DOI: 10.15514/ISPRAS-2023-35(6)-4

Онтология архитектурных знаний в совместно локализованной “живой” среде

[1] *Х.Л. Роблес*, ORCID: 0000-0002-0695-1021 <*jose.robles99633@potros.itson.edu.mx*>
[2] *Х. Боррего*, ORCID: 0000-0001-7315-5693 <*gilberto.borrego@itson.edu.mx*>
[2] *Р.Р. Паласио*, ORCID: 0000-0002-4059-2149 <*ramon.palacio@itson.edu.mx*>
[3] *Ф. Кастильо-Баррера*, ORCID: 0000-0001-7821-5819 <*ecastillo@uaslp.mx*>

[1] *Технологический институт Соноры,*
Мексика, штат Сонора, Сьюдад-Обрегон.
[2] *Технологический институт Соноры,*
Мексика, штат Сонора, Навохоа.
[3] *Автономный университет Сан-Луис-Потоси,*
Мексика, Сан-Луис-Потоси.

Аннотация. Небольшие компании, ведущие “живую” разработку программного обеспечения, сталкиваются с новой реальностью удаленной разработки. В удаленном взаимодействии создаётся множество видеозаписей, извлекаемых из видеозвонков и используемых в последующей работе. Записанные видеозвонки содержат архитектурные знания, собранные на виртуальных встречах, они важны для компаний, сталкивающихся с проблемой испарения знаний. Однако в литературе редко встречаются предложения по управлению архитектурным знанием, имеющимся в видеозаписях. В статье предлагается решение для восстановления архитектурных знаний, содержащихся в видео, которое, следуя концепции конденсации архитектурных знаний, в качестве классификационной схемы использует онтологию. В соответствии с руководящими принципами Methontology была проведена валидация предложений по управлению архитектурными знаниями. Внедрение онтологии как схемы классификации представляет собой шаг вперед к достижению консолидации архитектурных знаний в гибкой среде разработки для микропредприятий.

Ключевые слова: микропредприятия; живая разработка; архитектурное знание; конденсация знания.

Для цитирования: Роблес Х. Л., Боррего Х., Паласио Р. Р., Кастильо-Баррера Ф. Онтология архитектурных знаний в совместно локализованной agile-среде. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 75–94. DOI: 10.15514/ISPRAS–2023–35(6)–4.

Благодарности: Эта исследование стало возможным, благодаря Мексиканскому национальному совету по науке и технологии CONACYT, Технологическому институту Соноры и Автономному университету Сан Луис Потоси.

Ontology for Architectural Knowledge Condensation in a Co-Localized Agile Environment

[1] *J.L. Robles*, ORCID: 0000-0002-0695-1021 <*jose.robles99633@potros.itson.edu.mx*>
[2] *G. Borrego*, ORCID: 0000-0001-7315-5693 <*gilberto.borrego@itson.edu.mx*>
[2] *R.R. Palacio*, ORCID: 0000-0002-4059-2149 <*ramon.palacio@itson.edu.mx*>
[3] *F. Castillo-Barrera*, ORCID: 0000-0001-7821-5819 <*ecastillo@uaslp.mx*>

[1] *Instituto Tecnológico de Sonora*,
Ciudad Obregón, Sonora, Mexico.
[2] *Technological Institute of Sonora*,
Navojoa, Sonora, Mexico.
[3] *Autonomous University of San Luis Potosi*,
San Luis Potosi, Mexico

Abstract. Agile software development companies considered small entities (VSE) face a new reality of remote development. Remote communication has generated many videos derived from video calls recorded for later reference. The video calls recorded contains architectural knowledge from virtual meetings and is essential for companies facing the knowledge vaporization problem. However, only some proposals in the literature can potentially manage AK in videos. This article proposes a solution to recover this architectural knowledge contained in videos using an ontology as a classification scheme, following the architectural knowledge condensation concept. We validated our proposal to manage architectural knowledge following the Methontology guidelines. Implementing an ontology as a classification scheme represents a step forward to achieving the condensation of architectural knowledge in an agile development environment for VSE.

Keywords: very small entities; agile development; architectural knowledge; condensation knowledge.

For citation: Robles J. L., Borrego G., Palacio R. R., Castillo-Barrera F. Ontology for architectural knowledge condensation in a co-localized agile environment. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 75-94 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-4.

Acknowledgements. This research work was possible thanks to CONACYT, the Instituto Tecnológico de Sonora, and the Universidad Autónoma de San Luis Potosí.

1. Введение

В настоящее время в компаниях[1] все чаще внедряются принципы “живой” (agile) разработки программного обеспечения. Ее популярность обусловлена в основном возможностью быстрого получения результатов [1] и тем, что работающая программа важнее сложной документации [2]; кроме того, люди и их общение важнее процессов и инструментов, присущих процессу разработки [2]. Однако чрезмерное внимание к кодированию может привести к пробелам в документации, которые представляют собой разновидность технического долга (technical debt), возникающего из-за пренебрежения или приостановки действия одного или нескольких активов, связанных с процессом разработки, то есть из-за недостаточности документации по разработке [3], часто оставляемой в незаконченном виде из-за нежелания тратить время и силы [4]. Такая ситуация влияет на архитектурное знание (Architectural Knowledge, AK), которое строится из архитектурных проектов и решений с их обоснованиями, на основе которых определяется архитектура программного обеспечения [5].

Отсутствие документации по архитектурному знанию может быть результатом следования ценностям манифеста живой разработки [2], и часто воспринимается разработчиками как проблема только после наступления последствий, например, таких, как потеря или “испарение” архитектурного знания [6]. Среди распространенных последствий “испарения”

[1] *https://info.digital.ai/rs/981-LQX-968/images/SOA15.pdf*

можно назвать следующие: неправильно понятые требования, задержки в процессе разработки и отсутствие технического видения проекта [4, 7-8].

Кроме того, в “живой” разработке предпочтение отдается личному взаимодействию, поскольку оно считается наиболее эффективным и действенным способом передачи информации внутри команды разработчиков [2] в общей для них коммуникационной среде [9]. Таким образом, следование манифесту может привести к преобладанию неявного архитектурного знания. Согласно Нонаке и Такеучи [10], для эффективного обмена знаниями необходимо пройти четыре этапа модели создания знания (SECI). Сюда входят следующие этапы:

- Обобществление (Socialization), когда архитектурное знание передается другому человеку устно (оставаясь неявным),
- Проявление (Externalization), когда архитектурное знание передается путем записи в нестандартные документы, становясь явным,
- Сборка (Combination), когда документы и артефакты формализуются путем применения правил, форматов или протоколов, облегчающих их интерпретацию (например, UML [11]), и
- Отчуждение (Internalization), когда документы и артефакты с формальным архитектурным знанием используются для передачи знаний другим людям, инициируя новый цикл.

Поскольку в “живой” разработке создание документации отходит на второй план, этап сборки знаний не доводится до конца, следовательно, архитектурное знание не формализуется [12]. Складывающаяся ситуация не дает возможности эффективно обмениваться знаниями, а это приводит к их “испарению” [6] и, как следствие, к появлению незрелого или неполного архитектурного проекта [13], который не соответствует уровню качества, необходимому для полноценной разработки программ [14-15]. Эти проблемы возникают из-за неявности управления архитектурным знанием или из-за неформальностей, либо чрезмерной специализации обозначений [16], допущенных в документации (например, в чертежах, технических спецификациях, диаграммах, эскизах).

Несмотря на это, неявное архитектурное знание считается очень важным, благодаря опыту разработчиков, который необходим для принятия архитектурных решений [17] и тоже является источником знаний для разработчиков. Однако если архитектурное знание не формализовано, то в будущем оно не сможет быть правильно понято, а значит, в итоге становится ненадежным источником архитектурного знания [7].

