УДК 006

ПРЕДЛОЖЕНИЯ К СТАНДАРТУ, ОПИСЫВАЮЩЕМУ АРХИТЕКТУРУ И ФОРМАТ ДАННЫХ УМНЫХ СТАНДАРТОВ

Проказин М.Р., Теремов И.А.

«МИРЭА - Российский технологический университет», 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: prokazin@mirea.ru, teremov@mirea.ru

В настоящий момент ведутся работы над серией ПНСТ (Предварительных Национальных Стандартов), подготавливающих нормативное и формализованное поле для внедрения SMART-стандартов. В меру интенсивного развития данной области отечественной стандартизации возникает ряд предложений, связанных с подходами, на которые опираются разработчики данной серии ПНСТ. Статья вносит некоторые предложения, основанные на лучших практиках современной программной инженерии, для повышения общего уровня рассматриваемой области. В процессе разработки необходимо внимательно отнестись к таким аспектам, как разработка системы типизации над пространством объектов стандартизации, а также организация системы связей между подобными объектами. Помимо этого, немаловажно также остановиться на разделении уровней абстракции и ответственности между каждым из уровней модельной структуры подобных документов. Положения статьи основаны на результатах рассмотрения редакции ПНСТ «Умные (SMART) стандарты. Архитектура и форматы данных» от марта 2024 года и исходят из известного опыта внедрения и исследования требование-ориентированных форматов, а именно их ориентированность на степень выразительности самих требований. На момент публикации данный ПНСТ доработан по результатам публичного обсуждения.

Ключевые слова: SMART-стандарты, цифровая трансформация, нормативная база, программная инженерия, требования

PROPOSALS FOR A STANDARD DESCRIBING THE ARCHITECTURE AND DATA FORMAT OF SMART STANDARDS

Prokazin M.R., Teremov I.A.

«MIREA - Russian Technological University», 119454, Moscow, 78 Vernadsky Avenue, Russia, e-mail: prokazin@mirea.ru, teremov@mirea.ru

Nowadays work is underway on a series of preprinted national standards that should prepare the normative and formalized field for the implementation of SMART standards. With the intensive development of this area of domestic standardization, a number of proposals arise related to the approaches on which the authors of this series of preprinted national standards rely. The article makes some suggestions, based on the best practices of modern software engineering, to improve the overall level of the field under consideration. During the development process, it is necessary to pay special attention to such aspects as the development of a typing system over the space of standardization objects, as well as the organization of a system of connections between similar objects. In addition, it is also important to dwell on the division of levels of abstraction and responsibility between each of the levels of the model structure of such documents. In addition, the article is based on the results of the review of the revision of the "Smart (SMART) standards. Architecture and data formats" from January 2024 and relies on the well-known experience of implementing and researching requirement-oriented formats, namely their focus on the degree of expressiveness of the requirements themselves. At the time of publication, this preprinted national standard has been revised based on the results of public discussion.

Keywords: SMART standards, digital transformation, regulatory framework, software engineering, requirements

Введение

Прогресс в области цифровой трансформации экономики привёл к обострению вопроса эффективности стандартизации в трансформирующейся промышленности России и других стран. Ответом на запрос увеличения эффективности стандартизации и стандартов должны стать “умные”, или SMART-стандарты. Достаточно хорошо известная на рынке справочных информационных систем Консорциум Кодекс, являющийся ведущей организацией ПТК 711 «Умные стандарты», и активно работающей в этой области, занимается разработкой

базовой инфраструктурой и соглашений о том, как должны быть устроено нормативное поле стандартизации в результате процесса цифровой трансформации.

В настоящий момент принят ПНСТ 864-2023 “Умные (SMART) стандарты. Общие положения” [1], описывающий основополагающие определения и принципы в рассматриваемом поле “умной” стандартизации. Вместе с этим, разработан также проект ПНСТ «Умные (SMART) стандарты. Классификация объектов стандартизации. Общие положения», предлагающий принципы универсальной классификации сущностей в нормативных документах, которая может быть применима в том числе при разработке SMART-стандартов [2].

