УДК 004.415.2. (043.3) https://doi.org/10.15407/pp2020.02-03.074

ДЕСКРИПТИВНАЯ МОДЕЛЬ ЭКОСИСТЕМЫ СТИЛЯ ПРОГРАММИРОВАНИЯ

Н.А. Сидоров, Н.Н. Сидорова, Е.Н. Сидоров

В процессах разработки и сопровождения программного продукта создается и используется множество вещей, которые называются артефактами программного обеспечения. Артефакты изменяются, повторно используются и меняют связи в процессах разработки и сопровождения программного продукта. Сложность и разнообразие связей артефактов программного обеспечения требуют адекватных средств описания и управления. Ими может быть экосистема артефактов программного обеспечения. В статье, впервые, предлагается концепция экосистемы артефакта программного обеспечения. В рамках концепции описана обобщенная модель экосистемы артефакта программного обеспечения, которая относится к типу Cornerstone и состоит из трех актеров: платформа, программное обеспечение и артефакт. Указаны роли актеров в экосистеме, описаны связи между актерами. Применение концепции показывается на примере экосистемы стиля программирования.

Ключевые слова: инженерия программного обеспечения, артефакт программного обеспечения, программирование, стиль программирования, экосистема программного обеспечения, онтология.

У процесах розробки і супроводу програмного продукту створюється і використовується безліч речей, які називаються артефактами програмного забезпечення. Артефакти змінюються, повторно використовуються і змінюють зв’язки в процесах розробки і супроводу програмного продукту. Складність і різноманітність зв'язку артефактів програмного забезпечення вимагають адекватних засобів опису та управління. Таким засобом може бути екосистема артефактів програмного забезпечення. У статті, вперше пропонується концепція екосистеми артефакту програмного забезпечення. В рамках концепції описана узагальнена модель екосистеми артефакту програмного забезпечення, яка відноситься до типу Cornerstone і складається з трьох акторів: платформа, програмне забезпечення та артефакт. Вказані ролі акторів в екосистемі, описані зв'язки між акторами. Застосування концепції показується на прикладі екосистеми стилю програмування.

Ключові слова: інженерія програмного забезпечення, артефакт програмного забезпечення, програмування, стиль програмування, екосистема програмного забезпечення, онтологія.

In the process of developing and maintaining a software product, many things are created and used that are called software artefacts. Software artifacts are changed, reused, and change relationships in the development and maintenance processes of a software product. The complexity and variety of software artifact relationships require adequate means of description and management. They may be a software artifact ecosystem. In the article, for the first time, a concept of a software artifact ecosystem is proposed. The concept describes a generalized model of the software artifact ecosystem, which is the Cornerstone ecosystem type and consists of three actors – the platform, the software, and the artifact. The roles of actors in the ecosystem are indicated, the relationships between actors are described. As an example, based on the generalized model of the software artifact ecosystem, a declarative model of the programming style ecosystem has been developed.

Key words: software engineering, software artifact, programming, programming style, software ecosystem, ontology.

Вступ

В процессах разработки и сопровождения программного продукта создается и используется множество вещей, которые называются артефактами. Артефакты могут быть разными по форме и представлению. Они могут входить в состав программного продукта или обеспечивать процессы его разработки и сопровождения, быть промежуточными результатами процессов или входить в состав других артефактов. Таким образом, существует огромное разнообразие артефактов программного обеспечения, включающее, проектные планы, рабочие продукты (спецификации, архитектурный и детальный проекты, код, документация), пользовательские истории, отчеты об ошибках, инструменты, в том числе для обработки артефактов, но не ограниченное этим. Между артефактами устанавливаются разнообразные и часто сложные связи. Артефакты изменяются, повторно используются и меняют связи в процессах разработки и сопровождения программного продукта. Поэтому, артефакты играют важную роль в жизненном цикле программного обеспечения независимо от его модели и требуют внимания всех заинтересованных сторон.

Стиль программирования, обеспечивая понятность исходного кода, является важным артефактом в условиях широко распространенного многократного и повторного использования программного обеспечения [1]. Применение стиля программирования требует особого внимания не только от разработчиков программного обеспечения, но и от менеджмента проекта. Это ведет к необходимости решения задач выбора или разработки стиля, его применения, анализа эффективности использования стиля и изменения [2].