Отсутствие надежных источников архитектурного знания приводит к взаимозависимости в команде разработчиков при решении вопросов о программных проектах, создавая “склады знаний”, то есть взаимозависимость с опытными разработчиками (или старшими по должности в компании) [18]. Такая взаимозависимость часто становится раздражителем для разработчиков, являющихся источником архитектурного знания, поскольку им приходится много раз отвечать на одни и те же вопросы. Это приводит к разрушению личных отношений и негативно сказывается на общении разработчиков [19]. Возможным выходом из этой ситуации могло бы стать включение в практику формализация архитектурного знания (например, использование унифицированного языка моделирования (UML) [11]), однако, это может противоречить правилам “живой” разработки [20] и желаниям разработчиков [21]. Другим возможным решением может быть применение специальной среды разработки, например, среды Scaled Agile Framework (SAFe[2]), для работы в которой явно назначается ответственный за разработку архитектуры. Для крупных компаний, в силу их высокой штатной численности, это вполне осуществимо. В отличие от крупных компаний, небольшие

[2] *https://www.scaledagileframework.com*

компании или отделы разработки программного обеспечения (фактически, микропредприятия – Very Small Entities, VSE) не могут применять такие подходы из-за ограниченного количества сотрудников (согласно ISO/IEC-29110-4-2[3], микропредприятия имеют в штате не более 25 сотрудников). Это представляет собой существенную проблему, поскольку только на американском рынке 99% предприятий относятся к малому бизнесу (в том числе и микропредприятиям)[4]. Тем самым, микропредприятия полностью подвержены проблеме испарения архитектурного знания.

Таким образом, из-за нежелания разработчиков документировать, а также из-за того, что применение сред разработки или методов формального представления архитектурного знания для микропредприятий не представляется возможным, проявляется тенденция поиска архитектурного знания в репозиториях и на известных платформах, например, на Stack Overflow или Github [22-23]. Другой распространенной практикой является постоянный анализ исходного кода проектов с целью поиска архитектурного знания [24], однако этот процесс крайне медленный, неэффективный и чреват наличием ошибок, поскольку получаемые таким образом знания, как правило, не структурированы, неполны и непоследовательны. Таким образом, в обоих случаях эти методы могут нарушить привычный рабочий процесс разработчика и, следовательно, замедлить его, а проблема испарения останется латентной.

В последнее время появилась новая возможность уменьшить испарение архитектурных знаний. Поскольку среди компаний, ведущих живые разработки стал популярен удаленный режим работы (к этому подтолкнули ограничения из-за распространения COVID-19 [25]), увеличилось количество виртуальных встреч, и эти видеозвонки часто записываются для последующего использования [26]. В этих записях может содержаться значительное архитектурное знание из сессий с вопросами-ответами (QA sessions) и периодических совещаний, на которых решаются важные вопросы и принимаются архитектурные решения. Однако извлечение необходимых знаний может быть затруднено из-за отсутствия механизма организации и классификации видеозаписей, полученных в ходе виртуальных встреч; таким образом, эти знания тоже подвержены риску испарения. Такая ситуация грозит стать постоянной, поскольку многие компании, занимающиеся разработкой программного обеспечения, решили продолжить работу в удаленном режиме, ощутив новые возможности и преимущества такого способа организации работы [27-28].

Чтобы облегчить извлечение знаний из записей виртуальных встреч может оказаться полезной концепция конденсации этих знаний [12]. Эта концепция заключается в сборе архитектурных знаний, разбросанных на различных носителях, их классификации и облегчении доступа к ним с помощью поискового механизма [12]. Однако знания часто распространяются на различных носителях, платформах, в виде различных артефактов, что делает их классификацию сложной [29]. Знания могут распространяться и в виде записей видеозвонков. Реализация механизма классификации архитектурных знаний облегчила бы реализацию концепции конденсации знаний, а согласно работе [29], в онтологиях эта реализация действительно возможна.

В этой работе мы предлагаем цикл конденсации архитектурного знания, основанный на исходной концепции конденсации [12] и следующий рекомендациям модели SECI [10] и Интегрированного цикла управления знаниями (Integrated knowledge management cycle [30]). Цикл конденсации архитектурного знания включает в себя три этапа:

(I) Сбор информации (Acquisition), при котором из различных источников усваивается архитектурное знание, которое тем самым приобретается людьми.

[3] *https://www.iso.org/standard/77735.html*

[4] *https://www.forbes.com/advisor/business/small-business-statistics/*

(II) Классификация (Classification), при которой собранные архитектурные знания классифицируются по определенной схеме, и

(III) Объединение знаний (Sharing), при котором архитектурное знание передается другим людям для начала нового цикла (см. рис. 1).

Из рис. 1 видно, что некоторые активности напрямую связаны с конкретным состоянием модели SECI, например, групповая виртуальная встреча, предполагающая групповое взаимодействие посредством видеозвонка, где архитектурное знание записью видеозвонка трансформируется из неявного в явное. Запись видеозвонка становится источником знания, эта запись может быть использована для получения и классификации знания, а впоследствии также и для обмена знаниями. Следуя этим рекомендациям, мы использовали цикл конденсации знаний для анализа возможностей такой конденсации применительно к конкретной работе, для определения этапов цикла, которые можно реализовать, и для управления сферой применения архитектурного знания. Сформулировав наши предложения по циклу конденсации знаний и проанализировав предложения, опубликованные другими авторами, мы, стремясь добиться полноценной конденсации знаний, предложили решение, которое соответствует руководящим принципам цикла конденсации архитектурных знаний, ориентированным на видеозаписи, создаваемые по видеозвонкам и размещаемыми на различных платформах, содержащих такие знания. Наше предложение основано на концепции онтологий, которую мы рассматриваем в качестве неотъемлемой части механизма классификации архитектурных знаний. Целью этой статьи является реализация онтологии, способной организовать и облегчить поиск этих видеозаписей, в независимости от их происхождения и формата, содействуя поиску и доступу к знаниям в них, следуя рекомендациям цикла конденсации архитектурных знаний. Мы представляем сценарий нашего онтологического решения, чтобы пояснить реализацию концепции конденсации знаний при помощи механизма классификации, умеющего размечать видеозаписи (например, такого, как система социальных тегов), обеспечивая управление этими записями и доступ к ним путем реализации механизма поиска.

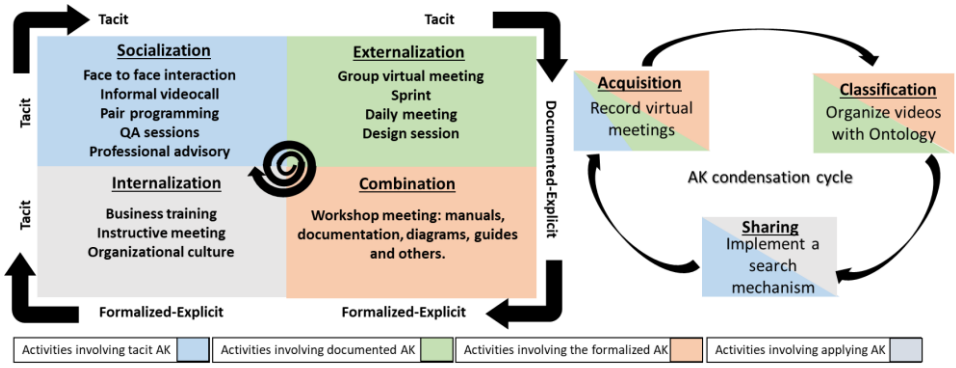

Рис. 1. Взгляд на "живые" активности модель SECI и цикл конденсации. Отображено, какие состояния знаний рассматриваются на каждом цикле конденсации
Fig 1. View of the agile activities related to the state of knowledge through the SECI model and the condensation cycle. We show which knowledge states are considered in each phase of the condensation cycle

Остальная часть статьи построена следующим образом. В разделе 2 представлены смежные работы, посвященные механизмам обмена архитектурными знаниями, использованию видеозаписей для обмена знаниями и предложению новой онтологии. В разделе 3 представлены наши предложения по сценарию решения поставленной задачи. В разделе 4 представлена методология построения предложенной онтологии. В разделе 5 представлены

результаты, полученные в соответствии с этой методологией. В разделе 6 представлено обсуждение полученных результатов, а в разделе 7 – выводы и предложения по дальнейшим работам.

