9. Аналит. докл. «Подходы к формированию и запуску новых отраслей промышленности в контексте Национальной технологической инициативы, на примере сферы "Технологии и системы цифровой реальности и перспективные "человеко-компьютерные" интерфейсы (в части нейроэлектроники)». М.: Агентство стратегических инициатив – АСИ, 2015.

10. Колмогоров А. Н. Автоматы и жизнь // Кибернетика ожидаемая и кибернетика неожиданная / под ред. А. И. Берга, Э. Кольмана. Москва: Наука, 1968. С. 12–30.

11. Chomsky N. Syntactic Structures. The Hague: Mouton, 1957. (Переиздание: Chomsky N. Syntactic Structures. De Gruyter Mouton, 2002.)

12. Marr D. Vision. An informational investigation into the human representation and processing of visual information. Cambridge, Massachusetts, London, England: The MIT Press, 2010.

13. Станкевич Л. А. Искусственные когнитивные системы // Науч. сессия НИЯУ МИФИ. XII Всерос. науч.-техн. конф. «Нейроинформатика–2010». Лекции. М.: НИЯУ МИФИ, 2010. С. 106–160.

E. O. Kuznetsova, S. A. Petrenko
Saint Petersburg Electrotechnical University

COMPUTATIONAL COGNITIVISM-BASED APPROACH TO INTRUSION DETECTION

Explores the complex scientific problem of detecting intrusion in domestic departmental and corporate information systems. An approach to the creation of intrusion detection system based on the so-called «computational cognitivism» is proposed and substantiated – a relatively new scientific direction of research, in which cognition and cognitive processes are a kind of symbolic computation. It is shown that the cognitive approach makes it possible to create systems that fundamentally differ from the known systems for detecting, preventing and eliminating the consequences of computer attacks by the unique ability to independently associate and synthesize new knowledge about the qualitative characteristics and quantitative patterns of information countermeasures in cyberspace. The systematic appearance of a cognitive system for detecting intrusion based on computational cognitivism is proposed.

Information security, intrusion detection, intrusion detection system, computational cognitivism, quantitative patterns of countermeasures in cyberspace, converging NBIC-technologies, Big Data technologies

УДК 004.067

С. А. Аббас, А. И. Водяхо
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Н. А. Жукова
Федеральный исследовательский центр Российской академии наук

Архитектурное проектирование кибер-физических систем, построенных на платформах интернета вещей

Рассматриваются особенности архитектурного этапа проектирования кибер-физических систем, построенных на платформах туманных вычислений. Анализируются основные типы вариабельности, встречающиеся в современных кибер-физических системах, определяются архитектурно-значимые характеристики и приводятся типовые постановки задач архитектурного проектирования для рассматриваемого класса систем. Предлагается последовательность принятия архитектурных решений при проектировании кибер-физических систем с высоким уровнем вариабельности. Рассматриваются типовые механизмы управления вариабельностью. Оценивается влияние вариабельности на основные архитектурно значимые параметры – производительность, готовность, совокупную стоимость владения. Предлагаемый подход ориентирован, прежде всего, на проектирование крупномасштабных распределенных кибер-физических систем. Рассмотренный подход был использован при построении ряда реальных кибер-физических систем, принадлежащих различным предметным доменам – производственным системам, системе мониторинга состояния систем кабельного телевидения и т. д. В настоящее время авторы ведут работы по созданию системы управления образовательным контентом.

Архитектурное проектирование, вариабельность, кибер-физические системы, интернет вещей, туманные платформ

Актуальность. Прогресс техники и технологии позволяет создавать и применять все более сложные антропогенные системы. При этом их сложность выражается не только и не столько в

увеличении числа элементов и числа связей, усложнении поведения, но и в том, как постоянно изменяется структура и поведение подобных систем. Можно утверждать, что практически все сложные антропогенные системы способны изменять свою структуру и поведение в зависимости от внутренних и внешних событий. Общие тенденции развития антропогенных систем позволяют считать, что системы новых поколений в еще большей мере будут обладать свойством вариабельности структуры и поведения. Примером сложных антропогенных систем могут служить кибер-физические системы (Cyber-PhysicalSystems, CPS) [1], в которые входят не только программные элементы, но и другие компоненты – электро-механическое, технологическое оборудование, транспортные системы, природные и биологические системы, а также люди. Многие современные CPS строятся на платформах интернета вещей и промышленного интернета вещей [2].

