		not published	(example is half a page, table form)	/	1-2 pages per ADR	to be ordered by importance	1 (long) sentence per Y-statement
Publication year	2011	1998(internal)	2005	2005	2011	2012	2012

D. SOAD and SDA (2006-2011)

D. SOAD 및 SDA (2006-2011)

The SOA Decision (SOAD) Modeling project2 [28] and Solution Decision Advisor (SDA) tool [13] addressed similar research problems as we do here, but chose different solutions both on the conceptual and on the technical level: for instance,. options were modelled as child elements of problems, whereas ADMentor sees options as first class model elements; multiple. options occurrences can be chosen per problem occurrence. Furthermore, the metamodel of ADMentor is not hardcoded or fixed otherwise, but extensible via typed meta-information attributes/annotations (see Section III). Moreover, ADMentor defines a UML Profile (see Section IV) and leverages Sparx Enterprise Architect as a UML and general modeling platform.

SOAD(SOAD) 모델링 프로젝트2 [28] 및 SDA(Solution Decision Advisor) 도구 [13]는 여기에서 하는 것과 유사한 연구 문제를 다루었지만 개념적 수준과 기술적 수준 모두에서 다른 솔루션을 선택했습니다. 옵션은 문제의 하위 요소로 모델링된 반면 ADMentor는 옵션을 일류 모델 요소로 봅니다. 배수. 옵션 발생은 문제 발생별로 선택할 수 있습니다. 또한 ADMentor의 메타 모델은 하드코딩되거나 수정되지 않지만 유형화된 메타 정보 속성/주석을 통해 확장할 수 있습니다(섹션 III 참조). 또한 ADMentor는 UML 프로필(섹션 IV 참조)을 정의하고 Sparx Enterprise Architect를 UML 및 일반 모델링 플랫폼으로 활용합니다.

Derivation of requirements. Let us now establish additional requirements for ADMentor.

요구 사항의 도출. 이제 ADMentor에 대한 추가 요구 사항을 설정해 보겠습니다.

Functional Requirements (FRs)(FRs) . Tools for decision capturing and sharing must support:

기능 요구 사항 (FRs)(FRs) . 의사 결정 캡처 및 공유를 위한 도구는 다음을 지원해야 합니다.

A user interface applying the master-details pattern so that both the big picture and the nuts and bolts of. individual problems and options can be portrayed. Rich text editing (e.g., URIs, bullet lists, emphasis on. certain words with italic and bold fonts, headings).. Powerful model refactoring capabilities. Semantic queries (e.g., returning all problems to be solved for particular milestone or gate). Reporting, e.g. RTF/PDF/HTML exports that can be customized for project stakeholders that are involved in the decision making, but do not work with the tool.

마스터-디테일 패턴을 적용하여 큰 그림과 너트와 볼트를 모두 적용하는 사용자 인터페이스입니다. 개별 문제와 옵션을 묘사할 수 있습니다. 서식 있는 텍스트 편집(예: URI, 글머리 기호 목록, 기울임꼴 및 굵은 글꼴이 있는 특정 단어, 제목 강조).. 강력한 모델 리팩토링 기능. 의미론적 쿼리(예: 특정 이정표 또는 게이트에 대해 해결해야 할 모든 문제 반환). 보고, 의사 결정에 관여하지만 도구와 함께 작동하지 않는 프로젝트 이해 관계자를 위해 사용자 정의할 수 있는 e.g. RTF/PDF/HTML 내보내기.

An API for semi-automatic model creation/updating and tool integration.

반자동 모델 생성/업데이트 및 도구 통합을 위한 API입니다.

FRs for AKM tools, both for retrospective and for proactive. methods and tools, have also been published in [2] (and other literature referenced in Section II.A to II.D)..

AKM 도구에 대한 FR(회고적 및 사전 예방적 모두). 방법 및 도구는 [2](및 섹션 II.A에서 II.D까지 참조된 기타 문헌)에도 게시되었습니다.

Other success criteria. In our opinion, mere architectural decision trees resembling Unified Modeling Language (UML) visualizations are valuable, but not sufficient to represent the nature of design knowledge adequately. Successful decision capturing and sharing requires a combination of text processing. capabilities (for a powerful, but still lean knowledge authoring). and graph-oriented visualizations (e.g., of dependencies)..

다른 성공 기준. 우리의 의견으로는 UML(Unified Modeling Language) 시각화와 유사한 단순한 아키텍처 의사 결정 트리는 가치가 있지만 설계 지식의 특성을 적절하게 나타내기에는 충분하지 않습니다. 성공적인 의사 결정 캡처 및 공유를 위해서는 텍스트 처리의 조합이 필요합니다. 기능(강력하지만 여전히 간결한 지식 작성용). 그래프 지향 시각화(예: 종속성)..

A conceptual integration into both agile and more conservative design process practices also is required. This can be achieved with rich meta-information attribution/annotations with appropriate default values to minimize work (also including a confidentiality flag).

민첩한 설계 프로세스 관행과 보다 보수적인 설계 프로세스 관행 모두에 대한 개념적 통합도 필요합니다. 이는 작업을 최소화하기 위해 적절한 기본값(기밀성 플래그도 포함)을 사용하여 풍부한 메타 정보 속성/주석을 사용하여 달성할 수 있습니다.

From the related work analysis and comparison of practiced approaches (template analysis in Section II.A), we can conclude that even when using light templates (e.g., Nygard's ADRs, arc42 pages, or Y-statements), decision documentation remains a lot of (typically unwelcome) work. AKM/AD tools like Decision Architect help, but do not

reduce the effort sufficiently. According to our experience and community feedback, a tool cannot dictate a single set of attributes (knowledge structure), but must be flexible and accommodate a variety of architectural capturing and sharing styles in a bestof-breed manner.

관련 작업 분석 및 실천된 접근 방식의 비교(섹션 II.A의 템플릿 분석)를 통해 가벼운 템플릿(예: Nygard의 ADR, arc42 페이지 또는 Y-문)을 사용하는 경우에도 의사 결정 문서화는 여전히 많은 (일반적으로 달갑지 않은) 작업으로 남아 있다는 결론을 내릴 수 있습니다. Decision Architect와 같은 AKM/AD 도구는 도움이 되지만 노력을 충분히 줄이지는 않습니다. 우리의 경험과 커뮤니티 피드백에 따르면 도구는 단일 속성 세트(지식 구조)를 지시할 수 없지만 유연해야 하며 동급 최고의 방식으로 다양한 아키텍처 캡처 및 공유 스타일을 수용해야 합니다.

Before we introduce the contribution of this paper, let us now establish a running example indicating that many architectural decisions recur indeed. This example will serve as an exemplary instantiation of our concepts later in the paper.

이 논문의 기여를 소개하기 전에 이제 많은 아키텍처 결정이 실제로 반복된다는 것을 나타내는 실행 사례를 설정해 보겠습니다. 이 예는 논문 뒷부분에서 개념의 예시적인 인스턴스 역할을 할 것입니다.

Modelling and usage example. An example of a recurring design issue when moving a Web application to a cloud is session state management (e.g., think of a shopping session in an online store). The three top-level design options (patterns) are client session state, server session state and database session state [4]. Client session state scales well, but has security and possibly performance problems; this does not change when moving to a cloud platform. Server session state uses main memory or proprietary data stores in an application server (e.g., an HTTP session in a JEE servlet container); this approach is no longer recommended when deploying to a cloud due to scalability and reliability concerns. Finally, database session state is well supported in many clouds, e.g. via highly scalable key-value storages (NoSQL).

모델링 및 사용 예. 웹 애플리케이션을 클라우드로 이동할 때 반복되는 설계 문제의 예로는 세션 상태 관리(예: 온라인 상점의 쇼핑 세션 생각)가 있습니다. 세 가지 최상위 디자인 옵션(패턴)은 클라이언트 세션 상태, 서버 세션 상태 및 데이터베이스 세션 상태입니다[4]. 클라이언트 세션 상태는 잘 확장되지만 보안 및 성능 문제가 있을 수 있습니다. 이는 클라우드 플랫폼으로 이동할 때 변경되지 않습니다. 서버 세션 상태는 애플리케이션 서버의 주 메모리 또는 독점 데이터 저장소(예: JEE 서블릿 컨테이너의 HTTP 세션)를 사용합니다. 이 접근 방식은 확장성 및 안정성 문제로 인해 클라우드에 배포할 때 더 이상 권장되지 않습니다. 마지막으로, 데이터베이스 세션 상태는 확장성이 뛰어난 키-값 스토리지(NoSQL)와 같은 많은 클라우드에서 잘 지원됩니다.

If session management is a requirement, this decision has to. be made or revisited when designing a cloud-native application. or when migrating a Web-based application to the cloud; while the decision outcomes may vary, the issues and options to be. considered stay the same (i.e., they recur). Multiple instances of this decision may exist per project (e.g., in multi-channel. applications that serve different user groups)..

세션 관리가 요구 사항인 경우 이 결정이 필요합니다. 클라우드 네이티브 애플리케이션을 설계할 때 만들거나 다시 검토할 수 있습니다. 또는 웹 기반 애플리케이션을 클라우드로 마이그레이션할 때; 결정 결과는 다를 수 있지만 문제와 옵션은 다양합니다. 동일한 상태로 간주됩니다(즉, 반복). 이 결정의 여러 인스턴스가 프로젝트당 존재할 수 있습니다(예: 다중 채널에서). 서로 다른 사용자 그룹에 서비스를 제공하는 애플리케이션).

III. CONCEPTUAL DESIGN OF ADMENTOR

III. ADMENTOR의 개념 설계

This section presents our research contributions and is. organized into six steps: (A) metamodelling and problem space creation, (B) problem space modeling, (C) meta-information annotation, (D) tailoring, (E) solution space creation and (F). solution space usage (decision backlog management)..

이 섹션에서는 우리의 연구 기여를 제시합니다. (A) 메타모델링 및 문제 공간 생성, (B) 문제 공간 모델링, (C) 메타 정보 주석, (D) 조정, (E) 솔루션 공간 생성 및 (F)의 6단계로 구성됩니다. 솔루션 공간 사용량(의사 결정 백로그 관리)..

A. Metamodeling and Problem Space Creation

A. 메타모델링 및 문제 공간 생성

Figure 1 specifies our AKM model structuring. A fourquadrant design space structure is shown in the bottom half of the figure (in the last two table rows and columns). The upper part of the figure elaborates on the semantic differences between problem spaces and solution spaces, i.e., their different reach (i.e., scope and lifetime), owner role (i.e., model creator and maintainer), and purpose.

그림 1은 AKM 모델 구조를 지정합니다. 4사분면 설계 공간 구조는 그림의 아래쪽 절반(마지막 두 테이블 행과 열)에 표시됩니다. 그림의 상단 부분은 문제 공간과 솔루션 공간 간의 의미론적 차이, 즉 서로 다른 범위(즉, 범위 및 수명), 소유자 역할(즉, 모델 생성자 및 유지 관리자) 및 목적에 대해 자세히 설명합니다.