Немаловажным этапом над разработкой концепта SMART-стандартов является продолжающаяся на сегодняшний день работа над проектом стандарта, который будет описывать архитектуру и формат данных умных стандартов, то есть техническую спецификацию. Данный проект ПНСТ “Умные (SMART) стандарты. Архитектура и форматы данных” на сегодняшний день завершил активную фазу публичного обсуждения [3].

В ходе работы над данной статьей авторам удалось ознакомиться с предварительным вариантом данного стандарта, а также сформулировать замечания и обозначить некоторые проблемные положения, которые утверждались в данном документе. По мнению авторов данной статьи, для повышения эффективности разработчику стандартов при дальнейшей работе над данной серией стандартов, либо схожих по смыслу нормативов и форматов, предлагается учесть взгляд на описанные положения с точки зрения программной инженерии.

Очевидно, что важность качественного формального описания концепции SMART-стандартизации в виде подобной серии “мета-стандартов” нельзя переоценить, так как сейчас на основе рассматриваемой серии ПНСТ будет заложен технический фундамент для всех области стандартизации на многие годы вперёд, и даже самое небольшое техническое или эргономическое несовершенство на этапе описания форматов, архитектуры и правил функционирования приведет к колоссальным затратам на исправления, до- и переработки в будущем [4].

Цель данной статьи – сформулировать ряд свойств, которыми может обладать спецификация формата данных “умных” стандартов, и которые по меньшей мере стоит учитывать при разработке такого формате, ориентируясь на возможные варианты дальнейшего использования и развития экосистемы SMART-стандартов.

Стоит отметить, что авторы данной статьи являются квалифицированными специалистами в области программной инженерии и обладают достаточным для вынесения суждений по теме практическим опытом. Данное обстоятельство позволяет проводить экспертную оценку положений, представленных в документе, особенно в разрезе использования описанного в документе контракта SMART-стандарт в первую очередь при производстве SMART-сервисов, которые являются предметом программной инженерии.

Краткий обзор рассматриваемого стандарта

Предложенный ПНСТ 864-2023 “Умные (SMART) стандарты. Общие положения” описывает архитектуру данных и форматы, которые используются внутри этих стандартов. Он предоставляет руководящие принципы для разработки и внедрения SMART-стандартов, сосредоточенных на структурировании данных в машиночитаемом и машинопонимаемом виде. Стандарт устанавливает единую объектную модель данных, что облегчает взаимодействие и обмен данными между системами, использующими SMART-стандарты. Также стандартизируется формат для хранения, передачи и обновления этих стандартов.

Архитектура данных включает логические структуры, которые определяют, как различные типы данных внутри SMART-стандарта связаны и управляются. Информационные объекты используются для представления различных элементов стандарта, обеспечивая согласованность и точность при работе как с структурированными, так и с неструктурированными данными. Стандарт определяет методы классификации, идентификации и актуализации этих объектов данных, предлагая надежный механизм для поддержания целостности данных и контроля версий на протяжении всего жизненного цикла стандарта. Использование XML для транспортных форматов и стандартизированных метаданных направлено на упрощение процесса интеграции SMART-стандартов в промышленные и информационные системы.

Консолидация и разделение уровней абстракции

Естественно, что стандарт, описывающий архитектуру и формат данных, которые должны управлять содержимым “умных” стандартов, рано или поздно столкнется с необходимостью отвечать на вопросы, касающиеся постулатов некого “метастандарта”, определяющего правила и нормы организации SMART-стандартов.

Можно выделить две основных группы пользователей (в т.ч. разработчиков ПО) SMART-стандарта, которым в рамках своей работы необходимо взаимодействовать непосредственно с содержимым такого стандарта:

– разработчики прикладного ПО, в котором требуется интерпретировать данные стандарта для

выполнения своей прикладной функции (например, различные САПР либо прочие интеллектуальные киберфизические системы). Таким разработчикам необходимо понимать семантику данных умного стандарта;

– разработчики ПО для разработки и сопровождения самих “умных” стандартов. Им необходимо углубленное понимание принципов сериализации документа.

Исходя из этого, рациональным является выделение как минимум двух уровней абстракции при описании структуры “умного” стандарта: логического и физического. Описание логической структуры должно дать пользователям понимание сущности и семантической структуры SMART-стандарта (из чего он состоит, какие выразительные средства имеет). Описание физической структуры, в свою очередь, должно объяснить способ считывания или записи данных стандарта на физический носитель. Подобная декомпозиция во многом упростят эксплуатацию всего стандарта, разделив ответственность за различную функциональность между отдельными уровнями абстракции [5].