Индустрия программного обеспечения постоянно развивается и меняется. Развиваются не только продукты и технологии, но многие компании разработчики программного обеспечения, экспериментируя с новыми бизнес-моделями, что иногда приводит к фундаментальным изменениям во всех отраслевых структурах как компании, так и ее клиента. В последнее время, многие компании используют для развития концепцию «экосистем программного обеспечения», создавая их вокруг себя или своих продуктов, с учетом связей клиентов. Экосистемы показали себя перспективным средством управления, эволюционирующим программным продуктом.

Сложность и разнообразие связей артефактов программного обеспечения требуют адекватных средств описания и управления. Ими может быть экосистема артефактов программного обеспечения. В такой экосистеме рассматривается более детальный уровень, чем в экосистеме программного обеспечения, однако на этом уровне может использоваться большинство походов, методов и инструментов, из тех, которые применяются в экосистеме программного обеспечения.

В статье, впервые предлагается обобщенная модель артефакта программного обеспечения, применение которой показывается на примере экосистемы стиля программирования. В рамках концепции описана обобщенная модель экосистемы артефакта программного обеспечения, которая относится к типу Cornstoun ecosystem и состоит из трех актеров – платформа, программное обеспечение и артефакт. Указаны роли актеров в экосистеме, описаны связи между актерами. Типы, правила и атрибуты актеров, связей и действий могут уточняться для моделей конкретных экосистем атрибута программного обеспечения. Это же относится к удовлетворению требований для анализа свойств экосистемы.

На основе обобщенной модели экосистемы артефакта программного обеспечения разработана декларативная модель экосистемы стиля программирования, который является артефактом, играющим важную роль в разработке и сопровождении программного обеспечения. Описание процессов создания и использования стиля программирования выполнено путем применения онтологии.

Обзор литературы

Артефакты программного обеспечения. В жизненном цикле программного обеспечения для обеспечения процессов создания и сопровождения программного продукта создается и используется множество разных вспомогательных вещей – артефактов. Артефакты создаются в доменном анализе или в других процессах жизненного цикла. В качестве артефактов рассматривается широкий спектр составляющих, от документации, рабочих продуктов и их частей, до вспомогательных инструментов. Взаимодействуя, артефакты обеспечивают эффективное выполнение процессов жизненного цикла.

В работе [3] артефакты анализируются, в контексте повторного использования как оборудование в смысле работы [4]. При этом, рассматриваются три цели (написание, обработка и передача артефактов) и три аспекта оборудования (the in-order-to of equipment, readiness-to-hand, presence-and-hand). Кроме этого, так как артефакты анализируются как повторно используемые компоненты, которые встраиваются в создаваемый программный продукт, то принимаются во внимание их характеристики: холизм, общность, повторная используемость и зрелость. Рассматривая артефакт, как оборудование – вещь, встроенное в контекст программного продукта, исследуется взаимовлияние указанных характеристик артефактов программного.

В работе [5] артефакты рассматриваются в контексте software product line и делятся на три типа – архитектура, разделяемые компоненты и компоненты, произведенные из разделяемых. Для каждого типа артефактов идентифицируется три уровня зрелости в зависимости от степени интеграции артефакта соответствующего типа в software product line.

В работе [6] артефакты рассматриваются как информационные части, которые создаются, модифицируются и используются в процессах технологии RUP. Артефакты могут быть разных видов и принимать разные формы, от UML моделей до исполняемого кода, и могут использоваться в создании и сопровождении программного продукта. Артефакты являются входом и результатом действий в процессах RUP. При этом, основной задачей поддерживающего рабочего процесса Configuration & Change Management является управление артефактами, созданными членами команды, разрабатывающей проект.

В работе [7] рассматриваются теоретические основы представления и интерпретации артефактов программного обеспечения. Исходя из различных уровней восприятия артефактов человеком - пользователем артефактов вводится три уровня представления артефактов – физический (физическое представление), структурный (синтаксическая структура) и семантический (семантический контент). Кроме этого вводится два шага обработки артефактов – синтаксический анализ физического представления (parsing), анализ синтаксической структуры – результата первого шага (interpretation). На основе уровней представления и шагов обработки строится мета модель артефактов.