Model Type	**Problem Space**	**Solution Space**
Reach/Level	Asset (Community)	Project
Owner	Knowledge Engineer	Software Architect
Purpose	Design Guidance	Decision (Back-)Log
Need for Architectural Decision	raises, Problem, instantiates, Problem Occurrence	
Design Candidates	addressed by, raises, Option, supports, ..., instantiates, Option Occurrence	

Fig. 1. Problem/solution space model elements and their link types
그림 1. 문제점/솔루션 공간 모델 요소 및 해당 링크 유형

Problems are addressed by options. Problems and options can raise (i.e., lead to) further problems that need to be solved. Recurring problems can be instantiated one or more times in a solution space; option occurrences instantiate options (also one or more times). Additional, self-explanatory link types connecting options, not shown in Figure 1, are: suggests, conflicts with, and bound to. In our session state management example from above, there might be two problems, a) the conceptual decision which pattern to use (with the three state management patterns from above modeled as options) and b) a technology-level decision which storage medium to use for database session state (with options like relational MySQL database and MongoDB key-value store, among others). The database session state option of the conceptual pattern selection decision raises the technology-level decision about storage.

문제는 옵션으로 해결됩니다. 문제와 옵션은 해결해야 할 추가 문제를 야기(즉, 이끌어낼 수 있음)을 유발할 수 있습니다. 반복되는 문제는 솔루션 공간에서 한 번 이상 인스턴스화될 수 있습니다. 옵션 발생은 옵션을 인스턴스화합니다(또한 한 번 이상 포함). 그림 1에 표시되지 않은 추가 설명이 필요 없는 링크 유형 연결 옵션은 제안, 충돌 및 바인딩입니다. 위의 세션 상태 관리 예에서는 a) 사용할 패턴의 개념적 결정(위의 세 가지 상태 관리 패턴이 옵션으로 모델링됨)과 b) 데이터베이스 세션 상태에 사용할 스토리지 매체의 기술 수준 결정(관계형 MySQL 데이터베이스 및 MongoDB 키-값 저장소와 같은 옵션 포함)의 두 가지 문제가 있을 수 있습니다. 다른 사람들 중에서). 개념적 패턴 선택 결정의 데이터베이스 세션 상태 옵션은 스토리지에 대한 기술 수준 결정을 제기합니다.

This approach is faithful to our earlier vision of architectural decision modeling with reuse, but also different and more advanced: we model problem spaces and solution spaces separately, and provide two abstractions each, problem and solution (yielding four structural elements instead of three). We also define novel link types connecting the four elements. We decided to define semantically rich links that can be used in queries etc., inspired by existing work in the AKM community (see Section II) and by REST maturity level three [5], which promotes hypermedia controls as/for typed link relations.

이 접근 방식은 재사용을 통한 아키텍처 의사 결정 모델링에 대한 초기 비전에 충실하지만, 문제 공간과 솔루션 공간을 별도로 모델링하고 문제와 솔루션이라는 두 가지 추상화를 각각 제공합니다(3개가 아닌 4개의 구조 요소 생성). 또한 네 가지 요소를 연결하는 새로운 링크 유형을 정의합니다. 우리는 AKM 커뮤니티의 기존 작업(섹션 II 참조)과 형식화된 링크 관계에 대한 하이퍼미디어 제어를 촉진하는 REST 성숙도 수준 3[5]에서 영감을 받아 쿼리 등에 사용할 수 있는 의미론적으로 풍부한 링크를 정의하기로 결정했습니다.

The following Figure 2 (on the next page) shows the six. processing steps dealing with the elements from the quadrants from the bottom half of Figure 1. The processing steps are.

다음 그림 2(다음 페이지)는 6개를 보여줍니다. 그림 1의 아래쪽 절반에서 사분면의 요소를 처리하는 처리 단계. 처리 단계는 다음과 같습니다.

named A to F; they are associated with the owner roles from Figure 1, knowledge engineer and software architect (e.g., solution or product architect).

A에서 F까지의 이름; 이는 그림 1의 소유자 역할, 지식 엔지니어 및 소프트웨어 설계자(예: 솔루션 또는 제품 설계자)와 연관됩니다.

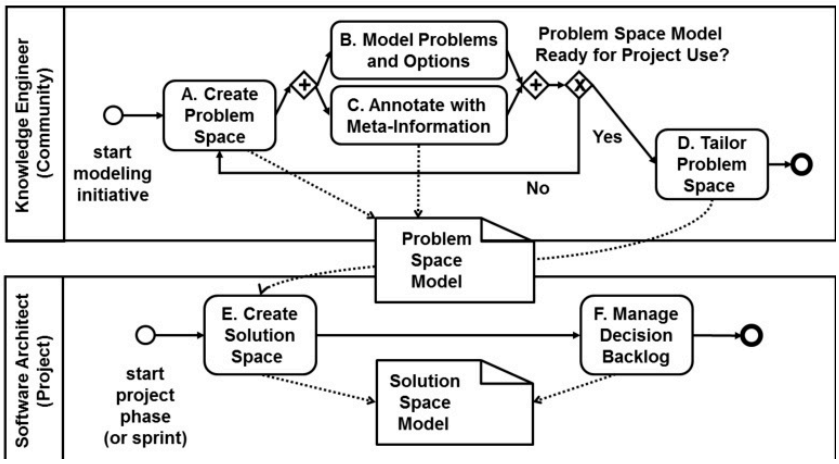

Fig. 2. Workflow for ADMentor users (notation: BPMN)
그림 2. ADMentor 사용자를 위한 워크플로(표기법: BPMN)

The remainder of the section follows the steps B to F from Figure 2, and specifies the Create, Read, Update, Delete (CRUD) operations that are performed on instances of the four structural elements from Figure 1. Feedback from solution space owners (on projects) to community-level knowledge engineers is out of scope of this paper; see [27] for a suggestion how to organize a continuous review-update loop.

이 섹션의 나머지 부분에서는 그림 2의 B-F 단계를 따르고 그림 1의 네 가지 구조 요소 인스턴스에 대해 수행되는 CRUD(Create, Read, Update, Delete) 작업을 지정합니다. 솔루션 공간 소유자(프로젝트에 대해)에서 커뮤니티 수준의 지식 엔지니어에 대한 피드백은 이 백서의 범위를 벗어납니다. 지속적인 검토-업데이트 루프를 구성하는 방법에 대한 제안은 [27]을 참조하십시오.

B. Problem Space Modelling (Problems, Options)

B. 문제 공간 모델링(문제, 옵션)

The second modeling step after problem space creation is. problem space modeling. Problem and option model elements are created and positioned into knowledge packages here..

문제 공간 생성 후 두 번째 모델링 단계는 다음과 같습니다. 문제 공간 모델링. 문제 및 옵션 모델 요소는 여기에서 생성되어 지식 패키지에 배치됩니다..

Rich text support is one of the requirements we identified in Section II for instance, the knowledge aspects appearing in the practitioner templates from Section II can be represented as rich text sections separated by headings. As motivated in Sections I and II, we do not want to dictate any particular decision capturing template (for the

description of problems, options, and their occurrences). AKM usage and maintenance should be lean; hence, ADMentor encourages knowledge engineers not to copy much text into problem spaces, but to leverage the Web. Hence, Web links e.g. to pattern texts can be added in this step, and other model elements can be linked in.

리치 텍스트 지원은 섹션 II에서 식별한 요구 사항 중 하나이며, 예를 들어 섹션 II의 실무자 템플릿에 나타나는 지식 측면은 제목으로 구분된 리치 텍스트 섹션으로 표현될 수 있습니다. 섹션 I 및 II에서 동기를 부여한 대로 특정 의사 결정 캡처 템플릿(문제, 옵션 및 발생에 대한 설명을 위해)을 지시하고 싶지 않습니다. AKM 사용 및 유지 관리는 간결해야 합니다. 따라서 ADMentor는 지식 엔지니어가 문제 공간에 많은 텍스트를 복사하지 말고 웹을 활용하도록 권장합니다. 따라서 패턴 텍스트와 같은 웹 링크는 이 단계에서 추가될 수 있으며 다른 모델 요소는 링크될 수 있습니다.

With these concepts and standard UML tool extensibility features, QOC diagram support' almost comes for free: questions (Q) and options (O) are modelled with ADMentor, and criteria (C) are assumed to be supported by its host tool (here: Sparx Enterprise Architect, see Section IV), just like components and connectors in UML class or component diagrams. We added QOC diagram support rather late in our iterative development work, which demonstrates that such extensions and configurations are indeed feasible. Only minor additional customization effort had to be invested in the host tool (i.e., the naming of link types via UML stereotypes).

이러한 개념과 표준 UML 도구 확장성 기능을 통해 QOC 다이어그램 지원은 거의 무료로 제공됩니다: 질문(Q)과 옵션(O)은 ADMentor로 모델링되고 기준(C)은 UML 클래스 또는 구성 요소 다이어그램의 구성 요소 및 커넥터와 마찬가지로 호스트 도구(여기: Sparx Enterprise Architect, 섹션 IV 참조)에서 지원되는 것으로 가정됩니다. 우리는 반복적인 개발 작업에서 다소 늦게 QOC 다이어그램 지원을 추가했는데, 이는 이러한 확장 및 구성이 실제로 실현 가능하다는 것을 보여줍니다. 호스트 도구에는 약간의 추가 사용자 정의 노력만 투자해야 했습니다(즉, UML 스테레오타입을 통한 링크 유형의 이름 지정).

C. Meta-Information Annotation (a.k.a. Typed Tagging)

C. 메타 정보 주석(일명 유형 태깅)

Table 2 specifies the output of knowledge engineering activity B, annotate problems and options with ADK metainformation. To compile the meta-information annotations in the table, we reviewed the software architecture and method engineering literature and reflected on own project experiences (in software architect and project management roles):

표 2는 지식 엔지니어링 활동 B의 출력을 지정하고 ADK 메타 정보로 문제 및 옵션에 주석을 달 수 있습니다. 표의 메타 정보 주석을 컴파일하기 위해 소프트웨어 아키텍처 및 방법 엔지니어링 문헌을 검토하고 자신의 프로젝트 경험(소프트웨어 설계자 및 프로젝트 관리 역할)을 반영했습니다.