Проанализировав опыт применения ReqIF [6], мы находим важным выделение еще одного уровня абстракции. В связи с тем, что, как и ReqIF, умные стандарты в большинстве проектов скорее всего будут являться контейнерами, содержащими не только текст на формальном языке и описывающий требования стандарта, но и множество других данных (например, форматированный человекопонимаемый текст, файл исходного кода предикатов, описывающих специфичные для стандартизируемой сущности требования).

Мы считаем, что описание способа существования всех данных должно быть описано отдельно, по двум причинам:

– Обработка никаких данных, кроме самого формального текста умного стандарта в любом случае невозможна в программных реализациях стандарта;

– Удобные и эффективные способы обработки и транспортировки контейнеров данных меняются от среды к среде и со временем.

Ориентированность на требования и выразительность

SMART-стандарт должен быть хорошим выразительным средством для описания требований к объектам стандартизации, которые сами по себе представляют явления реального мира [7]. Если в формате формального описания “умного” стандарта не будет возможности описывать все множество возможных требований, то разработчики стандартов будут переносить эту ответственность из формального представления в прилагающиеся к нему документы с исключительно человекопонимаемым содержимым на естественном языке. Такое решение, в сущности, превращает всю структуру умного стандарта в сложную и хорошо описанную метаинформацию к архиву с файлами.

Только в случае, если требования выражены на формальном языке, указанном в спецификации “умного” стандарта, будет действительно возможна реализация ПО, увеличивающего эффективность стандартизации и разработки с применением стандартов, таких как автоматизированный поиск противоречий в требованиях и продвинутая автоматизированная классификация.

Для того, чтобы SMART-стандарты действительно оказались достаточно выразительными, необходимо учесть практику из языков программирования, так как они, в сущности, тоже являются формальными языками, которые должны выразительно описывать явления реального мира [8].

Выразительность системы типов

В языках программирования не всегда были системы типов. Ранние низкоуровневые языки не предлагали никаких абстракций над данными, кроме связанных с аппаратной организацией вычислителя, такие как байт, машинное слово. Позже появились более сложные, композирующие типы, такие как структуры. Затем, была введена абстракция вариативности сущности, которую описывает тип, внутри некоторой категории. Отсюда появились такие принципы, как полиморфизм и наследование, которые широко применяются повсеместно в программной инженерии при объектно-ориентированном моделировании. Самые современные языки предлагают алгебраические типы, то есть их системы типов позволяют выражать типы через операции над типами (объединение, пересечение, включение, и т.д.) [9].

Важно отметить, что появление и всеобщее распространение таких абстракций произошло не случайно [10]. В первую очередь, подход к абстрагированию обрабатываемых данных через типы, обладает некоторой эргономикой для человека, который оперирует сущностями программной инженерии. Абстракции позволяют инкапсулировать многие несущественные детали сущностей, упрощая операции над комплексами подобных сущностей.

Многие форматы и системы требование-ориентированных документов не представляют выраженного способа управлять типами данных. Зачастую, подобные форматы допускают применение лишь скалярных

примитивных типов, таких как целые и вещественные числа, строки, точки во времени [11ref <3]. Однако, применение сложных составных типов, таких, как типизированные кортежи, либо вложенные кортежи, не распространено среди рассматриваемых форматов.

Как в случае языков программирования, так и в случае требование-ориентированных документов, их автором (разработчиком) является человек. Из этого следует, что выработанные требования к эргономике языков программирования следует применять и к формальному языку умных стандартов.

Исходя из этого, можно сделать вывод, что для того, чтобы эргономично структурировать данные, над которыми в стандарте будут выражаться требования, следует использовать системы, похожие на системы типов в языках программирования.

Умные стандарты должны иметь возможность строго определять структуры данных, которыми они оперируют для моделирования и представления требований к объектам стандартизации. Определения структур данных должны поддерживать самые продвинутые подходы из систем типов языков программирования: дизъюнкцию, конъюнкцию и включение над структурами данных. За счёт этого умный стандарт сможет определить состав стандартизируемых атрибутов и их структуру, при этом обеспечив эффективность машинной интерпретации и эргономичность восприятия структуры требования человеком.