2. *Смежные* работы

Конденсация архитектурного знания – это процесс сбора и классификации знаний, снижающий риск их потери и обеспечивающий доступ к ним для их совместного использования. В ходе данного исследования мы проанализировали работы по управлению архитектурным знанием через поддержку платформ как источников знаний, ввели цикл конденсации знаний, основанный на оригинальной концепции конденсации архитектурного знания [12] и интегрированном цикле управления знаниями Далкира [30], включающем состояния архитектурного знания модели SECI [10], чтобы объяснить трансформацию знаний на каждом этапе (см. рис. 1).

2.1 Предложения по управлению архитектурным знанием

Управление архитектурным знанием основано на спирали знаний [10] и имеет состояния, в которых знания существуют в неявном (I) и явном (II) видах. Однако согласно модели SECI [10], анализируя активности на каждом этапе, можно выделить “явно документированные” подсостояния архитектурного знания, когда это знания могут существовать в неформализованных документах или в виде конспективных записей (например, заметки на наклейках). Существует также “явно формализованные” подсостояния, когда правила или форматы обеспечивают создание и интерпретацию артефактов и документов (например, UML [11]).

В работе [31] предложена схема, характеризующая потерю знаний, вызванную текучкой кадров. Они выделяют полную потерю, когда носитель неявных знаний отсутствует постоянно, и частичную, когда она носит временный или внезапный характер. Они описывают 20 последствий потери знаний, объединенных в четыре темы: отсутствие руководства, опора на документацию, работа с коллегами и воссоздание знаний. Авторы рекомендуют своевременно оформлять документацию для предотвращения потери знаний, привлекая коллег к обсуждению проблем и стратегий их решения. По сути работа представляет собой руководство по документированию для предотвращения потери знаний.

В работе [32] предложена программная среда для управления проектными решениями по архитектуре крупных программных проектов. Эта среда предназначена для решения конкретных задач – извлечения и классификации проектных решений, аннотирования архитектурных элементов, выработки рекомендаций по альтернативным вариантам решений и назначения экспертов для принятия проектных решений. Для реализации этих задач необходимо проводить планирование и оценку, итеративно используя при этом научный, исследовательский подход к проектированию. Среда позволяет соотнести систему управления архитектурными знаниями с этапами проектирования, разработки и сопровождения. Она помогает заинтересованным сторонам документировать проектные решения и извлекать уроки из уже выполненных проектов.

С другой стороны, в работе [33] предлагается инструмент для принятия архитектурных решений, которым документируются решения и проблемы заинтересованных сторон. Инструмент записывает решения по пяти позициям: взаимоотношения, вовлечение заинтересованных сторон, сила, хронология и детали. Это помогает разработчикам архитектуры анализировать и управлять решениями, однако для его более эффективного использования заинтересованным сторонам на адаптацию может потребоваться время. Для небольших команд, ведущих живые разработки с прямым общением, процесс документирования можно упростить, что сократит количество решений. В работе [34] предлагается метод документирования архитектурного знания, названный подходом C3A

(CA[5] Agile Architecture) и предназначенный для фаз выработки архитектурных решений и проектирования жизненного цикла разработки программного обеспечения. Предлагаемый авторами метод основан на использовании эталонной архитектуры. Он заключается в регистрации шаблона для написания характеристик каждого программного компонента с описанием каждого элемента, необходимого для развития процесса разработки. Авторы приходят к выводу о том, что архитектура CSA позволяет сопоставлять процесс разработки с элементами системной архитектуры. Такой подход может помочь устранить разрыв между стратегическим мышлением организации, занимающейся разработкой, и тактикой проводимой ею "живой" реализации. Аналогичным образом, система поддержки принятия решений по разработке архитектуры (Architectural Design Decision Support System, ADDSS) представляет собой веб-инструмент, который помогает создавать и документировать архитектурные проектные решения и их связи с другими программными артефактами [35]. В работе [36] предлагается система Knowledge Trace Retrieval (KTR), предназначенная для извлечения элементов решения проектных задач из деловой электронной почты. Хотя инструменты и методы управления знаниями являются промышленными стандартами, их использование в небольших проектах не является регулярным. Система помогает пользователям извлекать следы знаний из большого объема различных электронных писем, полученных в рамках прошлых проектов. Для поиска знаний система использует расширенный контекст (данные о проекте, компетенции и профили пользователей), методы машинного обучения и алгоритмы ранжирования. Аналогичным образом в работах [37-38] для сокращения количества документации предложен метод репертуарной сетки (Repertory Grid Technique, RGT), который помогает предотвратить потерю архитектурного знания за счет документирования архитектурных решений.

Для решения проблемы фиксации и обмена архитектурными решениями в работе [39] представлены высокоуровневые истории проектирования (High-level Design stories, HLDs), которые представляют собой небольшие артефакты, содержащие архитектурные решения проекта, контекст, предположения, анализ качественных характеристик и нерешенные вопросы. Истории HLDs являются модульными, они создаются и уточняются в ходе архитектурно-ориентированного процесса разработки для подтверждения решений и проблем, помогая получить подробный глобальный проект архитектуры. Еще одним предложением по поиску архитектурного знания является дополнение Slack под названием TaggerBot [40], которое представляет собой виртуального тематического помощника, призванного уменьшить потерю знаний, неявно возникающих при разработке программного обеспечения. Помощник использует следы архитектурного знания в сообщениях и взаимодействиях через платформу Slack и реализует механизм классификации знаний на основе социальных тегов. Однако, как и в более ранних предложениях, цель была сосредоточена в основном на документировании знаний.

По этой причине некоторые авторы предложили повторное использование знаний в качестве быстрого решения проблемы поиска знаний, особенно в процессе разработки. В работах [41-42] повторное использование кода рассматривается как необходимое для понимания проблем в процессе разработки программного обеспечения. Однако, хотя разработчики могут увеличить свои знания и принимать архитектурные решения на основе консультаций по текстам программ [24, 26], консультации по архитектурному знанию обусловлены необходимостью документирования [43, 24], вызванной задолженностью по документации [4]. Заинтересованные стороны постоянно ищут наилучший способ обмена и сохранения архитектурного знания [26], приспосабливая решения к своей среде разработки, например, записывая виртуальные встречи и обмениваясь видеороликами для ответов на вопросы. В этом смысле в работе [44] предлагается метод анализа различий между использованием

[5] *https://www.broadcom.com/*

видеоматериалов для обучения и подготовки к работе и ручного обучения без видеоматериалов. Анализ направлен на повышение производительности труда путем минимизации времени и затрат на обучение работников за счет использования видеоматериалов в качестве источника обучения. Результаты показали, что использование видеоматериалов в качестве источника обучения значительно повышает производительность труда, минимизируя время и стоимость обучения работников. По мнению авторов работы [45], видеоматериалы могут рассматриваться как эффективный инструмент обмена информацией в любой отрасли и могут способствовать обучению, развитию критического мышления и сотрудничеству. Хотя существуют работы, в которых использование видеоматериалов рассматривается как источник знаний и механизм коммуникации, большинство проанализированных предложений ориентировано на применение видеоматериалов для тренировок персонала или в образовательных целях, что далеко от потребностей микропредприятий, работающих в гибкой среде, где видеоматериалы необходимы как средство консультаций по вопросам архитектурного знания.

В табл. 1 приведены краткие сведения о предложениях авторов работ, рассмотренных в этом разделе. В столбце “Уровень знаний” указан уровень управления архитектурными знаниями в соответствии с моделью SECI [10]. В столбце “Проверено?” указано, сообщалось ли в той или иной статье о валидации предложений авторов. В столбце “Этап” указаны этапы цикла конденсации архитектурного знания, охваченные предложениями, где С – это сбор знаний, К – классификация знаний, а О – обмен знаниями.