Существующие подходы к проектированию CPS. Современные подходы к проектированию CPS можно охарактеризовать как интеграционные. С одной стороны, примером может служить достаточно популярная в последние годы парадигма проектирования Девопс (Development и Operations, DevOps) [3], ориентированная на уменьшение разрыва между этапами жизненного цикла (ЖЦ) – разработкой и постоянной модернизацией.

Данная тенденция просматривается уже в течение ряда лет. В частности, переход от водопадных к циклическим и спиральным моделям проектирования [4] можно рассматривать как попытки уменьшения семантического разрыва между отдельными фазами проектирования.

В качестве другого характерного примера уменьшения семантического разрыва между этапами ЖЦ программной системы можно рассматривать модель IT Library (ITIL) и реализующий ее фреймворк The Open Group Architecture Frame work (TOGAF) [5], [6], который описывает процесс постоянного совершенствования корпоративных информационных систем (КИС). В этом плане подход DevOps можно воспринимать как адаптацию ITIL-подхода применительно к другим типам распределенных систем. Следует заметить, что современные сложные CPS во многом похожи на КИС. В частности, их отличает высокая сложность: как КИС, так и CPS – это адаптивные системы со многими заинтересованными сторонами. Их модернизации проводятся на протяжении всего ЖЦ.

Таким образом, можно утверждать, что CPS и КИС – это достаточно близкие классы антропогенных систем, при этом CPS работают на более низком уровне. Еще одной точкой соприкосновения между CPS и КИС можно считать промышленный интернет вещей, на платформе которого строятся современные производственные системы, которые, в свою очередь, относятся к подсистемам КИС.

Реализации DevOps можно рассматривать как бизнес-процесс (БП), обеспечивающий поддержание и развитие КИС. В определенном смысле, процесс проектирования ИС, в частности CPS, можно рассматривать как БП. Таким образом, DevOps-подход можно рассматривать как процесс интеграции процессов проектирования CPS и TOGAF-процессов.

DevOps-подход относится к интеграции только двух этапов ЖЦ: разработке и модернизации, но это не весь ЖЦ.

В настоящее время используется много разных терминов, описывающих изменчивость структуры и поведения систем. В частности, применительно к архитектурам информационных и программных систем, к которым можно отнести и CPS, это «адаптивная», «гибкая», «реконфигурируемая», «динамическая», «вариабельность» (agility) и т. д. [7].

Вариабельность (Agility). Этот термин применительно к ИС в настоящее время обозначает способность адаптации системы к требованиям, определяемым некоторым контекстом. Термин «гибкий подход» (Agile Approach, AA) используется как применительно к процессу проектирования ИС (Agile Process, AP) [8], так и к архитектурам (Agile Architecture, AA) [7]. В течение ряда лет в начале 2010-х гг. активно обсуждался вопрос о соотношении этих понятий. Высказывались диаметрально противоположные мнения. С одной стороны, эти термины рассматривались почти как синонимы, а с другой, предлагалось считать эти понятия независимыми [9]. Ответ на этот вопрос, по мнению авторов, можно найти в определении понятия «архитектура» [10], в соответствии с которым архитектура определяется как

«фундаментальные понятия или свойства системы в ее среде, воплощаемые в ее элементах, отношениях, а также в принципах ее проектирования и эволюции». Из этого определения следует, что термин «архитектура» применим ко всем этапам жизненного цикла (ЖЦ) системы. Очевидно, что термин АР относится к этапу проектирования, а термин АА – к ЖЦ системы. На этапе проектирования АР предполагает адаптацию к требованиям заказчика, а на этапе эксплуатации система адаптируется к контексту, причем когда речь идет об АА, то обычно реализуется адаптация времени выполнения (runtime).