Выразительность системы ссылочных связей

Одной из немаловажных характеристик любых реляционных систем сущностей является принцип установления связей между сущностями. Характер реляционной структуры должен обеспечивать точное и однозначное описание отношений между описываемыми сущностями. Однако, в предложенном формате рассматриваемого документа присутствуют некоторые принципы, которые вносят явную неоднозначность в реляционную модель сущностей.

В первую очередь, необходимо остановиться на предлагаемой системе адресации. В документе сказано, что каждый информационный элемент (сущность) должны быть идентифицированы собственным глобально уникальным идентификатором. Уточняется также, что он должен иметь формат UUID - глобально уникальных идентификаторов (Universally Unique IDentifiers - англ.) [12].

Однако, согласно документу, адресация информационных элементов является совокупностью из идентификаторов самого SMART-стандарта, его редакции, информационного блока и самого элемента. При этом, есть дополнительное определение, позволяющее сокращать кортеж идентификаторов, описанный выше, в случаях, когда обращение выполняется из сущностей, находящихся в одном и том же элементе более низкого уровня. Возникает логичный вопрос - для чего требуется использование глобально уникальных идентификаторов, если требуется указание подобной цепочки адресов? В данном случае можно использовать локальные адреса, уникальные внутри некоторого уровня вложенности [11]. Помимо этого, также далее по документу, нигде не предусмотрено примера использования такой цепочки идентификаторов, а также они не предусмотрены в атрибутной детализации сущностей, которыми оперирует предлагаемый стандарт.

Можно предположить, что использование такого принципа призвано решить проблему с разрешением зависимостей. Например, в случае, если есть некоторый стандарт $\boldsymbol{\alpha}$ ревизии $\boldsymbol{\alpha_1}$, в элементе X которой есть обращение к элементу **Y** из другого стандарта $\boldsymbol{\beta}$ ревизии $\boldsymbol{\beta_0}$, в элементе X необходимо указать обращение вида $\boldsymbol{(\beta, \beta_0, Y)}$. Владея полным множеством всех идентификаторов всех стандартов **S**, можно вычислить, к какому стандарту относится данный элемент **X**. Однако, не владея множеством идентификаторов ревизий стандарта $\boldsymbol{\beta}$ и множеством идентификаторов внутри конкретной ревизии $\boldsymbol{\beta_0}$ дальнейший поиск конечного элемента **Y** невозможен.

Из приведенного примера видно, что есть попытка разбиения единого множества уникальных идентификаторов на утилитарные подмножества. Так или иначе, для вычисления графа элементов предполагаемому SMART-сервису потребуется доступ ко всем подмножествам. Разработчики стандарта не приводят никаких доводов в пользу целесообразности такого подхода. С точки зрения программной инженерии, логичнее отказаться от подобного использования идентификации, и указывать только адрес конечного элемента. Этого достаточно для вычисления целостного графа связей за счет характеристик глобально уникальных идентификаторов, а современные структуры данных позволяют выполнять поиск за константное время по пространству объявленных идентификаторов такого рода [13].

Неизменяемость и соглашение о наименовании атрибутов сущностей

Рассматриваемый в данной статье предварительный стандарт в некоторых положениях регламентирует изменения информационных элементов SMART-стандарта. Например, в документе явно указано, что при изменении содержимого некоторой актуальной ревизии $\boldsymbol{\alpha_n}$ SMART-стандарта α, обязательно сформировать

новую ревизию $\boldsymbol{\alpha_{n+1}}$. Скорее всего, стандарт нужно дополнить описанием понятия неизменяемости оперируемых сущностей – существования в единственном неизменном варианте – и явно декларировать такое требование к информационным элементам любого типа. Преимущества данного подхода достаточно очевидны: любой информационный элемент **X** является конечным и неизменным, его истинность подтверждается самим его существованием, а правила, по которым он может изменяться либо утрачивать силу – определяются сформированным позднее информационным элементом **Y**, который может обладать некими обратными ссылками на элемент **X**, описывающими его актуальный статус на момент создания элемента **Y**.