В работе [8] в качестве артефакта рассматривается документация программного обеспечения, которая определяется как средство представляющее информацию о программном обеспечении. Вводится модель сопровождения документации как артефакта программного обеспечения.

В работе [9] рассматривается архитектура средств, обеспечивающих создание и сопровождение метаданных об артефактах программного обеспечения, которые образуют среду, состоящую из ресурсов – артефактов разработки. Средства используются для управления средой артефактов.

К экосистеме артефактов программного обеспечения. Нам неизвестны работы, прямо посвященные рассмотрению задач, связанных с исследованием экосистем артефактов программного обеспечения. Однако, есть работы, результаты которых могут применяться для решения указанных задач. В работе [10], справедливо обращается внимание на то, что в экосистемах программного обеспечения сейчас внимание уделяется участникам только верхнего уровня – это, организации и команды, которые создают, внедряют и сопровождают программное продукты. Однако, имеет место нижний уровень – артефакты, роль которых в процессах жизненного цикла трудно переоценить. В работе [11] приводится множество требований для описания и

анализа экосистем программного обеспечения, которые используются в нашей статье для моделирования экосистем артефактов программного обеспечения.

Обобщенная модель экосистемы артефакта программного обеспечения

В этом разделе рассматривается обобщенная (generic) модель экосистемы артефакта программного обеспечения. Для моделирования экосистем программного обеспечения сейчас используется несколько методов [12]. Применение конкретного метода зависит от типа экосистемы и целей моделирования. Для представления экосистемы артефакта программного обеспечения используется i* метод [13]. В отличие, от наиболее часто используемого SSN метода, который ориентирован на описание экосистемы программного обеспечения на верхнем уровне (продукт, разработчик, поставщик, пользователь), i* метод обеспечивает описание экосистемы более детального уровня представления программного обеспечения, который соответствует уровню артефактов программного обеспечения. На рис. 1 методом i* показана обобщенная модель экосистемы артефакта программного обеспечения.

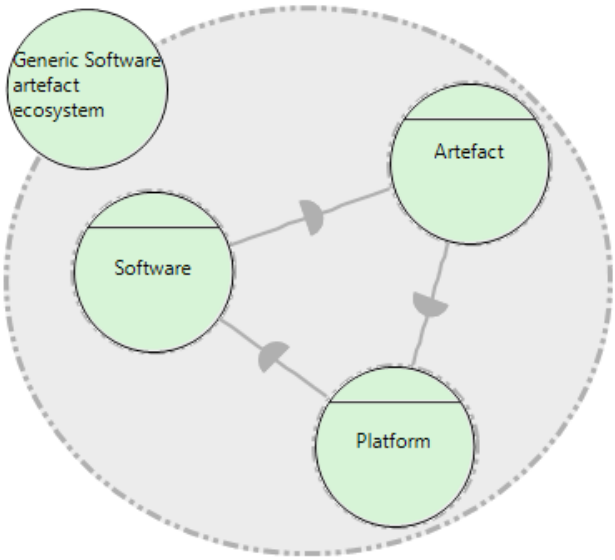

Рис. 1 Обобщенная модель экосистемы артефакта программного обеспечения

При проектировании экосистемы удовлетворяется две группы требований [11]: описательные и аналитические.

К первой группе, относятся требования определения актеров, связей между ними и их действий. Кроме этого формулируются требования определения типов, правил и атрибутов актеров, связей и действий, а также требования определения специфических характеристик, как экосистемы в целом, так и ее элементов, например, продуктивность, эффективность, защищенность.

Ко второй группе, относятся требования для определения характеристик, обеспечивающих анализ экосистемы от стимулов и мотивации до устойчивости и производительности.

В табл. приведены актеры и роли в экосистеме артефакта программного обеспечения. Рассматриваемая экосистема относится к типу Cornerstone, так как основу экосистемы составляет технологическая платформа для разработки и сопровождения программного обеспечения, функциональность которой расширяется путем использования артефактов [14]. Таким образом, актерами экосистемы являются платформа с ролью управления, программное обеспечение с ролью программного продукта, артефакт с ролью поставщика сервиса поддержки.