На рис. 1 приведена укрупненная классификация различных вариантов реализации вариабельности. Данная классификация создана с учетом классификации, приведенной в [11], но существенно от нее отличается. Понятие вариабельности можно связать с типом вариабельности и привязкой к отдельным этапам ЖЦ. При этом

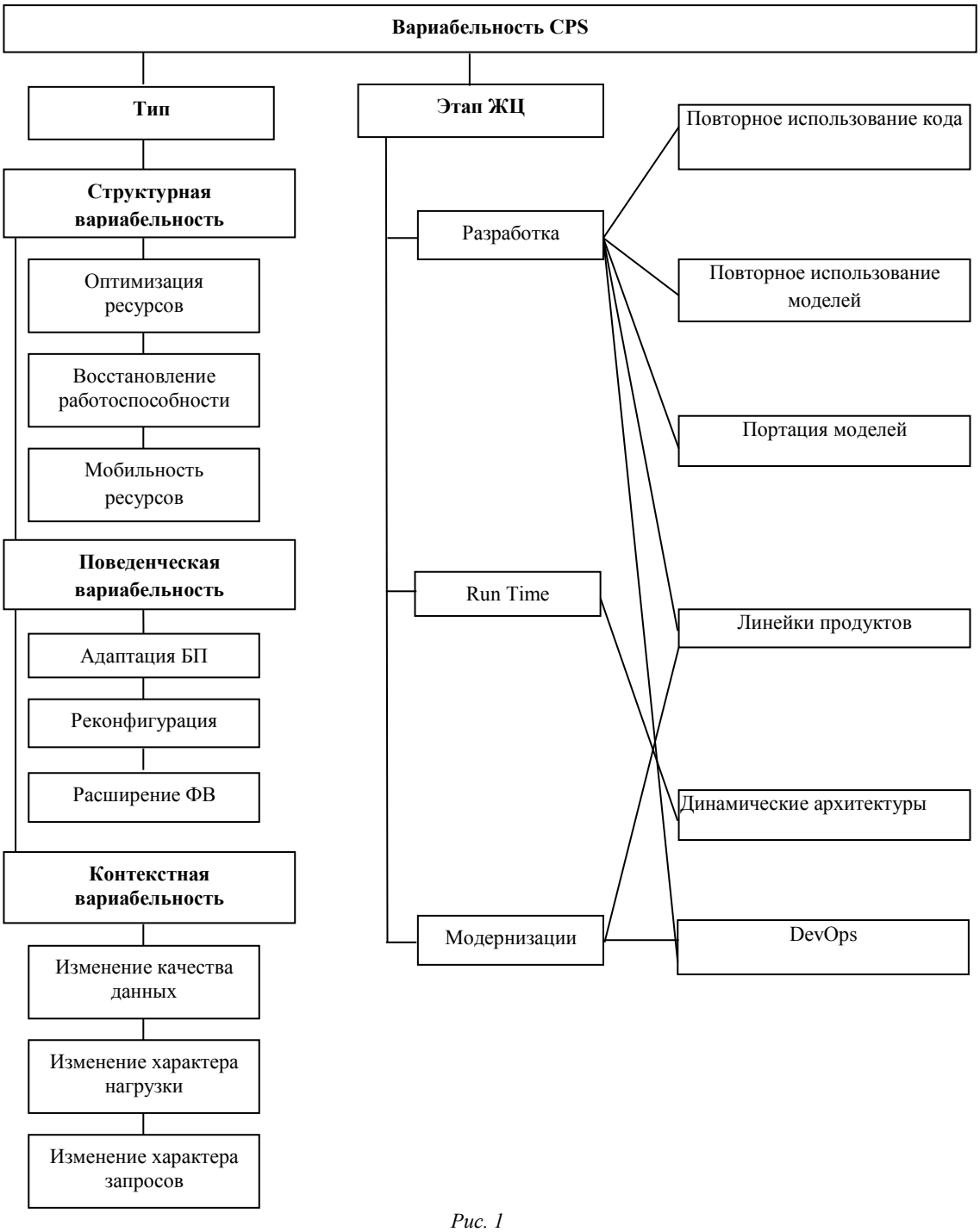

Рис. 1

можно выделить три основных типа вариабельности: структуры, поведения и контекста.