Данные принципы фигурируют в теле документа в том либо ином виде, однако, не являются основополагающими и упоминаются лишь вскользь. Более того, отталкиваясь от принципа неизменяемости, необходимо пересмотреть характер некоторых атрибутов, связанных с поддержкой исторических изменений внутри SMART-стандарта.

В первую очередь необходимо обратить внимание на характер некоторых атрибутов, связанных с поддержкой историчности внутри стандарта. Например, атрибут “Номер ревизии”, который содержит порядковый номер ревизии для некоторого SMART-стандарта, который должен последовательно увеличиваться на единицу при выпуске каждой новой версии. Помимо этого, каждая ревизия снабжена также атрибутом “Момент создания” - моментом времени, когда ревизия была создана.

Очевидно, что атрибут “Номер ревизии” некоторой ревизии $\boldsymbol{\alpha_x}$ может быть легко вычислен при использовании информации о моменте создания, учитывая, что каждая ревизия имеет явную связь с идентификатором самого SMART-стандарта $\boldsymbol{\alpha}$. Для этого достаточно отсортировать связанные с $\boldsymbol{\alpha}$ ревизии по атрибуту “Момент создания” **t**, что позволит вычислить порядковый номер n любой ревизии $\boldsymbol{a_n}$.

Заключение

Предложенные меры по работе управлению абстрагированием, ориентированности на требования, и поддержке неизменяемых сущностей, позволят увеличить эффективность внедрения умных стандартов, за счет снижения как когнитивной сложности для восприятия конечными пользователями-разработчиками прикладного ПО (за счет ограничения области данных, которых им необходимо будет воспринимать и понятной системы типов), так и пользователями-разработчиками SMART-сервисов (также, за счет определения необходимой к восприятию области данных, и кроме этого, с помощью целостной, согласованной и эргономичной системы связей).

Учет перечисленных в статье аспектов важен именно на этапе разработки основополагающих принципов, которые лягут в основу экосистемы SMART-стандартов. В противном случае развитие направление SMART-стандартизации может быть замедлено избыточной сложностью и затрудненным расширением формата, или необходимостью поддержки несовместимых моделей различных предметных областей.

Список литературы

1. "ОБ УТВЕРЖДЕНИИ ПРЕДВАРИТЕЛЬНОГО НАЦИОНАЛЬНОГО СТАНДАРТА РОССИЙСКОЙ ФЕДЕРАЦИИ" от 23.10.2023 № 41-пнст // Официальный интернет-портал правовой информации. – 2023.
2. Дмитриева С. Умные (SMART) стандарты: первые итоги // Connect. – 2024. – №3-4. – С. 74-75.
3. Статус ПНСТ “Умные стандарты. Классификация” // Федеральная государственная информационная система Росстандарта,

URL: https://fgis.gost.ru/share/page/rsprs/nds-details?uuid=3d26b629-ee04-42c5-a09f-ea401cc776d1 (дата обращения: 07.09.2024)

4. Dabney J., Barber G., Ohi D. Estimating direct return on investment of independent verification and validation // Proceedings of the IASTED Conference on Software Engineering and Applications, Международная конф. (Cambridge, 9-11 ноя. 2004)
5. Amit Mondal. A Survey of Software Architectural Change Detection and Categorization Techniques / Banani Roy, Chanchal K. Roy, Kevin A. Schneider // Journal of Systems and Software – 2022. Vol. 194 – P. 75
6. Robert Woll. Semantic Integration of Product Data Models for the Verification of Product Requirements / Haygazun Hayka, Rainer Stark, Christian Geißler, C. Greisinger // Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment (London, 10 авг. 2012) – London, 2013 – P. 157-168.
7. Игонина, И. Н. Смарт стандарты – шаг в цифровое будущее химической отрасли / И. Н. Игонина, Л. А. Солод, Н. А. Моисеев // Успехи в химии и химической технологии. – 2023. – Т. 37, № 2(264). – С. 168-170.