Name	Purpose, Rationale	Sample Value(s)
Intellectual PropertyRights	Intellectual Property Rights (IPR) for model element, e.g. confiden- tiality level, copyright statement	Public, Company- Confidential, Company X, 2015
Knowledge Provenance	Reference to a cited source and/or acknowledgment of contributor	CCPbook,PoEAA website, Project Y, ArchitectZ
Refinement Level	The abstractionlevelonwhich this problem typically occurs	Conceptual Level, Technology Level
ProjectStage	Gate,milestone,phase and/or elaboration point in incremental anditerative design(inwhichthis problem is typically tackled)	Inception, Elaboration, Construction (in OpenUP [15])
Organizational Reach	Sphere of influenceof the problem	Enterprise, Division, Business Unit, Project, Subsystem
OwnerRole	The role (as defined e.g.in OpenUP) that is responsible and accountable for the decision	Application Architect, Integration Architect
Stakeholder Roles	Peoplewith aninterest in this problem (note: the accountable person is annotated as owner role)	Enterprise Architects,Frontend Developers, Testers
Viewpoint(s)	e.g. one of the 4+1 views on software architecture or a Rozanski/Woodsviewpoint[16]	Logical Viewpoint, Deployment Viewpoint

Our compilation of meta-information does not claim to be. complete; knowledge engineers can add and remove metainformation as desired within their community contexts (e.g., to streamline their problem space modeling activities). Activities D and F leverage this meta-information, which we. implemented with UML Tagged Values (see Section IV).

우리의 메타 정보 편집은 그렇게 주장하지 않습니다. 완성하다; 지식 엔지니어는 커뮤니티 컨텍스트 내에서 원하는 대로 메타정보를 추가 및 제거할 수 있습니다(예: 문제 공간 모델링 활동을 간소화하기 위해). 활동 D와 F는 이 메타 정보를 활용합니다. UML 태그 값으로 구현됩니다(섹션 IV 참조).

*D*D Tailoring (from Problem Space to Solution Space)

*D*D 조정(문제 공간에서 솔루션 공간으로)

In the tailoring step, a problem space is trimmed down to. the problems and options that are relevant for a particular. project (context-specific filtering). The meta-information from activity C can support this filtering work; for instance, a role-. or phase-specific problem space can be obtained this way..

재봉 단계에서 문제 공간은 다음으로 잘립니다. 특정 문제와 관련된 문제와 옵션. 프로젝트(컨텍스트별 필터링)를 사용합니다. 활동 C의 메타 정보는 이 필터링 작업을 지원할 수 있습니다. 예를 들어 역할-입니다. 또는 위상별 문제 공간을 이 방법으로 얻을 수 있습니다..

This activity is not described in detail in this paper. Support for it requires tool engineering rather than design science work.

이 활동은 이 백서에서 자세히 설명되지 않습니다. 이를 지원하려면 설계 과학 작업보다는 도구 엔지니어링이 필요합니다.

E. Solution Space Creation

E. 솔루션 공간 창출

The following two user stories characterize this step:

다음 두 가지 사용자 스토리는 이 단계를 특징짓습니다.

Full copy/complete instantiation story: "As a solution architect starting a project, I would like to create a fully populated solution space containing one open problem occurrence for each problem and one option occurrence for each option that came out of the. tailoring so that I receive guidance for my design work and I do not forget to solve any problems.". On demand instantiation story: "As a solution architect, I would also like to be able to start with an empty solution space and create problem occurrences and option occurrences individually as needed during the project so that my decision log has a minimal size'.

전체 복사/전체 인스턴스화 스토리: "프로젝트를 시작하는 솔루션 아키텍트로서 각 문제에 대해 하나의 미해결 문제 발생과 각 옵션에 대해 하나의 옵션 발생을 포함하는 완전히 채워진 솔루션 공간을 만들고 싶습니다. 디자인 작업에 대한 지도를 받고 문제를 해결하는 것을 잊지 않도록 재봉합니다.". 주문형 인스턴스화 스토리: "솔루션 아키텍트로서 빈 솔루션 공간으로 시작하여 프로젝트 중에 필요에 따라 문제 발생 및 옵션 발생을 개별적으로 생성하여 의사 결정 로그의 크기를 최소화할 수 있기를 원합니다."

This activity also is straightforward to implement in tools.

이 활동은 도구에서 구현하기도 간단합니다.

F. Decsion Backlog Management (Solution Space Usage)

F. Decsion 백로그 관리(솔루션 공간 사용량)

Semantically rich meta-information as defined in activity C can be leveraged to search, filter and order solution space models in activity F. When combined with context and state information, a powerful and user-friendly representation of the solution space results - a decision backlog [9]. Architects do not have to follow any predefined decision making order (most architects probably would not do so anyway); they simply pick the decision backlog entries they deem to be particularly urgent (and decidable) in/for the current design iteration. Modeling tools can support such decision backlog elegantly in views and widgets with table layouts that are configurable w.r.t. filtering. and ordering. Table 3 sketches such decision backlog view:

활동 C에 정의된 의미론적으로 풍부한 메타 정보를 활용하여 활동 F에서 솔루션 공간 모델을 검색, 필터링 및 정렬할 수 있습니다. 컨텍스트 및 상태 정보와 결합하면 솔루션 공간에 대한 강력하고 사용자 친화적인 표현, 즉 의사 결정 백로그가 발생합니다[9]. 건축가는 미리 정의된 의사 결정 순서를 따를 필요가 없습니다(대부분의 건축가는 어쨌든 그렇게 하지 않을 것입니다). 그들은 단순히 현재 설계 반복에서 특히 긴급한(그리고 결정 가능한) 것으로 간주되는 결정 백로그 항목을 선택합니다. 모델링 도구는 필터링을 통해 구성 가능한 테이블 레이아웃이 있는 뷰 및 위젯에서 이러한 의사 결정 백로그를 우아하게 지원할 수 있습니다. 그리고 주문. 표 3은 이러한 의사 결정 백로그 보기를 스케치합니다.

Problem Occurrence	Status	Viewpoint	Owner Role	Comple- xity	
SessionState Management Occurrence1	Decided	Functional	Web architect	High	
Session Database Provider Occurrence1	Open	Information	Data Architect	Medium	

The problem state (on the occurrence level) is aggregated from the associated option occurrence's states in the following. way: A problem occurrence is in state open if all options are eligible; it is in state not applicable if all options are neglected, and in state decided if all options are either chosen or neglected; it is partially decided in all other cases (additional option states are tentative and challenged). Just like product backlogs in agile practices, the decision backlog never has to be emptied completely, as design time always is constrained.

문제 상태(발생 수준)는 다음과 같은 관련 옵션 발생의 상태에서 집계됩니다. way: 모든 옵션이 적합한 경우 문제 발생이 열린 상태입니다. 모든 옵션이 무시되면 적용되지 않는 상태이며, 모든 옵션이 선택되거나 무시되는지 결정된 상태입니다. 다른 모든 경우에는 부분적으로 결정됩니다(추가 옵션 상태는 잠정적이고 이의가 제기됨). 애자일 관행의 제품 백로그와 마찬가지로 설계 시간이 항상 제한되어 있기 때문에 의사 결정 백로그를 완전히 비울 필요가 없습니다.

Transition to technology level. The concepts introduced in this section are not specific any host platform, but can be realized in any extensible modelling tool. The following. Section IV transcends from platform-independent concepts to one particular platform-specific design and its implementation.

기술 수준으로의 전환. 이 섹션에 소개된 개념은 호스트 플랫폼에만 국한되지 않지만 확장 가능한 모델링 도구에서 실현할 수 있습니다. 다음은 다음과 같습니다. 섹션 IV는 플랫폼 독립적인 개념에서 하나의 특정 플랫폼별 설계 및 구현으로 초월합니다.

IV. IMPLEMENTATION OF ADMENTOR (VALIDATION PHASE 1)

IV. ADMENTOR 구현(검증 1단계)

ADMentor is an add-in to Sparx Enterprise Architect (EA) Version 10 (and higher). It is implemented in C#C# and uses .NET Version 4.5. ADMentor comes with an UML Profile and a MDG Technology (two extensibility concepts in Sparx EA) that carry Architectural Knowledge Management (AKM) semantics implementing the concepts from Section III.

ADMentor는 Sparx EA(Enterprise Architect) 버전 10 이상에 대한 추가 기능입니다. C#C# .NET 버전 4.5에서 구현되고 사용됩니다. ADMentor는 섹션 III의 개념을 구현하는 AKM(Architectural Knowledge Management) 의미 체계를 전달하는 UML 프로필 및 MDG 기술(Sparx EA의 두 가지 확장성 개념)과 함께 제공됩니다.

Our rationale for the selection of EA as the UML tool and. general-purpose modeling platform to be extended is threefold. First and foremost, it is used by many architects and therefore a preferred choice of our industry partner. Secondly, it has adequate support for rich text (e.g. Web links, bullet lists, etc.), model refactoring and UML tagged values, all of which is needed to implement our concepts. The third justification is that an early proof-of-technology and the predecessor project [11] demonstrated technical feasibility and usability of APIs and extensibility mechanisms of EA (some implementation difficulties had to be overcome).

EA를 UML 도구로 선택한 이유와 확장될 범용 모델링 플랫폼은 세 가지입니다. 무엇보다도 많은 건축가가 사용하므로 업계 파트너가 선호하는 선택입니다. 둘째, 서식 있는 텍스트(예: 웹 링크, 글머리 기호 목록 등), 모델 리팩토링 및 UML 태그 값에 대한 적절한 지원이 있으며, 이 모든 것이 우리의 개념을 구현하는 데 필요합니다. 세 번째 정당성은 초기 기술 증명과 이전 프로젝트[11]가 EA의 API 및 확장성 메커니즘의 기술적 타당성과 유용성을 입증했다는 것입니다(일부 구현 어려움을 극복해야 함).

The key features of ADMentor Version 1.1 are:

ADMentor 버전 1.1의 주요 기능은 다음과 같습니다.

Problem space modeling: recurring design decisions, options to be considered – providing a checklist effect Model tailoring and solution space modeling: decisions made and their rationale, decision backlog Model refactoring, reporting via Enterprise Architect integration; modeling patterns (template configuration) Rich text editing, decision capturing with lightweight decision capturing templates such as Y-statements Question, Option, Criteria (QOC) diagram support

문제 공간 모델링: 반복되는 설계 결정, 고려해야 할 옵션 – 체크리스트 효과 제공 모델 맞춤화 및 솔루션 공간 모델링: 내린 결정 및 그 근거, 의사 결정 백로그 모델 리팩토링, Enterprise Architect 통합을 통한 보고; 모델링 패턴(템플릿 구성) 리치 텍스트 편집, Y-문 QOC(Question, Option, Criteria) 다이어그램 지원과 같은 경량 의사 결정 캡처 템플릿을 사용한 의사 결정 캡처

The above feature list shows that that our implementation. stays very close to the concepts from Section III. The metainformation elements from Table 2 are realized as UML tagged values; the decision backlog is a customization of the package. browser, a standard EA feature. Additionally, ADMentor also supports decision space analytics (e.g., number of options and problems per package and meta-information tag and breakdown of problem occurrences by state), model validation, and a RESTful HTTP interface for tool integration..