Изменение структуры может происходить в случаях, когда необходимо оптимизировать ресурсы, восстанавливать работоспособность системы после обнаружения неисправности или требуется реализация механизмов мобильности ресурсов.

Поведенческая вариабельность реализуется в случаях, когда требуется оптимизировать бизнес-процессы (БП), реконфигурировать БП по причине изменения структуры, расширить функциональные возможности. Понятие контекстной вариабельности связано с изменениями внешней среды – качества данных, характера нагрузки и поступающих от пользователей запросов.

Механизмы вариабельности могут относиться к разным этапам ЖЦ CPS: проектирования, эксплуатации, модернизации. На этапе проектирования могут использоваться такие реализации механизмов вариабельности, как повторное использование кода или моделей, линейки продуктов. С этапом эксплуатации связано использование разного рода динамических архитектур [12]. На этапе модернизации обычно используются DevOps [3] и линейки (семейства) продуктов.

Agile Architecture. По мнению авторов статьи, этот термин следует использовать в том случае, когда речь идет о вариабельности CPS на архитектурном уровне. Если еще раз обратиться к вышеприведенному определению архитектуры, то можно считать, что реализация agility предполагает возможность реконфигурирования как структурной, так и поведенческой составляющих. Следует заметить, что при проектировании AA не обязательно использовать AP, но механизмы реализации вариабельности закладываются на этапе проектирования, т. е. можно говорить о целесообразности введения в рассмотрение в ряде случаев отдельной архитектурной перспективы (Agility Perspective) [13]. Проектирование CPS с гибкой (Agile) архитектурой можно рассматривать как одновременное проектирование некоторого класса систем [7].

Какой уровень вариабельности необходим? Перефразируя вопрос из известной монографии по программной архитектуре [14], можно спросить, какой объем AA действительно требуется, поскольку понятно, что ее реализация не бесплатна. Оценить затраты и положительный эффект можно следующим образом. Затраты можно оценить в терминах дополнительных затрат на разработку, на дополнительное оборудование и возможную потерю производительности, а положительный эффект – через функцию полезности, как это предлагается в [9], [13], [15].

Кроме того, при оценке полезности вариабельности действенным может оказаться подход, развиваемый в [9].

Кибер-физическая контекстно-ориентированная система (Context Aware System, CAS). По существу, всякая AA-система – CAS [16]. При этом следует различать статическую и динамическую вариабельность (рис. 2). На этапе разработки реализуется вариабельность времени разработки (Development time agility, DTA), а на этапе эксплуатации – времени выполнения (Runtime Agility, RTA).

Модель поддержки механизмов вариабельности. Несмотря на существование AA, авторы не могут назвать работы, в которых предлагаются конкретные технические решения.