8. Giorgio Ausiello, Luca Cabibbo. Expressiveness and Complexity of Formal Systems // Functional Models of Cognition. Theory and Decision Library – 1999. Vol. 27, P. 103–120

9. Martini, S. Gadducci F. Tavosanis M. Several Types of Types in Programming Languages // IFIP Advances in Information and Communication Technology. – Международная конф. (Pisa, 6 окт. 2016) – Pisa, 2016 – Vol. 487, P. 216-227

10. Glenn G. Chappell. A Primer on Type Systems // Образовательные материалы UEF к курсу CS 331. 29.01.2018.
URL: https://www.cs.uaf.edu/users/chappell/public_html/class/2018_spr/cs331/docs/types_primer.html (дата обращения: 13.09.2024)

11. Горелиц, Н. К. Управление требованиями к ответственным системам. Обзор решений / Н. К. Горелиц, Д. С. Кильдишев, А. В. Хорошилов // Труды Института системного программирования РАН. – 2019. – Т. 31, № 1. – С. 25-48.

12. Universally Unique IDentifiers (UUIDs) // Internet Engineering Task Force Datatracker URL: https://datatracker.ietf.org/doc/html/rfc9562 (дата обращения: 20.06.2024).

13. Garcia-Molina H., Ullman J. D., Widom J. Database systems: the complete book. 2nd ed. - Upper Saddle River, NJ: Pearson Prentice Hall, 2008. – 1248 p.

References

1. "ON APPROVAL OF THE PRELIMINARY NATIONAL STANDARD OF THE RUSSIAN FEDERATION" dated 10/23/2023 No. 41-pnst // Official Internet portal of legal information. – 2023.

2. Dmitrieva S. Smart standards: the first results // Connect. – 2024. – No.3-4. – pp. 74-75.

3. PNST status “Smart standards. Classification” // Federal State Information System of Rosstandart, URL: https://fgis.gost.ru/share/page/rsprs/nds-details?uuid=3d26b629-ee04-42c5-a09f-ea401cc776d1 (date of request: 09/07/2024)

4. Dabney J., Barber G., Ohi D. Estimating direct return on investment of independent verification and validation // Proceedings of the IASTED Conference on Software Engineering and Applications, International Conference (Cambridge, Nov 9-11. 2004)

5. Amit Mondal. A Survey of Software Architectural Change Detection and Categorization Techniques / Banani Roy, Chanchal K. Roy, Kevin A. Schneider // Journal of Systems and Software – 2022. Vol. 194 – P. 75

6. Robert Woll. Semantic Integration of Product Data Models for the Verification of Product Requirements / Haygazun Hayka, Rainer Stark, Christian Geißler, C. Greisinger // Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment (London, Aug 10, 2012) – London, 2013 – P. 157-168.

7. Igonina, I. N. Smart standards – a step into the digital future of the chemical industry / I. N. Igonina, L. A. Solod, N. A. Moiseev // Successes in chemistry and chemical technology. – 2023. – Vol. 37, No. 2(264). – pp. 168-170.

8. Giorgio Ausiello, Luca Cabibbo. Expressiveness and Complexity of Formal Systems // Functional Models of Cognition. Theory and Decision Library – 1999. Vol. 27, P. 103–120

9. Martini, S. Gadducci F. Tavosanis M. Several Types of Types in Programming Languages // IFIP Advances in Information and Communication Technology. – International Conference (Pisa, October 6, 2016) – Pisa, 2016 – Vol. 487, pp. 216-227

10. Glenn G. Chappell. A Primer on Type Systems // UEF educational materials for the CS 331 course. 29.01.2018. URL: https://www.cs.uaf.edu/users/chappell/public_html/class/2018_spr/cs331/docs/types_primer.html (date of reference: 09/13/2024)

11. Gorelits, N. K. Management of requirements for responsible systems. Review of solutions / N. K. Gorelits, D. S. Kildishev, A.V. Khoroshilov // Proceedings of the Institute of System Programming of the Russian Academy of Sciences. – 2019. – Vol. 31, No. 1. – pp. 25-48.

12. Universally Unique IDentifiers (UUIDs) // Internet Engineering Task Force Datatracker URL: https://datatracker.ietf.org/doc/html/rfc9562 (date of request: 06/20/2024).

13. Garcia-Molina H., Ullman J. D., Widom J. Database systems: the complete book. 2nd ed. - Upper Saddle River, NJ: Pearson Prentice Hall, 2008. – 1248 p.