위의 기능 목록은 우리의 구현을 보여줍니다. 섹션 III의 개념에 매우 가깝습니다. 표 2의 메타 정보 요소는 UML 태그가 지정된 값으로 실현됩니다. 의사결정 백로그는 패키지의 사용자 정의입니다. 브라우저, 표준 EA 기능. 또한 ADMentor는 의사 결정 공간 분석(예: 패키지당 옵션 및 문제 수, 메타 정보 태그, 상태별 문제 발생 분석), 모델 검증 및 도구 통합을 위한 RESTful HTTP 인터페이스도 지원합니다.

Problems/options and their respective occurrences are linked with the standard UML/EA concepts of classifiers and instances [18]. The model tailoring and filtering capabilities are based on tagged values (a standard UML element extension mechanism [18]). The typed links are realized as connector stereotypes that are defined in our UML profile. In addition to the concepts introduced in Section III, we also added two package stereotypes and additional links to our UML profile. The link definitions in ADMentor are compatible with other link modeling taxonomies; EA extensibility allows the user to add even more stereotypes, e.g. for relationships defined in the Kruchten/Lago/van Vliet ontology (see e.g. Chapter 3 in [1]).

문제/옵션 및 해당 발생은 분류기 및 인스턴스의 표준 UML/EA 개념과 연결됩니다[18]. 모델 조정 및 필터링 기능은 태그가 지정된 값(표준 UML 요소 확장 메커니즘[18])을 기반으로 합니다. 형식화된 링크는 UML 프로필에 정의된 커넥터 스테레오타입으로 구현됩니다. 섹션 III에서 소개한 개념 외에도 두 개의 패키지 스테레오타입과 UML 프로필에 대한 추가 링크도 추가했습니다. ADMentor의 링크 정의는 다른 링크 모델링 분류와 호환됩니다. EA 확장성을 통해 사용자는 예를 들어 Kruchten/Lago/van Vliet 온톨로지에 정의된 관계에 대해 더 많은 스테레오타입을 추가할 수 있습니다(예: [1]의 3장 참조).

Figure 3 on the next page shows a Problem Space Diagram (PSD) with fundamental cloud computing ADs modelled in ADMentor. Blue/dark diamonds represent recurring design problems, i.e., the need for a decision (which should not be confused with a quality attribute or stakeholder concern, e.g. in. late design); options appear as yellow/light rounded ovals. The primary link between problems and options is an addressedBy. relation. Two recurring problems are linked this way here;. additional relationship links from Section III are shown in the EA toolbar (on the left side). The element notes view on the bottom right contains rich text including hyperlinks (to websites, but also other model elements, e.g., UML classes).

다음 페이지의 그림 3은 ADMentor에서 모델링된 기본 클라우드 컴퓨팅 AD가 포함된 PSD(Problem Space Diagram)를 보여줍니다. 파란색/어두운 다이아몬드는 반복되는 설계 문제, 즉 결정의 필요성을 나타냅니다(예: 후기 설계와 같이 품질 속성 또는 이해관계자의 관심사와 혼동해서는 안 됨). 옵션은 노란색/밝은 둥근 타원으로 나타납니다. 문제와 옵션 간의 기본 링크는 addressedBy입니다. 관련. 두 가지 반복되는 문제가 여기에서 이런 식으로 연결됩니다. 섹션 III의 추가 관계 링크는 EA 도구 모음(왼쪽)에 표시됩니다. 오른쪽 하단의 요소 메모 보기에는 하이퍼링크(웹 사이트뿐만 아니라 UML 클래스와 같은 다른 모델 요소로도 포함)를 포함한 서식 있는 텍스트가 포함되어 있습니다.

Figure 4, also on the next page, is an Architecture. Overview Diagram (AOD) that illustrates the conceptual. architecture of ADMentor in a functional and logical view. The figure shows the components of ADMentor (either through customization of EA when possible or, alternatively, through. C#C# and .NET development), which can be traced back easily to. the research contributions from the previous section and the tool requirements from Section II.

다음 페이지에 있는 그림 4는 아키텍처입니다. 개념을 설명하는 AOD(Overview Diagram)입니다. 기능적이고 논리적인 관점에서 ADMentor의 아키텍처. 그림은 ADMentor의 구성 요소를 보여줍니다(가능한 경우 EA 사용자 정의를 통해, 또는 다음을 통해. C#C# 및 .NET 개발)을 쉽게 추적할 수 있습니다. 이전 섹션의 연구 기여와 섹션 II의 도구 요구 사항.

In the AOD, the components are placed in three logical layers (presentation, business logic, and data access/persistence layer) suggested by Fowler [4]. The left side of the AOD shows components supporting particularly relevant features in the standard EA product; the right side shows our extensions (addin components). Call and dependencies links are not shown due to space constraints.

AOD에서 구성 요소는 Fowler[4]가 제안한 세 가지 논리 계층(프레젠테이션, 비즈니스 논리 및 데이터 액세스/지속성 계층)에 배치됩니다. AOD의 왼쪽에는 표준 EA 제품에서 특히 관련된 기능을 지원하는 구성 요소가 표시됩니다. 오른쪽에는 확장 프로그램 (애드인 구성 요소)이 표시됩니다. 호출 및 종속성 링크는 공간 제약으로 인해 표시되지 않습니다.

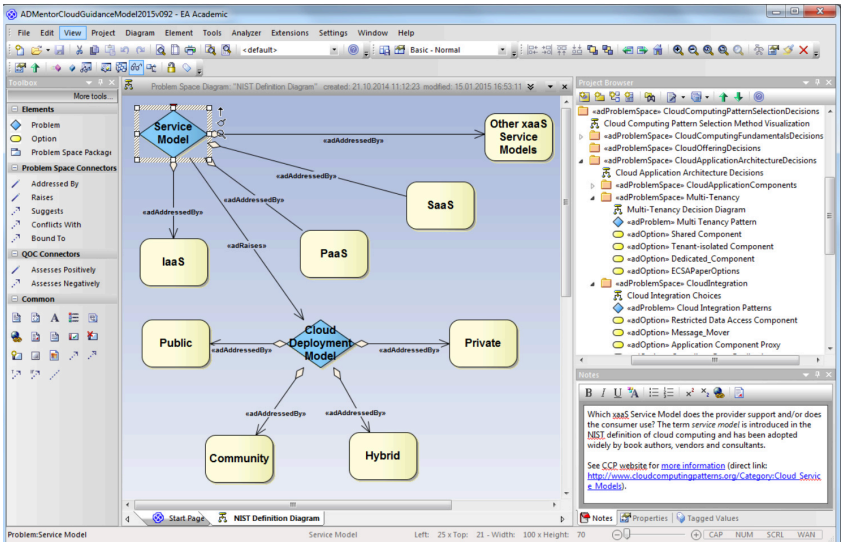

Fig. 3. Problem Space Diagram, Project Browser and Notes Editor in ADMentor (two sample problems and their options from cloud computing).
그림 3. ADMentor의 문제 공간 다이어그램, 프로젝트 브라우저 및 메모 편집기(클라우드 컴퓨팅의 두 가지 샘플 문제 및 해당 옵션).

V. MODELLING ACTIVITIES (VALIDATION PHASE 2)

V. 모델링 활동(검증 2단계)

Our main validation type in phase 2 was action research. (i.e., use of concepts and tool on our own projects) [17]. One. validation objective was to reconfirm that architecture design. problem indeed recur (as already shown in a different technical domain in our previous work)..

2단계의 주요 검증 유형은 행동 연구였습니다. (즉, 자체 프로젝트에서 개념 및 도구 사용) [17]. 하나. 검증 목표는 아키텍처 설계를 재확인하는 것이었습니다. 문제는 실제로 반복됩니다 (이전 작업의 다른 기술 영역에서 이미 표시된 대로).

We also evaluated the expressivity and usability of our. novel concepts (that address research questions 1 and 2 from Section I) and their implementation and measured the modelling and decision capturing efforts.

또한 우리의 표현력과 유용성도 평가했습니다. 새로운 개념(섹션 I의 연구 질문 1과 2를 다루는 것)과 그 구현을 측정하고 모델링 및 의사 결정 캡처 노력을 측정했습니다.

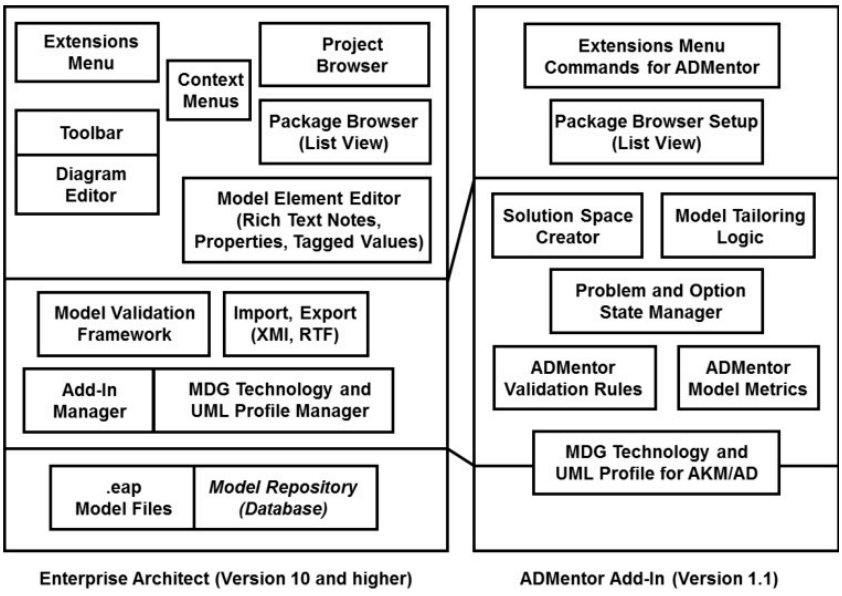

Fig. 4. ADMentor architecture (in a high-level functional/logical viewpoint)
그림 4. ADMentor 아키텍처(높은 수준의 기능/논리적 관점에서)

A. A Problem Space for Cloud Computing

A. 클라우드 컴퓨팅을 위한 문제 공간

In parallel to concept creation and tool development, we created a problem space model for cloud computing, a technical domain that is of interest to many architects at present. This model has the primary goal to demonstrate that our modeling concepts work in practice and are beneficial to architects. Some examples were already given (session state management, session database provider, cloud service model, and cloud deployment model as defined by NIST and [3]). Other cloud design topics covered by the problem space are:

컨셉 생성 및 도구 개발과 병행하여 현재 많은 건축가들이 관심을 갖고 있는 기술 영역인 클라우드 컴퓨팅을 위한 문제 공간 모델을 만들었습니다. 이 모델의 주요 목표는 우리의 모델링 개념이 실제로 작동하고 건축가에게 유익하다는 것을 입증하는 것입니다. 몇 가지 예가 이미 제공되었습니다(NIST 및 [3]에서 정의한 세션 상태 관리, 세션 데이터베이스 공급자, 클라우드 서비스 모델 및 클라우드 배포 모델). 문제 공간에서 다루는 다른 클라우드 디자인 항목은 다음과 같습니다.