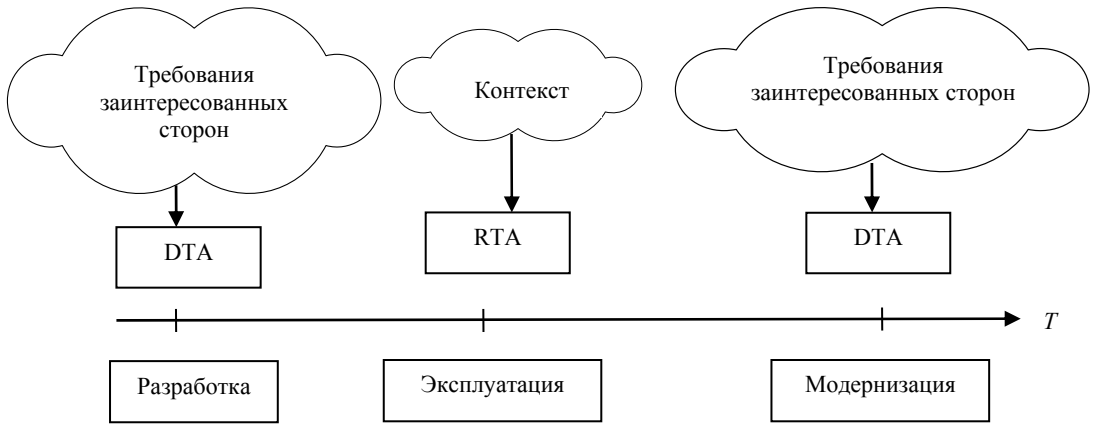

Рис. 2

Само понятие вариабельности предполагает, что одной системе соответствует несколько архитектур, а точнее несколько архитектурных состояний (АРС), переход между которыми происходит под воздействием внутренних или внешних событий. В первом случае можно говорить о реализации механизмов self* (selftesting, selfhealing-systems и т. п.), а во втором случае – о CAS [16].

В качестве модели гибкой системы можно использовать многоуровневый относительно конечный автомат, который описывает реализацию механизмов в терминах перехода между АРС [17].

К сожалению, подобрать адекватную меру для оценки понятия вариабельности достаточно сложно. В качестве грубой оценки уровня вариабельности можно использовать число АРС.

Задача проектирования АА CPS. Современные CPS – это многоуровневые системы с разными парадигмами и моделями на разных уровнях. Кроме того, для каждого уровня используются различные оценки стоимости реализации архитектурно-значимых характеристик (АЗХ).

При таком подходе задача проектирования АА CPS представляется далеко не тривиальной. В общем виде ее можно сформулировать следующим образом: требуется построить оптимальную в некотором смысле 5-уровневую систему, на каждом уровне которой функционируют АА-системы, которые могут находиться в разных АРС. Таким образом, для разных уровней предлагается использовать разные методы проектирования.

Более строго задачу проектирования АА CPS можно сформулировать следующим образом: с помощью процедуры синтеза (проектирования) S найти многоуровневую архитектурную модель A, для которой достигается экстремум основного показателя эффективности многоуровневого синтеза при ограничениях на вспомогательные показатели M:

$$A_0 = \text{Argextr}\, W\left(S\left(A_v\right),\ M(A_v)\right), \qquad (1)$$

где A_0 – требуемая архитектура; W – основной показатель эффективности; M – вспомогательные показатели $v \in V$ – множество возможных архитектур (в данном случае в качестве архитектуры выступает АРС); A_v – множество возможных архитектурных решений.

Основному показателю эффективности может быть поставлена в соответствие архитектурная тактика (АТАК) [15]. Выражение (1) описывает процесс проектирования на верхнем уровне. Число уровней обычно равняется 6, каждому из них соответствует свой набор АРС, и в общем случае для одной и той же архитектуры верхнего уровня, как будет показано далее, могут использоваться разные основные показатели эффективности. Для всех уровней (кроме верхнего) можно записать:

$$A_i = \text{Argextr}\, W_i\left(S_i\left(A_v\right),\ M\left(A_{iv}\right)\right),\ i = 1, 4. \quad (2)$$

Если принять во внимание тот факт, что в процессе проектирования реализуемая функциональность постоянно перераспределяется между уровнями, то понятно, что взаимосвязи между ними могут быть достаточно сложными.

Взаимосвязи между характеристиками, относящимися к разным уровням, в общем виде определить непросто, поскольку эти взаимосвязи в значительной мере определяются выбранной стратегией проектирования [13], [15].

Архитектурно-значимые параметры (АЗП). Параметры (характеристики) ИС определены в стандарте [18]. Для АА CPS эти параметры можно конкретизировать (рис. 3). Показатели качества (характеристики) можно разделить на те, которые важны для разных заинтересованных сторон, в частности для потенциальных пользователей, и характеристики собственно АА CPS.

К первой группе можно отнести функциональное соответствие, эффективность, свободу от рисков, гибкость, надежность, производительность, защищенность, доступность, сложность, расширяемость, TCO и т. п.