Согласно модели SECI [10], все найденные работы по поддержке управления архитектурным знанием в живых средах используют это знание в *явно задокументированном* состоянии (см. табл. 1). Как утверждается в работе [20], это происходит так, поскольку в этих предложениях знание, как правило, фиксируется неформально: в живых средах механизмы такого рода считаются слишком навязчивыми из-за того, что заинтересованные стороны создают документацию во время, отведенное прежде всего на разработки, отвлекаясь от “живого” процесса. Несомненно, удаленная работа повлияла на интенсивность обмена архитектурным знанием в живых средах [46], по этой причине возникли усилия по его минимизации, однако они предпринимались только на одном или двух этапах (сбор/классификация) цикла конденсации знаний и не касались этапа обмена знаниями. Можно также отметить, что во многих предложениях не был указан размер компании, на которую оно было ориентировано. Некоторые из них прошли валидацию, показав целесообразность сбора и/или классификации архитектурных знаний.

Табл. 1. Классификация связанных работ
Table 1. Related work classification

Название предложения	Уровень знаний	Тип предложения	Вид компании	Проверено?	Этап
Робиллард [31]	Явно документированный	Фреймворк	Не указан	Нет	С
RGT [37-38]	Явно документированный	Технология	Любой	Да	С
C3A [34]	Явно документированный	Подход	Не указан	Нет	С
Бхат [32]	Явно документированный	Фреймворк	Крупный	Нет	С
ADDSS [35]	Явно документированный	Инструмент	Не указан	Да	С
KTR [36]	Явно документированный	Инструмент	Не указан	Да	С
Мантейфель [43]	Явно документированный	Модель	Не указан	Нет	С
HLD [39]	Явно документированный	Модель	Не указан	Нет	С
TaggerBot [40]	Явно документированный	Инструмент	Крупный	Да	С, К
Типпаннавар [44]	Явно документированный	Метод	Не указан	Нет	С
Веддер-Вайс [45]	Явно документированный	Обучение	Образовательный	Да	С

Согласно имеющимся в литературе данным, существуют предложения, непосредственно связанные с управлением архитектурными знаниями (рис. 2). Рассматривая этап сбора знаний цикла конденсации, можно найти предложения по механизмам управления неявных знаний (например, [37-38]), явных знаний (например, [35-36]), переиспользуемых знаний ([41-42]), которые предполагают обращение к прикладным знаниям [24] или ранее принятым решениям [43], подчеркивая использование этих артефактов в качестве источника знаний [26], как в случае записанных видеозвонков, к которым можно обратиться позже. Что касается применения механизмов классификации и организации, то для конденсации содержащейся в них информации необходимо иметь записи знаний, хранящиеся в репозиториях. Найденные предложения, потенциально способные выявить архитектурные знания, обычно относятся к стадиям сбора и классификации знаний (документированные архитектурные знания) и, как правило, нуждаются в развитии или реализации в реальном сценарии. Поэтому следующий этап – обмен архитектурными знаниями – обычно пропускается или выполняется с использованием неформальных и очень специальных документов [12, 16], что нарушает этап объединения знаний модели SECI [10], вызывает задолженность по документации [4] и приводит к испарению знаний.

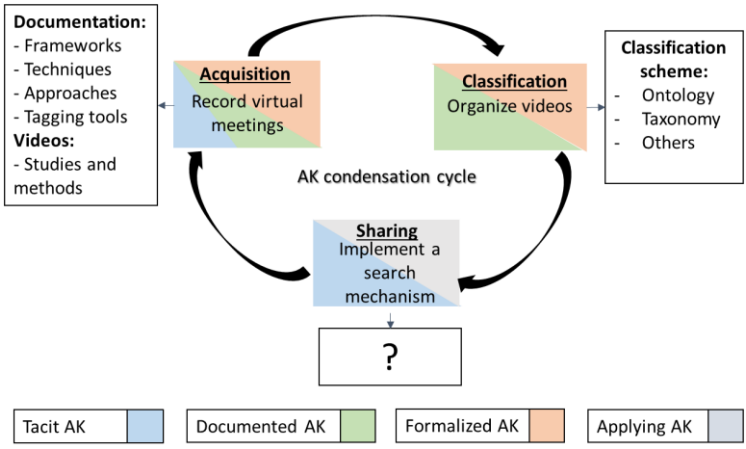

Рис. 2. Предложения по циклу конденсации знаний. Белые поля для каждого этапа цикла содержат предложения, представленные в смежных работах, посвященных соответствующему этапу. Предложения для этапа объединения знаний отсутствуют. В рамках каждого из этапов цикла конденсации показаны основные виды деятельности, связанные с нашим предложением, а также состояния и подсостояния, в которых находится архитектурное знание

Fig. 2. Proposals through the AK condensation cycle. The white boxes for each cycle phase contain proposals presented in the related work focused on the corresponding phase. There are no proposals for the Sharing phase. Within each of the phases of the condensation cycle, the main activities related to our

2.2 Онтологии как средство классификации архитектурных знаний

Одним из способов повторного использования архитектурного знания является применение онтологий, которые представляют собой формальное описание связанных друг с другом обобщенных понятий [47]. Онтология позволяет повторно использовать знания путем прояснения неоднозначностей [26] и способна облегчить процесс конденсации знаний, поскольку вводит словарь для обмена информацией и помогает принимать более конкретные архитектурные решения [48]. Онтологии возникают как инструмент управления знаниями, поддерживающий представление, обработку, хранение и извлечение знаний [49], что необходимо для разработки программного обеспечения [50].

Хотя некоторые из проанализированных здесь предложений направлены на сохранение знаний, в живой разработке программ сохранить его удается редко из-за нарушений цикла знаний [20], которые мешают формализации архитектурных знаний. Складывающаяся ситуация приводит к тому, что разработчики стремятся реализовать неявные механизмы управления знаниями или создают документы с неформальными обозначениями. Именно поэтому необходимость в механизме, формализующем документы и артефакты архитектурного знания, создаваемые в процессе живой разработки, побудила нас разработать онтологию, с помощью которой можно устанавливать семантические связи между документами и артефактами, независимо от их происхождения и формата. В работе [29] авторы в своем исследовании показывают, как онтология может классифицировать знания, помогая в поиске и сохранении знаний из различных источников, в организации знаний и их соотнесении с их создателями. Эта семантическая связь влияет на то, как разработчики принимают решения при возникновении различных вопросов или проблем. Поскольку разработчикам нужна надежная и точная информация [51], они обычно ищут эти решения в различных источниках знаний (помимо обращения к экспертам) [52]. В работе [49] предлагается онтология сбора знаний о среде разработки программного обеспечения для более успешного проведения его тестирования. Предложение касается тестировщиков и разработчиков, участвующих в тестировании программ, и включает базу знаний о контексте разработки, дающую возможность обращаться к этим знаниям, когда они необходимы, и способствующую принятию высокоэффективных решений, связанных с тестированием.

Можно сделать вывод о том, что будет интересно рассмотреть, каким образом онтология может способствовать конденсации архитектурных знаний, записанных в видеоформатах, которые, как правило, при проведении “живой” разработки содержат соглашения разработчиков и их важные решения, связанные с их деятельностью по разработке.

3. Предложение по управлению архитектурным знанием для микропредприятий, ведущих “живую” разработку

В этом разделе мы описываем проблему взрывного роста числа видеоматериалов и трудностей поиска архитектурных знаний на основе ранее проведенного экспериментального исследования [26]. Эта трудности проистекают из большого количества видеозаписей виртуальных встреч сотрудников микропредприятий, ведущих разработки по “живым” правилам.