Use of Cloud Computing Patterns (CCP) such as hypervisor, map-reduce, and key-value storage [3]. Patterns for multi-tenancy, workload, message delivery. Cloud service management (e.g., watchdog pattern).

하이퍼바이저, 맵 감소 및 키-값 스토리지와 같은 클라우드 컴퓨팅 패턴(CCP) 사용[3]. 다중 테넌시, 워크로드, 메시지 배달에 대한 패턴입니다. 클라우드 서비스 관리(예: 워치독 패턴).

The patterns in the CCP book turned out to be well suited for a PSD representation in ADMentor: some patterns share the same problem statement and/or describe alternative solutions in. a design category, e.g. multi tenancy. The CCP website can be linked to, which reduces the modeling effort significantly. No content was copy-pasted, but URIs added; the package structure in ADMentor mirrors that of book and website, e.g., the category "cloud application components" is found under "cloud application architecture". We blended in selected cloud knowledge from other sources, including books, but also blogs and own projects, and yielded 45 problems. All problems and options were modelled rather tersely to minimize creation and consumption effort (see Figure 4 in Section IV), but still be informative in the sense of a checklist and pattern language compass. The model was validated by creating a sample solution space (see below). In addition, one of the pattern book authors received a demonstration to confirm that the pattern knowledge in the book is represented properly in ADMentor.

CCP 책의 패턴은 ADMentor의 PSD 표현에 매우 적합한 것으로 판명되었습니다: 일부 패턴은 동일한 문제 설명을 공유하거나 대체 솔루션을 설명합니다. 디자인 범주(예: 다중 테넌시). CCP 웹 사이트를 링크할 수 있으므로 모델링 작업이 크게 줄어듭니다. 복사 붙여넣은 콘텐츠는 없지만 URI가 추가되었습니다. ADMentor의 패키지 구조는 책과 웹사이트의 구조를 반영합니다(예: "클라우드 애플리케이션 구성 요소" 범주는 "클라우드 애플리케이션 아키텍처"에서 찾을 수 있습니다. 우리는 책뿐만 아니라 블로그 및 자체 프로젝트를 포함한 다른 소스에서 선택된 클라우드 지식을 혼합하여 45개의 문제를 도출했습니다. 모든 문제와 옵션은 생성 및 소비 노력을 최소화하기 위해 다소 간결하게 모델링되었지만(섹션 IV의 그림 4 참조) 체크리스트 및 패턴 언어 나침반의 의미에서 여전히 유익합니다. 모델은 샘플 솔루션 공간을 만들어 유효성을 검사했습니다(아래 참조). 또한 패턴북 저자 중 한 명은 책의 패턴 지식이 ADMentor에서 제대로 표현되는지 확인하기 위해 시연을 받았습니다.

In a second step, an existing decision log from a cloud. project at our industry partner was transferred from Decision Architect (see Section II) to ADMentor to show feasibility and compatibility (of metamodels and tools) and to compare expressivity of concepts and their implementation. 14 problems were migrated (and re-modelled). We observed the same, or even better, expressivity of model elements and links; rich text notes in EA improve the user experience over plain text fields (as used in Decision Architect). Being inspired by the states suggested in the existing work [1], the state machine in Decision Architect initially was more powerful; when we realized this, we also integrated these concepts into ADMentor. and aligned them with the problem and option states propagation from Section III. The linkage was achieved via the Relationship Viewpoint in Decision Architect; decisions can link to problem and option occurrences. No specific semantically rich link type was defined for that, as the generic trace links in UML/EA were deemed to be sufficient..

두 번째 단계에서는 클라우드의 기존 의사 결정 로그입니다. 업계 파트너의 프로젝트는 (메타 모델 및 도구의) 타당성과 호환성을 보여주고 개념의 표현성과 구현을 비교하기 위해 Decision Architect(섹션 II 참조)에서 ADMentor로 이전되었습니다. 14개의 문제가 마이그레이션(및 재모델링)되었습니다. 우리는 모델 요소와 링크의 동일하거나 더 나은 표현력을 관찰했습니다. EA의 서식 있는 텍스트 메모는 일반 텍스트 필드(Decision Architect에서 사용되는 대로)에 대한 사용자 경험을 개선합니다. 기존 작업[1]에서 제안된 상태에서 영감을 받아 Decision Architect의 상태 머신은 처음에는 더 강력했습니다. 이를 깨달았을 때 이러한 개념도 ADMentor에 통합했습니다. 섹션 III의 문제 및 옵션 상태 전파와 일치시켰습니다. 연결은 Decision Architect의 관계 관점을 통해 달성되었습니다. 결정은 문제 및 옵션 발생과 연결될 수 있습니다. UML/EA의 일반 추적 링크가 충분하다고 간주되었기 때문에 이에 대해 의미론적으로 풍부한 특정 링크 유형이 정의되지 않았습니다..

In a third step, 26 problems about enterprise application. architecture and enterprise integration (messaging) were also. modelled as recurring problems; while these decisions are not. specific to cloud computing, they continue to be relevant and refine the ones in the CCP book. The session state management. problem and its pattern options from [4] fall in this category.

세 번째 단계에서는 엔터프라이즈 응용 프로그램에 대한 26개의 문제. 아키텍처 및 엔터프라이즈 통합(메시징)도 마찬가지였습니다. 반복되는 문제로 모델링됨; 이러한 결정은 클라우드 컴퓨팅에만 국한된 것은 아니지만 계속해서 관련성이 있으며 CCP 책에 있는 결정을 개선하고 있습니다. 세션 상태 관리입니다. [4]의 문제 및 패턴 옵션이 이 범주에 속합니다.

In total, Version 1.0 of the reusable problem space for cloud computing compiles 85 problems and 226 options.

클라우드 컴퓨팅을 위한 재사용 가능한 문제 공간의 버전 1.0은 총 85개의 문제와 226개의 옵션을 컴파일합니다.

B. Instantiation of Cloud Problem Space

B. 클라우드 문제 공간의 인스턴스화

This validation activity targeted the goals of validating the. problem space instantiation of the ADMentor tool as well as. the completeness and appropriateness of the cloud problem space that was created in validation activity A. This activity was performed by one of the authors of the paper, an industrial researcher who shadowed an architect of an ABB business unit. The approach to this validation was to import the problem space into the already existing UML model of a prototypical ABB cloud application that was built based on the conceptual architecture also used in step two of validation activity A. First, the problem space was tailored to fit the specific project, e.g.,. by omitting the Enterprise Application Decisions package (see step three in validation activity A), which was rated as not relevant for this project. The ADMentor user then instantiated a corresponding solution space and went through the problems step by step to document the decisions retrospectively. Of the total 48 problem occurrences in the joint cloud problem space 25 were completely solved, 4 partially solved, 8 deemed not applicable and 11 left open. Additionally, 17 key problem occurrences were linked to the corresponding model elements in the structural architecture model.

이 검증 활동은 검증 목표를 목표로 했습니다. ADMentor 도구의 문제 공간 인스턴스화 뿐만 아니라. 유효성 검증 활동 A에서 작성된 클라우드 문제점 공간의 완전성 및 적절성. 이 활동은 논문의 저자 중 한 명인 ABB 사업부의 설계자를 따라다녔던 산업 연구원이 수행했습니다. 이 검증에 대한 접근 방식은 검증 활동 A의 2단계에서도 사용된 개념 아키텍처를 기반으로 구축된 프로토타입 ABB 클라우드 애플리케이션의 기존 UML 모델로 문제 공간을 가져오는 것이었습니다. 첫째, 문제 공간은 특정 프로젝트에 맞게 조정되었습니다. Enterprise Application Decisions 패키지(유효성 검사 활동 A의 3단계 참조)를 생략하여 이 프로젝트와 관련이 없는 것으로 평가되었습니다. 그런 다음 ADMentor 사용자는 해당 솔루션 공간을 인스턴스화하고 문제를 단계별로 검토하여 결정을 소급하여 문서화했습니다. 공동 클라우드 문제 공간에서 발생한 총 48건의 문제 중 25건은 완전히 해결되었고, 4건은 부분적으로 해결되었으며, 8건은 해당 불가로 간주되었으며, 11건은 미해결 상태로 남아 있었습니다. 또한 17개의 주요 문제 발생이 구조 아키텍처 모델의 해당 모델 요소와 연결되었습니다.

The results of the validation activity were as follows: the ADMentor user was able to complete the instantiation within a relatively short time (around 2.5 hours). The tool provided enough guidance to efficiently fulfill the task. In addition to the retrospective documentation of the actually made 14 problem occurrences, which had already

partially been documented using a different tool, 23 further decisions were uncovered. Previously, these decisions were only implicitly documented or not made at all. With the help of the cloud guidance model it. was possible to fill these gaps and capture this knowledge. which might otherwise have been lost or only recoverable with significant additional effort.

검증 활동의 결과는 다음과 같습니다. ADMentor 사용자는 비교적 짧은 시간(약 2.5시간) 내에 인스턴스화를 완료할 수 있었습니다. 이 도구는 작업을 효율적으로 수행하기에 충분한 지침을 제공했습니다. 이미 다른 도구를 사용하여 부분적으로 문서화된 실제로 발생한 14건의 문제 발생에 대한 소급 문서화 외에도 23건의 추가 결정이 발견되었습니다. 이전에는 이러한 결정이 암묵적으로만 문서화되거나 전혀 내려지지 않았습니다. 클라우드 지침의 도움으로 모델링합니다. 이러한 격차를 메우고 이 지식을 포착하는 것이 가능했습니다. 그렇지 않으면 손실되거나 상당한 추가 노력을 통해서만 복구할 수 있습니다.

C. Workflow Decision Modeling (with Industry Participants)

C. 워크플로우 의사결정 모델링(업계 참가자 포함)

Our third validation activity was the participation in a twoday community meeting of 26 software architects from various German companies (including banks, insurance firms, telecommunications service providers, and professional IT consulting services). The action researcher prepared an initial problem space for the design of workflows, via reflection of own experience and an ad hoc literature review. He presented the ADMentor vision as well as sample content from the initial problem space to the meeting participants (at the start of the meeting); e.g., the shown content included decisions about transaction boundaries and service granularity. The action researcher then facilitated a short exercise in which participants were asked to identify recurring decisions themselves; selected decisions were then discussed in the plenum. The attendee feedback included general agreement as well as four additional problems with options (e.g. on placement of business data in workflow, process instance migration, and interface signature sourcing). This activity reconfirmed the general hypothesis that architectural decisions recur; as the modeling work was continued during subsequent workshop presentations, it also showed that the ADMentor tool can be used in meeting and design workshop situations (for decision modeling on the fly). The final problem space model for workflow design comprises 75 recurring problems with 150 options..