Ко второй – сложность, быстродействие, объем собираемых данных, динамичность структуры и поведения, защищенность, доступность, расширяемость, сопровождаемость. Интерпретация этих характеристик соответствует стандарту [18].

АЗП можно определить как параметры, в терминах которых формируются задачи архитектурного (высокоуровневого) этапа проектирования. Более конкретно этот этап проектирования можно определить как этап, на котором на основе требований формируется архитектурное описание в соответствии со стандартом [18].

Типовые постановки задач в терминах основных (W) и вспомогательных (M) параметров приведены в табл. 1.

Рис. 3

Таблица 1

W	*M*
Сложность	Доступность, гибкость, ФВ, безопасность
Гибкость	Сложность, ФВ, производительность, безопасность
Производительность	Сложность, TCO, ФВ
Доступность	Свобода от рисков, гибкость TCO, ФВ
ФВ	Гибкость, TCO, производительность
TCO	Сложность, гибкость, производительность
Свобода от рисков	Доступность, гибкость, TCO
Безопасность	Доступность, гибкость, TCO

ФВ – функциональные возможности; TCO (Total Costof Ownership) – совокупная стоимость владения

Использование предлагаемого модельного подхода целесообразно в случаях, когда присутствуют требования, связанные с высоким уровнем гибкости и масштабируемости или готовности, которая достигается через адаптивность.

Как указывалось ранее, для разных уровней CPS используются разные модели и, соответственно, для разных уровней определяются разные АЗП. К сожалению, нельзя утверждать, что $ch_i \in CH$, но можно утверждать, что $ch_{ij} = f(ch_{i,j-1})$. Другими словами, можно констатировать, что зависимости между одноименными параметрами, относящимися к разным уровням, могут быть самыми разными. Например, если на сенсорном уровне оказывается невозможным реализовать предварительную обработку по причине ограничений на передаваемые данные, то эту обработку приходится перенести на туманный уровень, т. е. изменить функциональность туманного уровня при одновременном изыскании возможностей расширения каналов связи между сенсорным и туманным уровнями.

Реализация механизмов вариабельности предполагает, что проектируется не один экземпляр системы, а некоторый класс систем. Таким

образом, задачу проектирования AA CPS можно сформулировать следующим образом: требуется спроектировать шестиуровневую CPS, на каждом уровне которой проектируемая система описывается множеством АРС, причем для разных сегментов могут использоваться разные модели и разные подхарактеристики одних и тех же характеристик из стандарта [18].

Относительно проектирования каждого из уровней, модель данного уровня, которая выступает в качестве результата проектирования,

$$ML_i = \langle CH_i, R_i, MM_i \rangle,$$

где CH_i – множество характеристик i-го уровня; R_i – требования, предъявляемые к характеристикам i-го уровня; MM_i – метамодель i-го уровня, определяющая правила формирования моделей i-го уровня.

Общую модель, в качестве которой выступает совокупность уровневых моделей, можно определить как

$$M_g = \langle ML_0,\ ML_1,\ ML_2,\ ML_3,\ ML_4,\ ML_5,\ CH_G,\ R_G,\ RT_{i-i+1} \rangle,$$

где M_g – целевая архитектура; CH_G – характеристики целевой архитектуры; R_G – требования, предъявляемые к целевой архитектуре; RT_{i-i+1} – система трансформации требований для перехода между уровнями.

Задачу архитектурного проектирования в этом случае можно определить как $R_G \to M_g$. Использование строгих математических методов на архитектурном этапе проектирования, по мнению многих авторов, не представляется возможным. Для решения этой задачи предлагается использовать традиционные подходы, основанные использовании АТАК [15] с учетом многоуровневости системы CPS. Под АТАК понимается архитектурное решение, принимаемое в определенном контексте с целью улучшения одного конкретного параметра, в качестве которого выступает W.