Вместо того чтобы делать заметки или вести протоколы заседаний, чтобы сохранить информацию о проекте (принятые решения, ответы на вопросы, решенные проблемы и так далее), которая может содержать архитектурное знание, заинтересованные стороны ведут видеозаписи виртуальных встреч. Чтобы компенсировать недостаток личного общения, заинтересованные стороны обычно проводят много виртуальных встреч, а это приводит к появлению большого количества многочасовых видеозаписей. Поиск конкретного знания в этих файлах может быть трудным, поскольку заинтересованным сторонам приходится вести поиск во многих видеозаписях, во всех файлах, минута за минутой. Это приводит к тому, что заинтересованные стороны тратят слишком много времени, надолго прерывая свою основную деятельность по разработке [20]. Из-за необходимости просмотра большого количества проверяемых видеофайлов, заинтересованные стороны часто не могут найти нужную им информацию. Это обстоятельство может привести к тому, что информацию приходится запрашивать у коллег, отправляя им текстовые сообщения или даже делая видеозвонки (что усугубляет проблему), часто вызывая у них раздражение, поскольку часто запрашивается информация, уже обсуждавшуюся на прошедших встречах [20].

Основываясь на концепции конденсации архитектурного знания [12] и учитывая потребности микропредприятий в управлении знаниями, мы представляем предлагаемое нами решение с учетом удаленной “живой” разработки программного обеспечения [26], используя

преимущества записи виртуальных встреч, которые происходят в коллективах и в которых члены коллективов сохраняют важные разговоры, содержащие архитектурное знание. Первым шагом цикла конденсации знаний является получение этих знаний, поэтому видеозаписи обязательно должны размещаться в репозиториях (например, на Google Drive) с включенным общим доступом, позволяющим обращаться к видеозаписям из любого источника.

На виртуальных совещаниях заинтересованные стороны имеют возможность помечать (ставить теги) конкретные видеофрагменты, содержащие ценные знания, для последующего их поиска. Эти теги должны соответствовать существующим метатегам в классах онтологии, поскольку в качестве классификационной основы предлагаемое нами решение предполагает использование онтологии. Онтология представляет собой модель знаний, устанавливающую смысловые связи между тегами знаний и заранее определенной иерархической структурой. Таким образом, использование тегов, связанных с онтологией, позволяет автоматически организовать знания, спрятанное в видеофайлах виртуальных встреч. Благодаря использованию онтологии этап классификации цикла конденсации архитектурного знания выполняется достаточно эффективно.

Более того, присущие онтологии характеристики дают определенные преимущества для реализации механизмов поиска. Онтология позволяет осуществлять систематический поиск, охватывая всю онтологию посредством иерархических сущностей. Теги могут быть связаны с классами и подклассами, что повышает удобство запросов к онтологии. Интеграция механизма поиска на основе онтологии позволяет завершить цикл конденсации знаний, упростить управление знаниями за счет хорошо организованных видеоматериалов, содержащих потенциальное знание. Для реализации предложенного нами решения очень важно создать схему классификации, поэтому мы и выбрали онтологию. Разработка онтологии предполагает проведение сложного процесса построения абстракций, в ходе которого определяются отношения, основанные на требованиях управления знаниями к организации записанных виртуальных встреч, хранящихся в репозиториях. Следующим логическим шагом является создание онтологии для очерчивания области знаний, полностью охватывающей все этапы цикла конденсации знаний. Такой целостный подход направлен на реализацию эффективного управления знаниями в рамках живой разработки для микропредприятий.

4. Методика разработки онтологии

Для проведения данного исследования мы следовали пяти этапам, рекомендованным технологией Methontology [53] и работой [29]. Далее мы опишем каждый этап, выполняемые на них работы и ожидаемые от них результаты.

Этап спецификации состоит в определении требований к онтологии. Мы определили требования на основе ранее проведенного экспериментального исследования [26]. Мы составили спецификацию требований к онтологии (Ontology Requirements Specification Document, ORSD), в которой описали назначение, область применения, язык реализации, предполагаемых конечных пользователей и возможные варианты использования онтологии.

Этап концептуализации: мы организовали и структурировали информацию, получив модель знаний, представленную в виде таксономии.

Этап формализации: определив модель знаний, мы создали онтологию с помощью редактора онтологий Protégé, определили классы и реализовали семантические связи между ними, в результате чего получилась онтологическая сущность.

Этап оценки: мы провели верификацию и валидацию онтологии с помощью набора компетентностных вопросов (Competency Questions, CQ), который был определен в соответствии с рекомендациями Methontology и в соответствии с нашим сценарием. Мы решили использовать механизм логического вывода Pellet, учитывая положительные

результаты, полученные при оценке онтологий OWL [54]. В процессе валидации мы убедились, что онтология соответствует своему назначению, а в процессе верификации выяснили что она удовлетворяет необходимым нам требованиям и корректно функционирует.

Этап сопровождения начинается с расширения исходной онтологии, предложенной в работе [29]. Этот этап также включает в себя обновление онтологии в соответствии с изменениями в домене или потребностями управления видеоинформацией. В рамках изменений, реализованных при обслуживании онтологии, мы добавили новые свойства данных и объектов, сделав акцент на актуальности семантических отношений архитектурных знаний, содержащихся в видеофайлах. В итоге мы получили полный список данных и свойств объектов онтологии.

5. Результаты

В данном разделе представлены результаты выполнения каждого этапа описанной выше методики.

Этап спецификации: результаты этапа спецификации являются частью документа ORSD[6], который определяет назначение и область применения онтологии, тем самым устанавливая параметры и ограничения, которые она будет иметь в процессе разработки.

Этап концептуализации: в рамках результатов этапа концептуализации мы создаем таксономию, которая объясняет отношения между объектом архитектурного знания и некоторым членом команды разработчиков. В то же время команда в рамках конкретного проекта может потребовать или создать артефакты архитектурного знания. В полученной таксономии мы показываем элементы, которые определяют организованные объекты и придают смысл отношениям, которые они имеют друг с другом; например, член команды может иметь различные роли и опыт, артефакты могут иметь различные форматы и размещаться на различных платформах, а проекты, к которым эти артефакты относятся, могут вестись на разных языках программирования.

Этап формализации: на основе классификационной структуры таксономии с помощью инструмента Protégé мы определили следующие классы (см. рис. 3.A): Artifacts (представление характеристик артефакта и место его размещения), Layers (представление архитектурного слоя, к которому относятся артефакты), Project (представление типа проекта, к которому относятся артефакты) и Team (представление члена команды разработчиков, связанного с артефактами: создатель и консультант). Связи между классами онтологии осуществлялись с помощью свойств объектов (рис. 3.B).

Этап оценки: проведена валидация и верификация онтологии с использованием логического вывода Pellet (рис. 4.A) со следующими вопросами:

CQ1. Какие видеоролики создал “имя участника”?
CQ2. Какие видеоролики использовал “имя участника”?
CQ3. Сколько тегов создал “имя участника”?
CQ4. Где расположено видео “имя видео”?
CQ5. Сколько видеороликов было создано в проекте “название проекта”?
CQ6. Сколько видеороликов было создано членом команды “имя участника”?
CQ7. Сколько видеороликов создал “имя участника» в проекте “название проекта”?
CQ8. Где в “имя видео” находится “имя тега”?

[6] *https://drive.google.com/file/d/1TPeMuI7udG7rtsx2jQO20xxIjGakqQMT/view?usp=sharing*

Эти вопросы были получены из сценария, представленного в разделе 3, который был основан на ранее проведенном эксперименте с разработчиками живых проектов [26]. Для проверки целостности нашей онтологии и подтверждения логической структуры мы использовали раздел DL Query программы Protégé (рис. 4.B). Использование механизм логического вывода облегчает оценку целостности определенных классов и их семантических связей. На рис. 4 видно, что процесс валидации и верификации с использованием вопросов по компетенциям прошел успешно, это подтверждает согласованность структуры онтологии и ее готовность ответить на любой другой вопрос того же типа.