세 번째 검증 활동은 다양한 독일 회사(은행, 보험 회사, 통신 서비스 제공업체 및 전문 IT 컨설팅 서비스 포함)의 26명의 소프트웨어 설계자가 참여한 이틀간의 커뮤니티 회의에 참여하는 것이었습니다. 행동 연구원은 자신의 경험을 반영하고 임시 문헌 검토를 통해 워크플로 설계를 위한 초기 문제 공간을 준비했습니다. 그는 ADMentor 비전과 초기 문제 공간의 샘플 콘텐츠를 회의 참가자에게 제시했습니다(회의 시작 시). 예를 들어, 표시된 콘텐츠에는 트랜잭션 경계 및 서비스 세분성에 대한 결정이 포함되었습니다. 그런 다음 행동 연구원은 참가자들이 반복되는 결정을 스스로 식별하도록 요청하는 짧은 연습을 진행했습니다. 그런 다음 선택된 결정이 전원회의에서 논의되었습니다. 참석자 피드백에는 일반적인 동의뿐만 아니라 옵션에 대한 네 가지 추가 문제(예: 워크플로에 비즈니스 데이터 배치, 프로세스 인스턴스 마이그레이션 및 인터페이스 서명 소싱)가 포함되었습니다. 이 활동은 건축적 결정이 반복된다는 일반적인 가설을 재확인했습니다. 후속 워크숍 프레젠테이션 중에 모델링 작업이 계속됨에 따라 ADMentor 도구가 회의 및 설계 워크숍 상황(즉석에서 의사 결정 모델링을 위해)에 사용될 수 있음을 보여주었습니다. 워크플로 설계를 위한 최종 문제 공간 모델은 150개의 옵션과 함께 75개의 반복되는 문제로 구성됩니다.

D. Third-party Problem Space Modeling

D. 제3자 문제 공간 모델링

This validation activity concerned the retrospective. modeling of a problem space and several solution spaces for industrial control systems. The goals were a) to validate ADMentor's problem space modeling capabilities concerning expressiveness and efficiency , b) to verify the traceability. features to other model elements, and c) to gain experience on working with multiple solution spaces for a given problem space. In this case, an industrial researcher with no prior experience with ADMentor created the problem space from company-internal input, thus this validation activity concerns an external application of ADMentor as problem space. modeling tool other than the tool authors themselves. However, only qualitative statements about the application can be made.

이 검증 활동은 회고전과 관련이 있습니다. 산업 제어 시스템을 위한 문제 공간 및 여러 솔루션 공간의 모델링. 목표는 a) 표현력 및 효율성에 관한 ADMentor의 문제 공간 모델링 기능을 검증하는 것, b) 추적성을 검증하는 것이었습니다. 다른 모델 요소에 대한 기능, c) 주어진 문제 공간에 대해 여러 솔루션 공간으로 작업하는 경험을 얻습니다. 이 경우 ADMentor에 대한 사전 경험이 없는 산업 연구원이 회사 내부 입력에서 문제 공간을 생성했으므로 이 검증 활동은 문제 공간으로서의 ADMentor의 외부 적용과 관련이 있습니다. 도구 작성자 자체 이외의 모델링 도구. 그러나 응용 프로그램에 대한 정성적 진술만 할 수 있습니다.

In this validation activity the approach was to exploit an existing technical report surveying the design concepts of different control systems and to model the included problem space. The problem space consisted of 16 problems and 36 options grouped into three different packages. Explanations for the options were copied over from the document into the model in some cases; in other cases, direct references to the technical report were created because reading lots of text inside Enterprise Architect proved to be cumbersome. From the problem space, three solutions spaces for three different control systems were derived. Existing UML models for these systems were then copied into the same Enterprise Architect project so that trace links between UML elements and ADMentor elements could be created. For example, traces were modelled from decisions to software components that had been implemented as a consequence of a particular decision.

이 검증 활동에서 접근 방식은 다양한 제어 시스템의 설계 개념을 조사하는 기존 기술 보고서를 활용하고 포함된 문제 공간을 모델링하는 것이었습니다. 문제 공간은 16개의 문제와 36개의 옵션이 세 가지 다른 패키지로 그룹화되었습니다. 옵션에 대한 설명은 경우에 따라 문서에서 모델로 복사되었습니다. 다른 경우에는 Enterprise Architect 내에서 많은 텍스트를 읽는 것이 번거롭기 때문에 기술 보고서에 대한 직접 참조가 생성되었습니다. 문제 공간에서 세 가지 다른 제어 시스템에 대한 세 가지 솔루션 공간이 도출되었습니다. 그런 다음 이러한 시스템에 대한 기존 UML 모델을 동일한 Enterprise Architect 프로젝트에 복사하여 UML 요소와 ADMentor 요소 간의 추적 링크를 만들 수 있습니다. 예를 들어, 추적은 특정 결정의 결과로 구현된 소프트웨어 구성 요소에 대한 결정에서 모델링되었습니다.

This validation activity provided the following results: we noticed reported the ability to quickly (i.e., less than two hours) to create the problem space with ADMentor based on the available reference material. The expressiveness of ADMentor was sufficient to capture the required information. In addition the problem and solution space models provided a condensed overview of the made decisions and neglected options which was deemed useful. The creation of trace links between the modeling elements worked successfully. This ability was well received as tracing both from a particular UML element to an ADMentor element and vice versa is supported. Thus, it is convenient to derive the decisions attached to a specific component in the model or to identify the components affected by a specific decision. One challenge is that the solution spaces are not updated when the problem space is extended, thus the user needs to manually update the solution spaces. The interplay between the different viewpoints of the Decision Architect and ADMentor elements worked well. Architects from an ABB business unit reviewed the resulting solution space model informally and provided positive feedback about accuracy and usefulness of the models.

이 검증 활동은 다음과 같은 결과를 제공했습니다. 우리는 사용 가능한 참조 자료를 기반으로 ADMentor로 문제 공간을 신속하게(즉, 2시간 미만) 생성할 수 있는 능력이 보고된 것을 발견했습니다. ADMentor의 표현력은 필요한 정보를 포착하기에 충분했습니다. 또한 문제 및 솔루션 공간 모델은 내린 결정과 무시된 옵션에 대한 요약된 개요를 제공하여 유용하다고 간주되었습니다. 모델링 요소 간의 추적 링크 생성이 성공적으로 작동했습니다. 이 기능은 특정 UML 요소에서 ADMentor 요소로, 또는 그 반대로 추적이 지원되기 때문에 호평을 받았습니다. 따라서 모델의 특정 구성 요소에 첨부된 결정을 도출하거나 특정 결정의 영향을 받는 구성 요소를 식별하는 것이 편리합니다. 한 가지 문제는 문제 공간이 확장될 때 솔루션 공간이 업데이트되지 않으므로 사용자가 솔루션 공간을 수동으로 업데이트해야 한다는 것입니다. 의사결정 설계자와 ADMentor 요소의 다양한 관점 간의 상호 작용이 잘 작동했습니다. ABB 사업부의 건축가들은 결과 솔루션 공간 모델을 비공식적으로 검토하고 모델의 정확성과 유용성에 대해 긍정적인 피드백을 제공했습니다.

Although ADMentor was not used for forward engineering. in this validation activity, its application proved valuable for our industrial partner. The usability of the tooling was deemed. satisfactory and the instantiation of the problem space in three different solution spaces was achieved quickly. The model can. potentially be used in the future for forward engineering when. new systems of the same class are designed..

ADMentor는 전방 엔지니어링에 사용되지 않았지만. 이 검증 활동에서 이 응용 프로그램은 산업 파트너에게 가치 있는 것으로 입증되었습니다. 툴링의 유용성이 고려되었습니다. 만족스럽고 세 가지 다른 솔루션 공간에서 문제 공간의 인스턴스화가 신속하게 달성되었습니다. 모델은 할 수 있습니다. 잠재적으로 미래에 전방 엔지니어링에 사용될 수 있습니다. 같은 등급의 새로운 시스템이 설계되었습니다..

Validation summary. The evaluation results demonstrate that.

유효성 검사 요약. 평가 결과는 이를 보여줍니다.

our concepts and their implementation work in practice; users.

우리의 개념과 구현은 실제로 작동합니다. 사용자.

have to invest relatively little effort to be productive and.

생산성을 높이기 위해 상대적으로 적은 노력을 투자해야 합니다.

experience benefits. Hence, our research contributions and their implementation answer research the questions from Section I..

경험 혜택. 따라서 우리의 연구 기여와 구현은 섹션 I.의 질문에 대한 연구에 답합니다.

VI. DISCUSSION

VI. 토론

Contributions and their novelty. The lessons learned on previous AKM projects were taken into account during design and development of ADMentor. In response to RQ1 and RQ 2 from Section 1, a leaner approach is now promoted, which means less modelling and model maintenance effort (for knowledge engineers), and also less consumption effort (for project architects). A decision backlog is available; metainformation is more comprehensive and more flexible so that it can be extended. We define AKM quadruples (i.e., problem, option, problem occurrence, option occurrence) and a workflow for processing them. The quadruples make the architectural decision knowledge more consumable, reusable and manageable. They are created and consumed in the context of a general-purpose modeling tool. In this approach, problems and options are harvested and curated as reusable assets; problem and option occurrences are then created by project architects as needed. This separation of knowledge management into 2x22x2 dimensions (and/or perspectives) is a key differentiator between our approach and the related work. Our meta-information attribution and its implementation with UML tagged values differs both from the universal/fixed modeling approach in SOAD and the findings and results of the GRIFFIN project (see Chapters 6 and 8 in [1]): unlike, SOAD, the GRIFFIN core model only defined knowledge entities, but not their attributes. For ADMentor, we found a "meet-in-themiddle" compromise: decision templates can be flexibly configured via the rich text editor; additional tagged values. (meta-information annotations) can also be defined.

기여와 참신함. 이전 AKM 프로젝트에서 배운 교훈은 ADMentor의 설계 및 개발 중에 고려되었습니다. 섹션 1의 RQ1 및 RQ 2에 대한 응답으로 이제 더 간결한 접근 방식이 촉진되어 모델링 및 모델 유지 관리 노력(지식 엔지니어의 경우)과 소비 노력(프로젝트 설계자의 경우)도 줄어듭니다. 의사 결정 백로그를 사용할 수 있습니다. 메타 정보는 확장할 수 있도록 더 포괄적이고 유연합니다. AKM 4배(즉, 문제, 옵션, 문제 발생, 옵션 발생)과 이를 처리하기 위한 워크플로를 정의합니다. 4중은 아키텍처 의사 결정 지식을 보다 소비 가능하고 재사용 가능하며 관리하기 쉽게 만듭니다. 범용 모델링 도구의 컨텍스트에서 생성되고 사용됩니다. 이 접근 방식에서는 문제와 옵션이 재사용 가능한 자산으로 수집되고 선별됩니다. 그런 다음 필요에 따라 프로젝트 설계자가 문제 및 옵션 발생을 생성합니다. 지식 관리를 2x22x2 차원(및/또는 관점)으로 분리하는 것은 우리의 접근 방식과 관련 작업 간의 주요 차별화 요소입니다. 우리의 메타 정보 속성과 UML 태그 값을 사용한 구현은 SOAD의 보편적/고정 모델링 접근 방식과 GRIFFIN 프로젝트의 결과 및 결과([1]의 6장 및 8장 참조)와 다릅니다. SOAD와 달리 GRIFFIN 핵심 모델은 지식 개체만 정의하고 속성은 정의하지 않았습니다. ADMentor의 경우 "중간만남" 절충안을 발견했습니다. 추가 태그가 지정된 값. (메타 정보 주석)도 정의할 수 있습니다.