Типовые варианты постановки задачи проектирования CPS. Можно выделить следующие типовые варианты постановки задачи проектирования AgileAmI CPS:

$$\text{TCO} \to \min \text{ при } P > P_{\text{треб}},\ AV > AV_{\text{треб}},$$
$$\text{ФВ} = \text{ФВ}_{\text{треб}},\ FoR > FoR_{\text{треб}};$$
$$P \to \max \text{ при } \text{ФВ} = \text{ФВ}_{\text{треб}},\ AV > AV_{\text{треб}},$$
$$\text{TCO} > \text{TCO}_{\text{треб}},\ FoR > FoR_{\text{треб}};$$
$$AV \to \max \text{ при } \text{ФВ} = \text{ФВ}_{\text{треб}},\ AV > AV_{\text{треб}},$$
$$\text{TCO} > \text{TCO}_{\text{треб}},\ FoR > FoR_{\text{треб}};$$
$$FoR \to \max \text{ при } \text{ФВ} = \text{ФВ}_{\text{треб}},$$
$$AV > AV_{\text{треб}},\ \text{TCO} > \text{TCO}_{\text{треб}},$$

где P – производительность (эта характеристика обычно встречается в форме время отклика или число обрабатываемых запросов); AV – надежность, чаще всего выступает в форме доступности; FoR – свобода от рисков

Реже встречаются постановки типа:

$$\text{SEQ} \to \max \text{ при } \text{ФВ} = \text{ФВ}_{\text{треб}},$$
$$AV > AV_{\text{треб}},\ \text{TCO} > \text{TCO}_{\text{треб}},$$
$$T_2D \to \max \text{ при } \text{TCO} > \text{TCO}_{\text{треб}},$$
$$\text{ФВ} = \text{ФВ}_{\text{треб}},\ AV > AV_{\text{треб}},$$

где SEQ – требования по безопасности; T_2D – время разработки.

Могут встречаться и другие постановки.

В табл. 2 приведены типовые основные (W) и вспомогательные (M) показатели эффективности для разных уровней CPS. Эти данные получены по результатам анализа последних конференций по IoT, Fog Computing и CPS. Данные достаточно приблизительные, но дают представление об общей картине.

Таблица 2

Уровень	*W*	*M*
Сенсорный	Энергопотребление	Время отклика. Доступность. ФВ
Сенсорный	Доступность	ДоступностьВремя отклика
Сенсорный	Безопасность	Свобода от рисков. Доступность. Время отклика TCO
Сенсорный	Время отклика	Доступность
Туманный	Производительность	ФВ. TCO
Туманный	Время отклика	Доступность
Туманный	TCO	ФВ. Доступность
Облачный	TCO	Время отклика
Облачный	Время отклика	TCO
CPS	Производительность	ФВ. Доступность. Свобода от рисков
CPS	Безопасность	Доступность. Свобода от рисков
CPS	Свобода от рисков	Доступность. Безопасность
CPS	TCO	ФВ
SoCPS	TCO	ФВ
SoCPS	Безопасность	Доступность. Свобода от рисков
AMI	Свобода от рисков	Доступность. Безопасность
AMI	ФВ	Доступность. Безопасность. Свобода от рисков