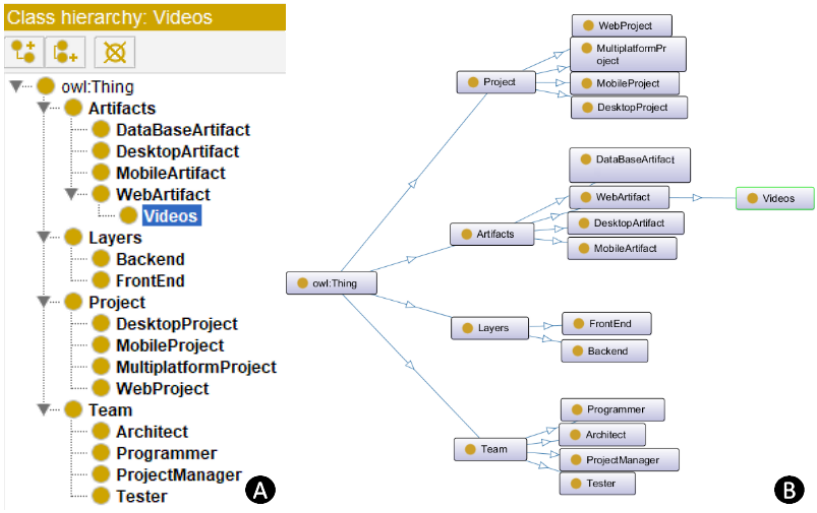

Рис. 3. Представления онтологии: (A) Вид иерархии классов. (B) Вид онтографа Protégé
Fig. 3. Ontology views: (A) Classes hierarchy view. (B) Protégé ontograph view

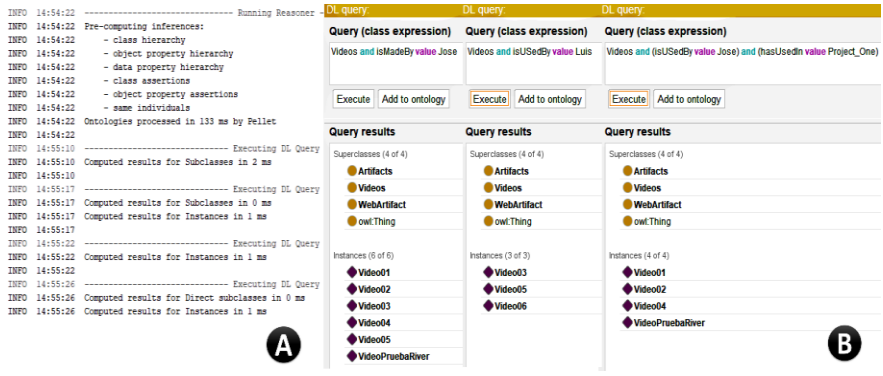

Рис. 4. Оценка онтологии:
(A) Результаты работы консоли системы Protégé. (B) Результаты DL-запросов
Fig. 4. Ontology evaluation: (A) Protégé system console results. (B) DL query results

Этап сопровождения: результаты этапа сопровождения начинаются с расширения онтологии, предложенного в работе [29]. Мы создали сущность “Video” как подкласс класса “WebArtifact”, чтобы классифицировать содержащее архитектурное знание видео, созданное на основе записей виртуальных встреч. Для класса “Programmer” мы создали суперкласс “Team” и добавили сущности “Architect”, “ProjectManager” и “Tester”, созданные в одной иерархии с классом “Programmer”, чтобы расширить возможности классификации и определить область архитектурного знания. Кроме того, в класс “Project” добавлены

подклассы “MultiplatformProject” для организации гибридных или мультиплатформенных проектов. Полная структура онтологии представлена на рис. 3.

Что касается свойств данных, то мы добавили новые атрибуты в “artifactDescription”. Мы добавили атрибут “artifactFormat” для определения формата видео, атрибут “artifactLocation” для определения платформы хранения видео, атрибут “artifactTag” для определения меток видеоконтента на временной шкале, а также атрибут “artifactCreationDate” для определения даты создания. Для свойства “projectLanguage” мы добавили атрибут “projectPlatform”, определяющий платформу проекта. Наконец, для свойства “userDescription” мы добавили атрибут “userRole” для определения ролей заинтересованных сторон в проекте. Для свойств объекта реализован атрибут “hasTaggedBy”, определяющий того, кто создает метку в видеоролике. Полный список данных и свойств объектов приведен ниже:

ArtifactDescription: *artifacLocation, artifactTag, artifactFormat, artifactLanguaje, artifactSubject, artifactCreationDate.*

ProjectDescription: *projectPlatform, projectName, projectLanguage*.

UserDescription: *userRole, userEmai, userExperience, userGende, userName.*

6. Обсуждение результатов исследования

Микропредприятия, работающие в стиле “живой” разработки, сталкиваются с серьезной проблемой, связанной с архитектурными знаниями, генерируемыми в ходе виртуальных встреч, которые обычно записываются в видеофайлы. Мы определили цикл конденсации знаний, с помощью которого смогли заметить, что большая часть опубликованных работ сосредоточена на предложениях, облегчающих документирование знаний и оставляющих в стороне усилия по классификации и облегчению доступа к знаниям, что необходимо для восстановления знаний, разбросанных на различных носителях, их классификации и облегчения доступа к ним, то есть конденсации знаний [12].

В ходе анализа литературы мы обнаружили предложения, направленные на смягчение последствий потери знаний. Такие предложения обусловлены важностью восстановления знаний, обычно размещенных на различных носителях. В качестве примера можно привести предложение СЗА [34] – артефакт, предназначенный для создания документации проектов на этапе разработки архитектуры. СЗА определяет, какие аспекты необходимо документировать, и для каждого аспекта определяет в документе отдельный раздел, что облегчает его создание. Следует, однако, помнить, что модель SECI [10] определяет состояние, называемое явно-документированным, которое относится к знаниям, зарегистрированным в нестандартных форматах; таким образом, СЗА не обеспечивает корректной интерпретации знаний в будущем, поскольку при его создании не соблюдаются никакие правила или протоколы. Со временем может возникнуть риск того, что архитектурное знание перестанут понимать или забудут, то есть оно испарится [12].

Напротив, предлагаемое нами решение использует в качестве механизма классификации онтологию и позволяет осуществлять регистрацию знаний в классифицированном и организованном виде, поскольку содержит формальную структуру, определяемую семантическими отношениями и иерархией классов, то есть онтологию, что позволяет осуществлять поиск знаний и облегчает доступ к ним путем реализации запросов через их классы. В отличие от работ, встречающихся в литературе, наше предложение позволяет придать регистрации знаний большую формальность, обеспечивая доступность знаний в случае необходимости. Это свойство снижает вероятность того, что знания будут утеряны и перестанут быть понятными, поскольку онтология позволяет семантически связать знания с рабочей средой микропредприятия, что создает контекст использования и происхождения знаний путем создания их иерархической организации. Таким образом, мы могли бы генерировать новые предложения по управлению знаниями, реализуя механизмы поиска,

ориентированные на документирование, классификацию и облегчение доступа к знаниям, хранящимся в проектах разработки программного обеспечения.

Интересным предложением для механизма классификации является помощник TaggerBot [40], в котором применен метод классификации архитектурного знания на основе таксономии. Однако в этой классификации рассматриваются только концепции, и проверить работоспособность такого механизма классификации можно только путем проведения оценок с помощью контролируемых экспериментов или в реальных сценариях, что требует значительных усилий и больших ресурсных затрат. Наше предложение может быть проверено инструментами логического вывода и компетентными вопросами, то есть проверка реализуемости онтологии требует меньших усилий, чем создание таксономии. Таким образом, онтология, составляющая суть нашего предложения, является базой для дальнейших запросов и выводов, подтверждающих возможность классификации новых типов артефактов.

Реализация нашей онтологии обеспечивает классификацию знаний, делая более удобным его извлечение. Однако остается необходимость реализации механизма разметки знаний тегами, подобный механизму, используемому в TaggerBot. Этот механизм должен быть основан на социальном тегировании и иметь ограничения, препятствующие взрывному росту числа тегов. В отличие от предложения TaggerBot, использующего реляционную базу данных, запросы ограничены конкретной архитектурой. Предлагаемое нами решение обеспечивает гибкость в реализации технологий и архитектур для хранения знаний. В то же время поддержка реляционной базы данных может потребовать значительных усилий. Кроме того, для сопровождения и поддержки онтологии нужны методики и средства управления изменениями и редактирования.