Таблица. Актеры и роли в экосистеме

Тип экосистемы	**Актер**	**Роль актера в экосистеме**
Cornstoun ecosystem	Платформа (Platform)	Управление (Orchestration)
	Программное обеспечение (Software)	Продукт (Product)
	Артефакт (Artefact)	Поставщик сервиса поддержки (Support service provider)

Между актерами можно указать общие связи. Платформа, в контексте которой рассматриваются такие составляющие как модель жизненного цикла, организационное и техническое обеспечение разработки и сопровождения, определяет и использует артефакт, как вспомогательное средство реализации процессов и заполнения структуры программного продукта. Программное обеспечение зависит от платформы, которая является основным средством реализации процессов разработки и сопровождения, и прямо, как компонент в структуре программного обеспечения или косвенно, как средство повышения эффективности процессов платформы использует артефакт.

Типы, правила и атрибуты актеров, связей и действий могут уточняться для моделей конкретных экосистем атрибута программного обеспечения. Это же относится к удовлетворению требований для анализа свойств экосистемы [11].

Экосистема стиля программирования

На сегодняшний день для создания и сопровождения продуктов программного обеспечения получили распространение методы и средства, которые основаны на многократном и повторном использовании. Применение этих методов и средств требует от разработчика чтения, анализа и понимания значительного количества представлений рабочих продуктов различных фаз жизненного цикла. Многократное и повторное использование сейчас распространено от спецификаций требований до исходных текстов и документации. Поэтому, к программному обеспечению, одним из главных, выдвигается требование понятности. Деятельность разработчика будет более эффективной, программное обеспечение понятным, а разработка и сопровождение дешевле, когда при создании программного обеспечения применяются стили (стандарты), обеспечивающих понятность рабочих продуктов разных фаз жизненного цикла [1].

На рис. 2 показана модель экосистемы стиля программирования, которая построена на основе обобщенной модели артефакта программного обеспечения (рис. 1).

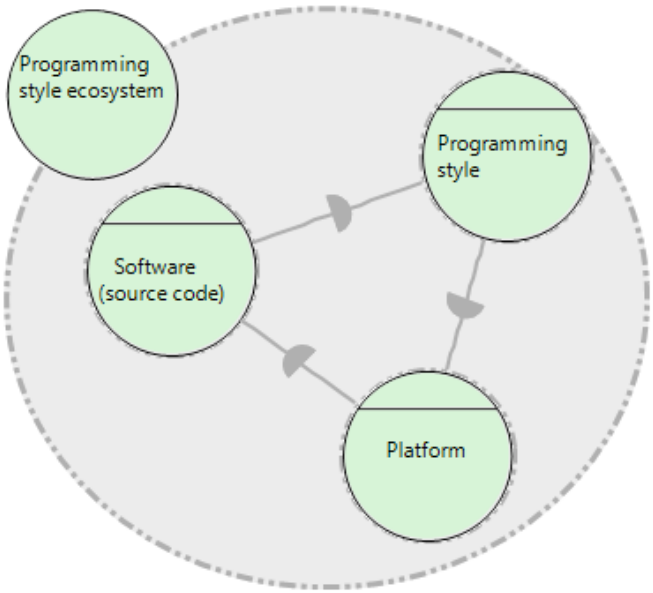

Рис. 2. Модель экосистемы стиля программирования

Артефактом в этой модели является стиль программирования, а актер – программное обеспечение, - представляется той своей частью – исходным кодом, для которой применяется стиль программирования. Артефакт – стиль программирования определяется платформой, так как правила стиля зависят от ряда платформенных условий, например, языка программирования, целей менеджмента, сроков, рисков и бюджета проекта. В свою очередь, стиль программирования используется при создании исходного кода и влияет на эффективность процессов конструирования и сопровождения. Применение стиля программирования предусматривает реализацию двух процессов [2]: создание, в результате выполнения которого строится стиль языка программирования (programming language style) и использование стиля при написании программ.

На рис. 3. показана онтология создания стиля программирования, которая детальнее описывает участников и действия, происходящие в связи с этим в экосистеме стиля программирования. Все концепты онтологии относятся к категории ресурсов в терминологии i*, за исключением концепта <<event>>, который представляет цель. При этом, концепты Coding phase, Party, Programming language относятся к актеру Platform, а концепты Creating work product style, Style party create guide, Style и Programming language style к актеру Programming style.