ADMentor provides a superset of the functionality in previous research prototypes, but has a very different technical design and implementation (e.g., metamodel, tool architecture, and codification in .NET/C#). The most significant differences to previous implementations are: a) any decision capturing template can be used in the rich text notes, b) an option is not physically contained in a problem, c) multiple options can be chosen, d) there is rich but flexible meta-information tagging, and e) the list view of the package browser serves as a fullfledged decision backlog now. Our approach does not mandate UML, but can be implemented in any extensible modeling tool that meets the requirements from Section II. Our UML profile can be exported via a platform feature for reuse.

ADMentor는 이전 연구 프로토타입의 기능 상위 집합을 제공하지만 매우 다른 기술 설계 및 구현(예: 메타모델, 도구 아키텍처 및 .NET/C#의 코드화)을 가지고 있습니다. 이전 구현과의 가장 중요한 차이점은 a) 모든 의사 결정 캡처 템플릿을 서식 있는 텍스트 메모에서 사용할 수 있음, b) 옵션이 문제에 물리적으로 포함되지 않음, c) 여러 옵션을 선택할 수 있음, d) 풍부하지만 유연한 메타 정보 태그 지정이 있음, e) 패키지 브라우저의 목록 보기가 이제 본격적인 의사 결정 백로그 역할을 합니다. 우리의 접근 방식은 UML을 의무화하지 않지만 섹션 II의 요구 사항을 충족하는 확장 가능한 모델링 도구에서 구현할 수 있습니다. UML 프로필은 재사용을 위해 플랫폼 기능을 통해 내보낼 수 있습니다.

Threats to validity of validation. We aimed at capturing both. the quality of the tool and the created guidance models as well. as the perceived efficiency in working with the tool. However, threats to the validity of our validation activities still exist: The researchers performed the majority of the validation activities themselves (action research). Furthermore, the amount of case studies was of course not statistically relevant, so all our results. reside on the qualitative level. Thus, the external validity of our validation results, at least regarding efficiency, might be reduced. Finally, a threat regarding the created problem spaces model exists: the cloud computing domain is still evolving, and other architects might find other sources of more suitable. knowledge and guidance than the ones that we picked..

유효성 검사의 유효성에 대한 위협. 우리는 둘 다 포착하는 것을 목표로 했습니다. 도구의 품질과 생성된 안내 모델도 마찬가지입니다. 도구 작업에서 인지된 효율성으로. 그러나 검증 활동의 타당성에 대한 위협은 여전히 존재합니다: 연구원들은 대부분의 검증 활동(행동 연구)을 직접 수행했습니다. 또한 사례 연구의 양은 물론 통계적으로 관련이 없었기 때문에 모든 결과가 있습니다. 질적 수준에 상주합니다. 따라서 적어도 효율성과 관련하여 검증 결과의 외부 타당성이 감소할 수 있습니다. 마지막으로, 생성된 문제 공간 모델과 관련된 위협이 존재합니다: 클라우드 컴퓨팅 도메인은 여전히 진화하고 있으며 다른 설계자는 더 적합한 다른 소스를 찾을 수 있습니다. 우리가 선택한 것보다 지식과 지도..

We countered these three threats by using our approach in different contexts and with different people. We did not rely on a single case study, but performed several quite different modeling and decision making activities to broaden the validation scope. Additionally, we collected feedback on our approach and the resulting models from external stakeholders.

우리는 서로 다른 상황과 다양한 사람들과 함께 접근 방식을 사용하여 이 세 가지 위협에 대응했습니다. 우리는 단일 사례 연구에 의존하지 않고 검증 범위를 넓히기 위해 몇 가지 완전히 다른 모델링 및 의사 결정 활동을 수행했습니다. 또한 외부 이해관계자로부터 접근 방식과 결과 모델에 대한 피드백을 수집했습니다.

Impact on practice. We foresee problem space models to become virtual mentors making formerly tacit knowledge explicit in an easy-to-consume way; this concept is well suited for globally distributed development organization (with some amount of coordination). As a result, better decisions can be made in less time; the decision makers are empowered, but also held accountable. Deviations from group-level standards. around patterns, technologies, and products can be chosen; but such new, non-standard solutions to known problems are now documented along with justifications. To increase the reuse. potential and the longevity of the architectural knowledge,. problems and options are pointed out (as an incentive to the. architect/decision maker to get engaged), but not blindly. promoted. We do not to try to make the chosen solutions reusable (as this would lead to an unrealistic one-size-fits-all approach to architecting systems that ignores project context, requirements, and constraints). As a result, architects remain masters of their project's destiny with ADMentor, but their decisions are backed by the guidance and recommendations given by the community that contributed to the problem space models used. For instance, the cloud guidance model can be applied and extended on future research and development projects. We hope that it will also serve as a steward for future decision modeling work in the community.

실습에 미치는 영향. 우리는 문제 공간 모델이 이전의 암묵적 지식을 사용하기 쉬운 방식으로 명시적으로 만드는 가상 멘토가 될 것으로 예상합니다. 이 개념은 전 세계적으로 분산된 개발 조직(어느 정도의 조정 포함)에 매우 적합합니다. 결과적으로 더 짧은 시간에 더 나은 결정을 내릴 수 있습니다. 의사 결정권자는 권한을 부여받지만 책임도 집니다. 그룹 수준 표준에서 벗어났습니다. 패턴, 기술 및 제품을 선택할 수 있습니다. 그러나 알려진 문제에 대한 이러한 새롭고 비표준적인 해결책은 이제 정당성과 함께 문서화되어 있습니다. 재사용을 늘리기 위해. 건축 지식의 잠재력과 수명. 문제와 옵션이 지적되지만(건축가/의사 결정권자가 참여하도록 인센티브로) 맹목적으로 지적되 승진. 우리는 선택한 솔루션을 재사용할 수 있도록 만들려고 하지 않습니다(이는 프로젝트 컨텍스트, 요구 사항 및 제약 조건을 무시하는 시스템 설계에 대한 비현실적인 일률적인 접근 방식으로 이어질 수 있기 때문입니다). 결과적으로 건축가는 ADMentor를 통해 프로젝트 운명의 주인으로 남아 있지만 그들의 결정은 사용된 문제 공간 모델에 기여한 커뮤니티에서 제공한 지침과 권장 사항에 의해 뒷받침됩니다. 예를 들어, 클라우드 안내 모델은 향후 연구 개발 프로젝트에 적용 및 확장될 수 있습니다. 우리는 또한 커뮤니티에서 향후 의사 결정 모델링 작업을 위한 청지기 역할을 하기를 바랍니다.

VII. SUMMARY AND CONCLUSIONS

VII. 요약 및 결론

To overcome inhibitors for decision capturing and sharing,. we conceptualized and implemented novel approaches to. problem space modeling, solution space modelling, and decision backlog management. Problem spaces look forward and anticipate decisions to be made along with their options; solution spaces offer a lightweight form of decision guidance and capturing of past decisions to project architects. We. implemented our modeling concepts and applied them to craft problem and solution space exemplars for cloud computing,. workflow management, and distributed control systems..

의사 결정 캡처 및 공유를 위한 방해 요인을 극복하기 위해. 우리는 새로운 접근 방식을 개념화하고 구현했습니다. 문제점 공간 모델링, 솔루션 공간 모델링 및 의사결정 백로그 관리. 문제 공간은 옵션과 함께 내려질 결정을 기대하고 예상합니다. 솔루션 공간은 프로젝트 설계자에게 경량 형태의 의사 결정 지침과 과거 의사 결정 캡처를 제공합니다. 우리. 모델링 개념을 구현하고 이를 적용하여 클라우드 컴퓨팅을 위한 문제 및 솔루션 공간 예시를 제작했습니다. 워크플로 관리 및 분산 제어 시스템..

Problem and solution spaces present decision knowledge to architects in a form that can be seen as checklist with work items. This approach to knowledge reuse makes weaker assumptions about target environment and target audience than previous knowledge sharing attempts - and therefore promises to produce more timeless knowledge and guidance that is easier to accept. As a result, poor decisions can be avoided and no important decisions are missed; faster time-to-market and less technical risk on development projects can be achieved.

문제 및 솔루션 공간은 작업 항목에 대한 체크리스트로 볼 수 있는 형식으로 설계자에게 의사결정 지식을 제공합니다. 지식 재사용에 대한 이러한 접근 방식은 이전의 지식 공유 시도보다 대상 환경과 대상 청중에 대한 가정이 약하므로 더 쉽게 받아들일 수 있는 시대를 초월한 지식과 지침을 생성할 것을 약속합니다. 결과적으로 잘못된 결정을 피할 수 있고 중요한 결정을 놓치지 않습니다. 시장 출시 시간을 단축하고 개발 프로젝트에 대한 기술적 위험을 줄일 수 있습니다.

The extensible UML modeling tool Sparx Enterprise Architect is the carrier platform of ADMentor; however, our concepts are designed and presented in such a way that they also work in other infrastructures (e.g. project wikis, decision capturing spreadsheets and other semi-structured decision logs).4 We consider integrating ADMentor with project management and team collaboration tools. ADRepo, a Web. repository with a publicly accessible RESTful HTTP interface, will simplify such integration. Potential future work also includes an extension of ADMentor to represent architectural refactorings as architectural decisions to be revisited.

확장 가능한 UML 모델링 도구인 Sparx Enterprise Architect는 ADMentor의 캐리어 플랫폼입니다. 그러나 우리의 개념은 다른 인프라(예: 프로젝트 위키, 의사 결정 캡처 스프레드시트 및 기타 반구조화된 의사 결정 로그)에서도 작동하는 방식으로 설계되고 제시됩니다.4 우리는 ADMentor를 프로젝트 관리 및 팀 협업 도구와 통합하는 것을 고려합니다. ADRepo, 웹. 공개적으로 액세스할 수 있는 RESTful HTTP 인터페이스가 있는 저장소는 이러한 통합을 단순화합니다. 잠재적인 향후 작업에는 아키텍처 리팩토링을 재검토해야 할 아키텍처 결정으로 표현하기 위한 ADMentor의 확장도 포함됩니다.