Аналогичным образом, видео является средством хранения и обмена знаниями [44-45]. Однако встречающиеся в литературе предложения пока не посвящены управлению их содержимым. Несомненно, богатство этого вида артефактов для микропредприятий имеет большое значение. Поэтому наше предложение может облегчить классификацию видеоконтента за счет использования тегов, например TaggerBot [40], для маркировки фрагментов видео, которые содержат интересные АЗ, а соответствующие вопросы АЗ могут решаться заинтересованными сторонами. При использовании нашего предложения заинтересованным сторонам не нужно будет просматривать множество видеоматериалов, чтобы найти конкретный фрагмент АЗ, поскольку искомое АЗ можно будет найти без особых усилий, так как оно уже будет помечено и классифицировано. Таким образом, АЗ будут связаны с сущностями онтологии, оставляя АЗ доступным для консультации и обсуждения любой из заинтересованных сторон. Предложенная онтология обеспечивает гибкость запросов, если они находятся в пределах области МСП.

7. Выводы и дальнейшая работа

В представленной работе рассматривалась необходимость извлечения архитектурных знаний, которые микропредприятиями хранятся в виде видеоматериалов. Важность данной работы заключается в том, что необходимо выявить видеофрагменты, содержащие необходимую информацию для проектов микропредприятий, поскольку из-за большого количества хранящихся видеозаписей и отсутствия протоколов или правил их сохранения и обеспечения доступа к ним в будущем найти необходимые знания, содержащиеся в видеозаписях, не так просто.

Определение цикла конденсации АЗ было необходимо для анализа литературы. Это позволило нам определить объем предложений с точки зрения управления архитектурными знаниями в живых средах. Определив цикл, мы обнаружили, что большинство работ сосредоточено на качестве только одного этапа цикла, который заключается в сборе знаний,

оставляя в стороне классификацию и доступ к этим знаниям. Мы поняли, что необходимо найти способ повысить эффективность реализации всех трех этапов цикла.

Для этого, используя формальные рекомендации, предложенные в [53], мы построили онтологию, а затем следовали этой методологии, стараясь оценить свою онтологию. В результате мы поняли, что онтология пригодна для классификации видеороликов и их содержания. В отличие от других предложений, известных по литературе, наше решение позволяет классифицировать и давать легкий доступ к видеофрагментам, в которых содержится соответствующая проектная информация, связанная с архитектурным знанием. Кроме того, в онтологию заложен потенциал не только для управления видео артефактами, но и для управления любыми артефактами с архитектурным знанием. Наше решение создает возможность извлечения знаний, содержащихся в видеоматериалах микропредприятий, тем самым предотвращая риск испарения знаний.

В данной работе мы представили разработку онтологии для решения проблемы классификации архитектурных знаний. Наше предложение также способствует реализации механизма поиска, позволяющего дополнить цикл конденсации знаний. Разработанная онтология была проверена с помощью инструментов логического вывода и компетентностных вопросов, что позволяет сделать вывод о том, что онтология полностью пригодна для реализации в соответствии с поставленными задачами, и заключить, что наше предложение может быть реализовано с использованием механизма классификации в виде онтологии, прошедшей валидацию. Кроме того, в смежных работах мы не нашли предложений, которые можно использовать в качестве онтологии для классификации знаний, которая могла бы быть реализована в виде системы. Таким образом, наше предложение может стать шагом вперед в поиске программных решений, решающих поставленную задачу управления знаниями. В качестве дальнейшей работы следующим шагом будет разработка программного комплекса, реализующего разработанную в данной работе онтологию, для проверки правильности модели знаний и наличия потенциала разработанной онтологии для управления знаниями. Определение механизма включает в нашу онтологию механизм разметки тегами и механизм поиска, которые облегчают реализацию всех этапов цикла конденсации архитектурных знаний.

По этой причине необходимо найти механизм, который сопряжет нашу онтологию с механизмом тегирования и механизмом поиска, которые поспособствуют реализации всех этапов цикла конденсации знаний. В рамках дальнейшей работы мы также рассматриваем возможность внедрения генератора слов для системы тегирования, чтобы помочь заинтересованным сторонам в создании тегов, и подчеркиваем, что для стимулирования создания тегов заинтересованными сторонами может потребоваться внедрение социальной системы тегирования. Наконец, для выполнения всех этапов цикла конденсации архитектурных знаний необходима разработка веб-системы, использующей разработанную в данной работе онтологию.

Список литературы / References

[1]. P. Jain, A. Sharma, and L. Ahuja, “The Model for Determining Weight Coefficients of Maintainability Criteria in Agile Software Development Process,” in 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1–4.

[2]. K. Beck et al., “Manifesto for Agile Software Development,” The Agile Alliance, 2001.

[3]. W. Cunningham, “The WyCash portfolio management system,” in Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, 1992.

[4]. E. Tom, A. Aurum, and R. Vidgen, “An exploration of technical debt,” J. Syst. Softw., 2013.

[5]. P. Kruchten, P. Lago, and H. Van Vliet, “Building up and reasoning about architectural knowledge,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2006.
[6]. J. Bosch, “Software architecture: The next step,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2004.
[7]. H. Holz and F. Maurer, “Knowledge management support for distributed agile software processes,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2003.
[8]. A. R. N. Uikey, U. Suman, “A Documented Approach in Agile Software Development,” Int. J. Softw., vol. Vol. 2, no, pp. 13– 22., 2011.
[9]. R. K. Kavitha and M. S. I. Ahmed, “A knowledge management framework for agile software development teams,” in Proceedings of 2011 International Conference on Process Automation, Control and Computing, PACC 2011, 2011.
[10]. I. Nonaka and H. Takeuchi, “Knowledge-Creating Company,” Knowledge-Creating Co., 1995.
[11]. P. Stevens and R. J. Pooley, Using UML: Software Engineering with Objects and Components. Addison-Wesley, 2006.
[12]. G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaíno, and F. O. García, “Towards a reduction in architectural knowledge vaporization during agile global software development,” Inf. Softw. Technol., 2019.
[13]. Z. Li, P. Liang, and P. Avgeriou, “Architectural Debt Management in Value-Oriented Architecting,” in Economics-Driven Software Architecture, 2014.
[14]. S. Fraser et al., “Technical Debt: From Source to Mitigation,” in Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, & Applications: Software for Humanity, 2013, pp. 67–70.
[15]. Y. Guo and C. Seaman, “A Portfolio Approach to Technical Debt Management,” in Proceedings of the 2nd Workshop on Managing Technical Debt, 2011, pp. 31–34.
[16]. R. Hoda, J. Noble, and S. Marshall, “How much is just enough?,” 2010.
[17]. R. K. Jonkers and K. E. Shahroudi, “Reducing the Costs of Engineering Design Changes Through Adoption of a Decision Support and Knowledge Management System Early in the Design,” in 2019 IEEE International Systems Conference (SysCon), 2019, pp. 1–8.
[18]. L. S. D. Annunzio and T. Sy, “Challenges and Organizations : Top-Level and Mid-Level Managerss’ Perspectives,” in Human Resource Planning Society, 2006.
[19]. S. Ryan and R. V. O’Connor, “Acquiring and Sharing tacit knowledge in software development teams: An empirical study,” Inf. Softw. Technol., 2013.
[20]. G. Bortis, “Informal Software Design Knowledge Reuse,” in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, 2010, pp. 385–388.
[21]. J. Burge and D. C. Brown, “Reasoning with Design Rationale,” in Artificial Intelligence in Design ’00, 2000.
[22]. B. Vasilescu, V. Filkov, and A. Serebrenik, “StackOverflow and GitHub: Associations between software development and crowdsourced knowledge,” in Proceedings - SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013, 2013.
[23]. J. Tantisuwankul et al., “A topological analysis of communication channels for knowledge sharing in contemporary GitHub projects,” J. Syst. Softw., 2019.
[24]. B. Selic, “Agile documentation, anyone?,” IEEE Softw., 2009.
[25]. M. Sako, “From remote work to working from anywhere,” Communications of the ACM. 2021.
[26]. Jose L. Robles; Gilberto Borrego; Ramon Palacio;, “Gestión del conocimiento arquitectónico posterior a COVID-19 en pequeñas entidades de desarrollo de software: un estudio cualitativo,” Abstr. Appl. 36 63 – 77, vol. 36, no. ISSN2007-2635, pp. 63–77, 2022.
[27]. A. Ozimek, “Remote Workers on the Move,” SSRN Electron. J., 2021.
[28]. A. Cetrulo, “Is remote working here to stay? Lessons and ideas for a post-pandemic future,” Sinappsi, 2021.