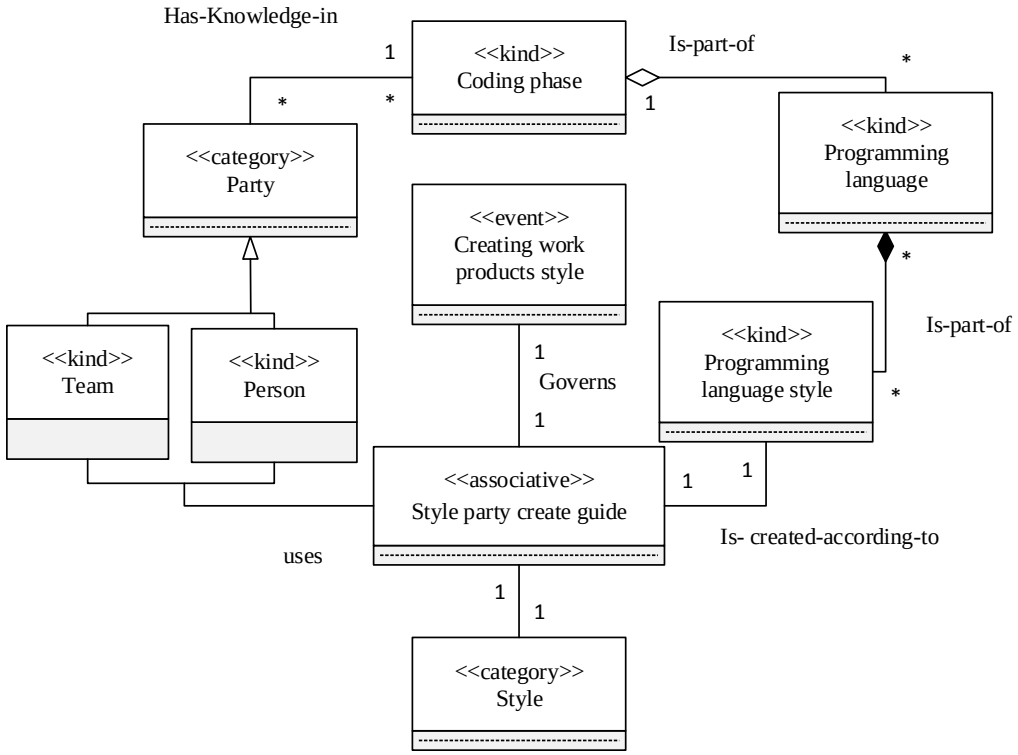

Рис. 3. Онтология создания стиля программирования

На рис. 4. показана онтология использования стиля программирования, которая также описывает соответствующих участников и действия в экосистеме стиля программирования. Концепты Party, Coding phase, принадлежат актеру Platform, концепты Program, Program style – актеру Software, а концепты Using work product style, Style party using guide, Program language style – актеру Programming style.