ACKNOWLEDGMENT

승인

The research work presented in this paper was supported by an ABB Research Grant 2014. We would like to thank Tobias Blaser, Zoya Durdik, Christoph Fehling, Laurin Murer, Martin Naedele, Roland Weiss and the participants of the 8th8th workshop on software architectures at Softwareforen Leipzig for their input, support and/or constructive criticism.

이 논문에 제시된 연구 작업은 2014년 ABB 연구 보조금의 지원을 받았습니다. Tobias Blaser, Zoya Durdik, Christoph Fehling, Laurin Murer, Martin Naedele, Roland Weiss 및 8th 8th Softwareforen Leipzig의 소프트웨어 아키텍처 워크숍 참가자들에게 의견, 지원 및/또는 건설적인 비판에 감사드립니다.

REFERENCES

참조

[1] M. Ali Babar, T. Dingsoyr, P. Lago, H. van Vliet (eds.), Software Architecture Knowledge Management: Theory and Practice, SpringerVerlag, 2009.

[1] M. Ali Babar, T. Dingsoyr, P. Lago, H. van Vliet(eds.), 소프트웨어 아키텍처 지식 관리: 이론 및 실습, SpringerVerlag, 2009.

[2] M. Anvaari, O. Zimmermann, Towards Reusing Architectural Knowledge as Design Guides, Proc. of SEKE 2014, KSI, 2014.

[2] 중. Anvaari, O. Zimmermann, 건축 지식을 설계 가이드로 재사용하기 위하여, SEKE 2014 논문집, KSI, 2014.

[3]C. Fehling, F. Leymann., R. Retter, W. Schupeck, P. Arbitter, P., Cloud Computing Patterns, Springer 2013.

[3]C. Fehling, F. Leymann., R. Retter, W. Schupeck, P. Arbitter, P., 클라우드 컴퓨팅 패턴, Springer 2013.

[4] M. Fowler, Patterns of Enterprise Application Architecture. Addison Wesley, 2003.

[4] M. Fowler, 엔터프라이즈 애플리케이션 아키텍처 패턴. 애디슨 웨슬리, 2003.

[5] M. Fowler, Richardson Maturity Model,. http://martinfowler.com/articles/richardsonMaturityModel.html

[5] M. 파울러, 리처드슨 성숙도 모델,. http://martinfowler.com/articles/richardsonMaturityModel.html

[6] A. Gaind., Key Architecture Decisions Template, October 2005, Available via http://www.bredemeyer.com

[6] A. Gaind., Key Architecture Decisions Template, 2005년 10월, http://www.bredemeyer.com 를 통해 사용 가능

[7] P. Hruschka, G. Starke, arc42, Resources for software architects, http:/arc42.org)

[7] P. Hruschka, G. Starke, arc42, 소프트웨어 아키텍트를 위한 리소스, http:/arc42.org)

[8] ISO/IEC/IEEE, Systems_and software engineering - Architecture description, ISO/IEC/IEEE 42010:2011(E), Dec.12011, http://www.iso-architecture.org/ieee-1471/templates/

[8] ISO/IEC/IEEE, Systems_and 소프트웨어 엔지니어링 - 아키텍처 설명, ISO/IEC/IEEE 42010:2011(E), Dec.12011, http://www.iso-architecture.org/ieee-1471/templates/

[9] P. Kruchten, What color is your backlog?, via InfoQ blog post, 2010.

[9] P. Kruchten, 백로그는 어떤 색인가요?, InfoQ 블로그 게시물, 2010.

[10] M. Lindvall, I. Rus, R. Jammalamadaka, and R. Thakker. Software Tools for Knowledge Management: A DACs State-of-the-Art Report. Technical report, Fraunhofer Center for Experimental Software Engineering Maryland and The University of Maryland, 2001.

[10] M. 린드발, I. 루스, R. 잠말라마다카, R. 타커. 지식 관리를 위한 소프트웨어 도구: DAC 최첨단 보고서. 기술 보고서, 프라운호퍼 메릴랜드 실험 소프트웨어 공학 센터 및 메릴랜드 대학교, 2001.

[11] C. Manteuffel, D. Tofan, H. Koziolek, T. Goldschmidt, P. Avgeriou: Industrial Implementation of a Documentation Framework for Architectural Decisions. Proc. Of IEEE/IFIP WICSA 2014, IEEE Computer Society, Los Alamitos (2014).

[11] C. Manteuffel, D. Tofan, H. Koziolek, T. Goldschmidt, P. Avgeriou: 아키텍처 결정을 위한 문서화 프레임워크의 산업 구현. 프로시저. IEEE/IFIP WICSA 2014, IEEE 컴퓨터 학회, 로스 알라미토스(2014).

[12] A. MacLean, R. Young, V. Bellotti, and T. Moran, Questions, Options, and Criteria: Elements of Design Space Analysis, Human-Computer Interaction, 6 (3&4), 1991.

[12] A. MacLean, R. Young, V. Bellotti 및 T. Moran, 질문, 옵션 및 기준: 설계 공간 분석의 요소, 인간-컴퓨터 상호 작용, 6 (3&4), 1991.

[13] C.Miksovic, O.Zimmermann, ArchitecturallySignificant Requirements, Reference Architecture and Metamodel for Knowledge. Management in Information Technology Services. Proc. of IEEE/IFIP WICSA 2011 (2011), IEEE Computer Society, Los Alamitos (2011).

[13] C.Miksovic, O.Zimmermann, ArchitecturallySignificant Requirements, Reference Architecture and Metamodel for Knowledge. 정보 기술 서비스 관리. IEEE/IFIP WICSA 2011(2011), IEEE 컴퓨터 학회, 로스 알라미토스(2011).

[14] M. Nygard, Documenting architecture decisions, http://thinkrelevance.com/blog/2011/11/15/documenting-architecturedecisions

[14] M. Nygard, 아키텍처 결정 문서화, http://thinkrelevance.com/blog/2011/11/15/documenting-architecturedecisions

[15] Open Unified Process (OpenUP), http://epf.eclipse.org/wikis/openup/

[15] OpenUP(Open Unified Process), http://epf.eclipse.org/wikis/openup/

[16] N. Rozanski, E. Woods, Software Systems Architecture 11 Working With Stakeholders Using Viewpoints and Perspectives, Addison Wesley, 2005.

[16] N. Rozanski, E. Woods, 11 관점과 관점을 사용하여 이해관계자와 협력하는 소프트웨어 시스템 아키텍처, Addison Wesley, 2005.

[17] M. Shaw, Writing Good Software Engineering Research Papers:. Minitutorial. Proceedings of the 25th International Conference on Software Engineering. IEEE Computer Society, 2003.

[17] M. Shaw, 좋은 소프트웨어 공학 연구 논문 작성:. 미니튜토리얼. 제25회 소프트웨어 공학 국제 컨퍼런스 회의록. IEEE 컴퓨터 학회, 2003.

[18] Sparx Enterprise Architect Version 11 Users Guide, http://www.sparxsystems.com/enterprise_architect_user_guide/11/

[18] Sparx Enterprise Architect 버전 11 사용자 가이드, http://www.sparxsystems.com/enterprise_architect_user_guide/11/

[19] The official site of the Stage-Gate, http://www.stage-gate.com/

[19] 스테이지 게이트 공식 사이트, http://www.stage-gate.com/

[20] A. Tang, M. Ali Babar, I. Gorton., and J. Han, 2005. A Survey of the Use and Documentation of Architecture Design Rationale. Proc. of the. 5th Working IEEE/IFIP Conference on Software Architecture. IEEE Computer Society, 2005.

[20] A. Tang, M. Ali Babar, I. Gorton., J. Han, 2005. 건축 설계 이론적 근거의 사용 및 문서화에 대한 조사. Proc. 의. 소프트웨어 아키텍처에 관한 제5회 IEEE/IFIP 회의. IEEE 컴퓨터 학회, 2005.

[21] A. Tang, Y. Jin, J. Han: A rationale-based architecture model for design traceability and reasoning. Journal of Systems and Software 80(6): 918- 934 (2007).

[21] A. Tang, Y. Jin, J. Han: 설계 추적성 및 추론을 위한 근거 기반 아키텍처 모델. 시스템 및 소프트웨어 저널 80(6): 918-934 (2007).

[22] J. Tyree, A..Akerman: ArchitectureDecisions: Demystifying Architecture. IEEE Software 22(2): 19-27 (2005)

[22] J. 타이리, A.. Akerman: 아키텍처 결정: 건축의 신비화 IEEE 소프트웨어 22(2): 19-27 (2005)

[23] R. Weinreich, I. Groher. A Fresh Look at Codification Approaches for SAKM: A Systematic Literature Review. Proc. of ECSA 2014, Springer.

[23] R. Weinreich, I. 그로허. SAKM에 대한 성문화 접근 방식에 대한 새로운 시각: 체계적인 문헌 검토. ECSA 2014 절차, Springer.

[24] U. Zdun, R. Capilla, H. Tran, O. Zimmermann, Sustainable Architectural Design Decisions, IEEE Software, Volume 30, Number 6 (2013).

[24] U. Zdun, R. Capilla, H. Tran, O. Zimmermann, 지속 가능한 건축 설계 결정, IEEE 소프트웨어, 30권, 6호(2013).

[25] O. Zimmermann, An Architectural Decision Modeling Framework for SOA and Cloud Design, SEI SATURN 2010 Tutorial, page 14. SEI Techncial Library, http://resources.sei.cmu.edu/library/

[25] O. Zimmermann, SOA 및 클라우드 디자인을 위한 아키텍처 의사 결정 모델링 프레임워크, SEI SATURN 2010 튜토리얼, 14페이지. SEI 기술 도서관, http://resources.sei.cmu.edu/library/

[26] O. Zimmermann, C. Miksovic, Decisions Required vs. Decisions Made: Connecting Enterprise Architects and Solution Architects via Guidance Models, in: I. Mistrik, A. Tang, R. Bahsoon, J. Stafford (eds.), Aligning Enterprise, System, and Software Architectures. IGI Global (2013).

[26] O. Zimmermann, C. Miksovic, 필요한 의사 결정 대 내린 의사 결정: 안내 모델을 통해 엔터프라이즈 아키텍트와 솔루션 아키텍트 연결, in: I. Mistrik, A. Tang, R. Bahsoon, J. Stafford(eds.), 엔터프라이즈, 시스템 및 소프트웨어 아키텍처 조정. IGI 글로벌 (2013).

[27] O. Zimmermann, U. Zdun, T. Gschwind, F. Leymann, Combining Pattern Languages and Architectural Decision Models into a Comprehensive and. Comprehensible Design Method. Proc..of IEEE/IFIP WICSA 2008. IEEE Computer Society, Los Alamitos (2008).

[27] O. Zimmermann, U. Zdun, T. Gschwind, F. Leymann, 패턴 언어와 아키텍처 의사 결정 모델을 포괄적이고 이해하기 쉬운 설계 방법으로 결합. 프로시저.. IEEE/IFIP WICSA 2008 의. IEEE 컴퓨터 협회, 로스 알라미토스(2008).