[29]. J. R. Martínez-García, F. E. Castillo-Barrera, R. R. Palacio, G. Borrego, and J. C. Cuevas-Tello, "Ontology for knowledge condensation to support expertise location in the code phase during software development process," IET Softw., 2020.
[30]. K. Dalkir, Knowledge management in theory and practice. 2013.
[31]. M. P. Robillard, "Turnover-Induced Knowledge Loss in Practice," in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021, pp. 1292–1302.
[32]. M. Bhat, K. Shumaiev, and F. Matthes, "Towards a Framework for Managing Architectural Design Decisions," in Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings, 2017, pp. 48–51.
[33]. C. Manteuffel, D. Tofan, P. Avgeriou, H. Koziolek, and T. Goldschmidt, "Decision architect - A decision documentation tool for industry," in Journal of Systems and Software, 2016.
[34]. E. Hadar and G. M. Silberman, "Agile Architecture Methodology: Long Term Strategy Interleaved with Short Term Tactics," in Companion to the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications, 2008, pp. 641–652.
[35]. M. Che, "An Approach to Documenting and Evolving Architectural Design Decisions," in Proceedings of the 2013 International Conference on Software Engineering, 2013, pp. 1373–1376.
[36]. R. Francois, M. Nada, and A. Hassan, "How to Extract Knowledge from Professional E-Mails," in 2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 2015, pp. 687–692.
[37]. D. Tofan, M. Galster, and P. Avgeriou, "Reducing architectural knowledge vaporization by applying the repertory grid technique," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2011.
[38]. D. Tofan, M. Galster, and P. Avgeriou, "Capturing tacit architectural knowledge using the repertory grid technique (NIER track)," in Proceedings - International Conference on Software Engineering, 2011.
[39]. J. A. Diaz-Pace and A. J. Bianchi, "High-Level Design Stories in Architecture-Centric Agile Development," in 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), 2019, pp. 137–144.
[40]. G. Borrego, G. Salazar-Lugo, M. Parra, and R. Palacio, "Slack's knowledge classification mechanism for architectural knowledge condensation," in Proceedings - 6th Annual Conference on Computational Science and Computational Intelligence, CSCI 2019, 2019.
[41]. Q. Jiang, C. H. Tan, C. L. Sia, and K. K. Wei, "Followership in an open-source software project and its significance in code ReUse," MIS Q. Manag. Inf. Syst., 2019.
[42]. A. P. Koenzen, N. A. Ernst, and M. A. D. Storey, "Code Duplication and Reuse in Jupyter Notebooks," in Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC, 2020.
[43]. C. Manteuffel, P. Avgeriou, and R. Hamberg, "An exploratory case study on reusing architecture decisions in software-intensive system projects," J. Syst. Softw., 2018.
[44]. R. P. Tippannavar, V. N. Kulkarni, and V. N. Gaitonde, "Productivity Improvement at Actuator Assembly Section Using Manual and Video Work Study Techniques," in Lecture Notes in Mechanical Engineering, 2020.
[45]. D. Vedder-Weiss, A. Segal, and A. Lefstein, "Teacher Face-Work in Discussions of Video-Recorded Classroom Practice: Constraining or Catalyzing Opportunities to Learn?," J. Teach. Educ., 2019.
[46]. J. Spatero, "2 Years of Digital Transformation in 2 Months," Microsoft 365, 2020.
[47]. C. W. Shiang, F. S. Tee, A. A. Halin, N. K. Yap, and P. C. Hong, "Ontology reuse for multiagent system development through pattern classification," Softw. - Pract. Exp., 2018.
[48]. T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," Int. J. Hum. - Comput. Stud., 1995.

[49]. S. Vasanthapriyan, J. Tian, D. Zhao, S. Xiong, and J. Xiang, “An ontology-based knowledge management system for software testing,” in Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, 2017.

[50]. S. Vasanthapriyan, J. Tian, and J. Xiang, “A Survey on Knowledge Management in Software Engineering,” Proc. - 2015 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS-C 2015, pp. 237–244, 2015.

[51]. N. Bin Ali, “Is effectiveness sufficient to choose an intervention?: Considering resource use in empirical software engineering,” in International Symposium on Empirical Software Engineering and Measurement, 2016.

[52]. S. Farshidi, S. Jansen, R. de Jong, and S. Brinkkemper, “A decision support system for software technology selection,” J. Decis. Syst., 2018.

[53]. M. Fernandez, A. Gómez-Pérez, and N. Juristo, “Methontology: from ontological art towards ontological engineering,” in Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, 1997.

[54]. J. A. Khan and S. Kumar, “OWL, RDF, RDFS inference derivation using Jena semantic framework & pellet reasoner,” in 2014 International Conference on Advances in Engineering and Technology Research, ICAETR 2014, 2014.

Информация об авторах / Information about authors

Хосе Луис РОБЛЕС – магистр инженерных наук Научно-исследовательского центра высшего образования в Энсенаде. С 2020 года аспирант Технологического института Соноры по отделению компьютерных наук. Сфера научных интересов: программная инженерия, проектирование и архитектура, управление знаниями, живая разработка, веб-разработка, приложения онтологий в программной инженерии.

José Luis ROBLES – Master in Engineering Sciences by the Centro de Investigación Científica y de Educación Superior de Ensenada. Currently, he is a PhD student in the Department of Computer Science and Design at the Instituto Tecnológico de Sonora since 2020. His research interests include software engineering, design and architecture, knowledge management, agile development, web development, and applications of Ontologies in Software Engineering.

Хильберто БОРРЕГО – доктор наук, с 2019 года – профессор Технологического института Соноры. Сфера научных интересов: процессы программной инженерии, парадигма “живой” разработки, управление знаниями, архитектура программного обеспечения.

Gilberto BORREGO - Doctor in Sciences, Full-time Professor at the Technological Institute of Sonora, since 2019. He works on improvements in the software engineering processes particularly on the agile paradigm. Research interests: Agile software engineering, knowledge management, software architecture.

Рамон Рене ПАЛАСИО – доктор наук, с 2004 года – профессор Технологического института Соноры. Исследует процессы текущей производственной практики, проектирует, разрабатывает, оценивает технологии, пригодные для различных видов операционного окружения. Сфера научных интересов: программная инженерия, взаимодействие человек-машина, интернет вещей.

Ramón René PALACIO – Doctor in Sciences, Full-time Professor at the Technological Institute of Sonora, since 2004. He conducts studies to gain a better understanding of current work practices and, based on this understanding, designs, develops, and evaluates technologies appropriate for various work environments. Research interests: Software engineering, Human-computer interaction, and Internet of Things.

Франсиско КАСТИЛЬО-БОРРЕРА - доктор наук в области информационных технологий, с 2002 года профессор инженерной школы факультета компьютерных наук Автономного университета Сан-Луи́с-Потоси́. Сотрудник факультета компьютерных наук

Государственного университета. Сфера научных интересов: приложения онтологий в программной инженерии, интеллектуальные системы, основанные на логическом выводе.

Francisco-Edgar Castillo BARRERA - Doctor in Information Technologies, Full-time Professor at the Autonomous University of San Luis Potosí, School of Engineering, Department of Computer Science since 2002. He is a specialist in the Department of Computer Science of State University. His research interests include Ontology applications in Software Engineering and Intelligent Systems based on Logic.