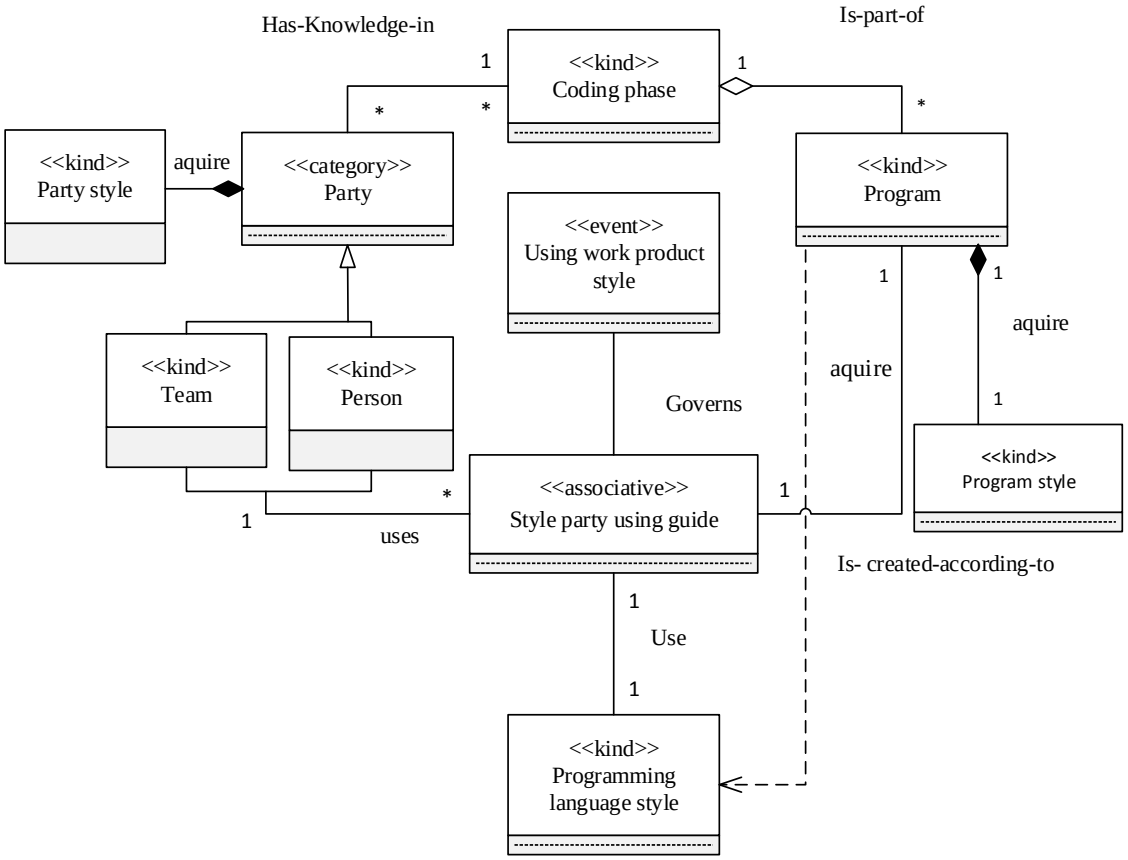

Рис. 4. Онтология использования стиля программирования

Для реализации процессов создания и использования стиля программирования создаются инструменты, которые могут рассматриваться, с одной стороны, как ресурсы артефакта Programming style, а с другой стороны, как артефакты в составе артефакта Platform (рис. 5). К ним относятся база знаний стиля программирования и ризонер. Таким образом, программист, кодируя программу, применяет онтологию стиля программирования, как для изучения стиля, так и для проверки соблюдения стиля в программе. Поэтому нужны два средства – одно, для создания онтологии и поддержки программиста в процессе кодирования, а второе, для контроля за применением стиля программирования в исходном тексте программы [2].

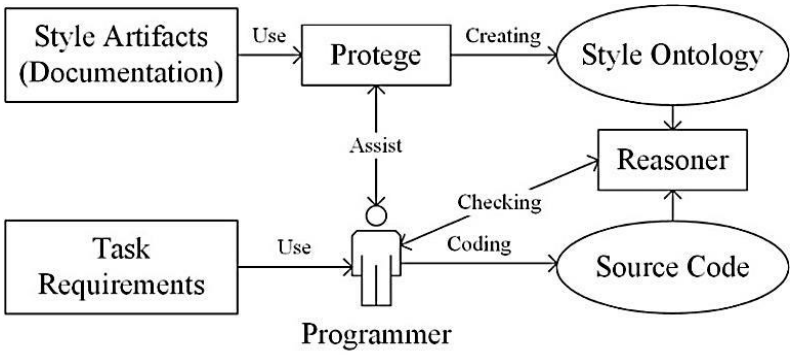

Рис. 5. Схема использования инструментов

Аналитик стиля, применяя Protégé, настраивает онтологию на соответствующий стиль программирования, создавая TBox (рис. 6). После настройки, программист, используя онтологию, с помощью Protégé знакомится со стилем программирования. Второе средство, функционально похож на ризонер, но добавляется функция идентификации ошибок соблюдения стиля (Style Errors). В терминах дескриптивной логики ризонер верифицирует совместимость онтологии.

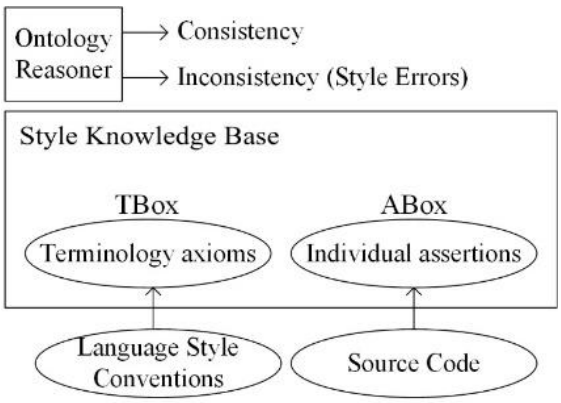

Рис. 6. База знаний стиля программирования

Protégé используется для создания TBox, части онтологии, содержит термины, описывающие стиль программирования. Утверждения относительно исходного кода (ABox), написанного программистом, создаются соответствующей частью ризонером, поскольку она обеспечивает соответствующий сервис, применяя базу знаний (TBox и ABox). Сервис включает, во-первых, верификацию совместимости онтологии (прямая функция ризонера), а во-вторых, поиск стилистических ошибок в исходном тексте программы.

Выводы

1. Впервые, предложена концепция экосистемы артефакта программного обеспечения. В рамках концепции описана обобщенная модель экосистемы артефакта программного обеспечения, которая относится к типу Cornstoun ecosystem и состоит из трех актеров –платформа, программное обеспечение и артефакт. Указаны роли актеров в экосистеме, описаны связи между актерами. Типы, правила и атрибуты актеров, связей и действий могут уточняться для моделей конкретных экосистем атрибута программного обеспечения. Это же относится к удовлетворению требований для анализа свойств экосистемы.

2. На основе обобщенной модели экосистемы артефакта программного обеспечения разработана декларативная модель экосистемы стиля программирования, который является артефактом, играющим важную роль в разработке и сопровождении программного обеспечения. Описание процессов создания и использования стиля программирования выполнено путем применения онтологии.

В продолжение описания экосистемы будет расширено описание актеров экосистемы стиля программирования и разработаны типы, правила и атрибуты актеров, связей и действий. Кроме этого, предполагается рассмотреть метрическое обеспечение экосистемы относительно определения эффективности, устойчивости и надежности экосистемы стиля программирования.

Литература

1. Сидоров Н.А. Стилистика программного обеспечения. *Проблеми програмування*. 2018. 2, 3. С. 245–254
2. Sydorov N., Sydorova N., Sydorov E., Cholyshkina O., Batsurovska I. Development of the approach to using a style in software engineering, Eastern-European Journal of Enterprise Technologies. 2019. 4/2 (100) P. 41–51.
3. Nuwangi S.M. & Darshana S. (2015) Software artefacts as equipment: a new conception to software development using reusable software artefacts. Thirty-Sixth International Conference on Information Systems, 2015, Texas, USA.
4. Heidegger M. 1927/1962. Being and Time, Translated by John Macquarrie & Edward Robinson. USA: Harper & Row.
5. Bosch J. (2002) Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization, Software Product Lines, Second International Conference, SPLC 2, San Diego, CA, USA, August 19–22, 2002.
6. Rational Unified Process: Best Practices for Software development Teams, Rational Software White Paper TP026B, Rev 11/01, 1998, 18 p.
7. Fernandez, D M., Bohm W., Broy M., (2018) Artefacts in Software Engineering: A Fundamental Positioning, *International Journal on Software and Systems Modeling*. 2018. 26. 9 p.
8. Glass R. Software maintenance documentation, SIGDOC '89, Pittsburg, Pennsylvania, USA, ACM Press. P. 18–23.
9. Dewar R.G. Managing Software Engineering Artefact Metadata, Department of Computer Science, Heriot-Watt University, Edinburgh, UK.
10. Seichter D., Dhungana D., Pleuss A., Hauptmann B. Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens, ECSA 2010 August 23–26, 2010, Copenhagen, Denmark. P. 119–126.
11. Sadi M., Yu E. (2015) Designing Software Ecosystems: How Can Modeling Techniques Help?, Springer-Verlag, Berlin Heidelberg 2015. 15 p.
12. Сидоров Н.А. Экология программного обеспечения. Інженерія програмного забезпечення, 2010. С. 53–61.
13. Yu E.: Modelling Strategic Relationships for Business Process Reengineering. Ph.D., thesis. Dept. of Computer Science, University of Toronto. 1995.
14. Knodel J., and Manikas K. Towards a typification of software ecosystems. In Fernandes et al. Software Business – 6th International Conference, ICSOB 2015, Braga, Portugal, June 10–12, 2015, Proceedings (2015), Vol. 210 of Lecture Notes in Business Information Processing, Springer. P. 60–65.

References

1. Sydorov N.A. (2005) Software Stylistics, Problems of Programming. 2018. 2, 3. P. 245–254.
2. Sydorov N., Sydorova N., Sydorov E., Cholyshkina O., Batsurovska I. (2019) Development of the approach to using a style in software engineering, Eastern-European Journal of Enterprise Technologies. 2019. 4/2 (100). P. 41–51.
3. Nuwangi S.M. & Darshana S. (2015) Software artefacts as equipment: a new conception to software development using reusable software artefacts. Thirty-Sixth International Conference on Information Systems, 2015, Texas, USA.
4. Heidegger, M. (1927/1962) Being and Time, Translated by John Macquarrie & Edward Robinson. USA: Harper & Row.
5. Bosch J.,(2002) Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization, Software Product Lines, Second International Conference, SPLC 2, San Diego, CA, USA, August 19-22, 2002.
6. Rational Unified Process: Best Practices for Software development Teams, Rational Software White Paper TP026B, Rev 11/01. 1998. 18 p.
7. Fernandez, D M., Bohm W., Broy M., (2018) Artefacts in Software Engineering: A Fundamental Positioning. International Journal on Software and Systems Modeling. 2018. 26. 9 p.
8. Glass, R. (1989) Software maintenance documentation, SIGDOC '89, Pittsburg, Pennsylvania, USA, ACM Press. P. 18–23.
9. Dewar, R.G., (2005) Managing Software Engineering Artefact Metadata, Department of Computer Science, Heriot-Watt University, Edinburgh, UK.
10. Seichter D., Dhungana D., Pleuss A., Hauptmann B. (2010) Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens, ECSA 2010 August 23–26, 2010, Copenhagen, Denmark. P. 119–126.
11. Sadi M., Yu E. (2015) Designing Software Ecosystems: How Can Modeling Techniques Help?, Springer-Verlag, Berlin Heidelberg 2015. 15 p.
12. Sydorov N. (2010) Software Ecology, Software Engineering. 2010. P. 53–61.
13. Yu E.: (1995) Modelling Strategic Relationships for Business Process Reengineering. Ph.D., thesis. Dept. of Computer Science, University of Toronto. (1995)
14. Knodel J., and Manikas K. (2015) Towards a typification of software ecosystems. In Fernandes et al. Software Business – 6th International Conference, ICSOB 2015, Braga, Portugal, June 10–12, 2015, Proceedings (2015), Vol. 210 of Lecture Notes in Business Information Processing, Springer. P. 60–65.

Получено 17.03.2020

Об авторах:

Сидоров Николай Александрович,
доктор технических наук, профессор.
Количество научных публикаций в общегосударственных базах данных – 18.
Количество научных публикаций в международных базах данных – 2.
https://orcid.org/0000-0002-3794-780X,

Сидорова Ника Николаевна,
кандидат технических наук, доцент.
Количество научных публикаций в общегосударственных базах данных – 15.
Количество научных публикаций в международных базах данных – 7.
https://orcid.org/0000-0002-2989-3637,

Сидоров Евгений Николаевич,
кандидат технических наук, доцент,
старший инженер.
Количество научных публикаций в общегосударственных базах данных – 13.
Количество научных публикаций в международных базах данных – 5.
https://orcid.org/0000-00022609-0230.

Место работы авторов:

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute".
E-mail: nyksydorov@gmail.com.
Тел.: 0677980361,

Кафедра компьютерных и информационных технологий
Межрегиональная академия управления персоналом
03039, г. Киев, Украина, ул. Фрометовская, 2.
E-mail: nika.sidorova@gmail.com.
Тел.: 0681006173,

P&S Integrated Media Enterprise
Avid Development GmbH
Paul-Heyse-Straße, 29, München, Bavaria, 80336.
E-mail: Eugen.sidorov@live.com