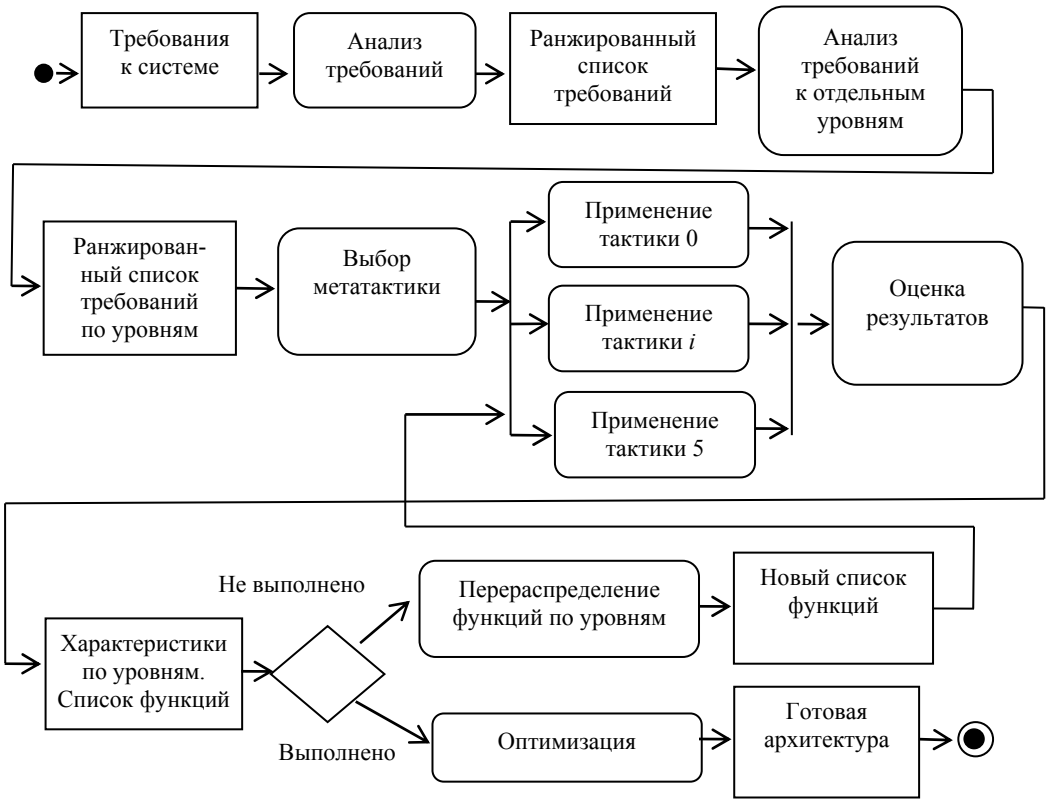

Рис. 4

Типовые подходы к последовательности принятия архитектурных решений. Типовая процедура проектирования CPS показана на рис. 4.

Процесс проектирования начинается с анализа требований. В результате анализа формируется ранжированный список требований, на базе которого формируется первая версия ранжированного списка требований по уровням. На основании этого списка выбирается метатактика, которая определяет порядок применения отдельных тактик. Затем осуществляется оценка результатов.

Применение *i*-й тактики предполагает выполнение следующих действий.

Шаг 1. Формулировка требований в терминах *i*-го уровня.

Шаг 2. Анализ требований, выделение и ранжирование архитектурно значимых требований для *i*-го уровня.

Шаг 3. Анализ списка требований и определения задач моделирования с целью получения нужных характеристик.

Шаг 4. Моделирование.

Стоимость работ по этому этапу можно оценить как

$$C_{\text{tr}} = C_{\text{an}} + C_{\text{ar}} + C_{\text{mod}},$$

где C_{tr} – стоимость реализации; C_{an} – затраты на работу аналитика (работа с требованиями); C_{ar} – затраты на работу архитектора (выбор архитектуры, составление архитектурного описания) и C_{mod} – затраты на моделирование. Наиболее непредсказуем компонент C_{mod}, особенно если речь идет о натуральном моделировании, – это основные архитектурные риски, связанные с данным этапом.

Если требования выполняются, то осуществляется оптимизация, которая предполагает возможность коррекции отдельных характеристик, например с целью расширения ФВ или повышения гибкости. Фактически это процедура использования возникших запасов по отдельным характеристикам и уровням.

Чаще всего с первого раза не удается удовлетворить все требования. В таком случае функции перераспределяются по уровням. Они могут «мигрировать» как вверх, так и вниз. Например, при невозможности получить требуемое время отклика часть функционала может быть реализована аппаратно. Если позволяет быстродействие, то часть функций может быть перенесена на туманный уровень. Для нового списка функций по уровням запускаются архитектурные тактики, и процесс повторяется итеративно до получения положительного результата.

Типовые метатактики. Метатактику *MMT* можно определить как

$$MMT = \langle MT, MTR \rangle,$$

где *MTR* – правила (сторожевые условия) использования *MT*.

Метатактики можно определить как

$$MT = \langle T, TR \rangle,$$

где *T* – тактики; *TR* – правила (сторожевые условия) использования *T*.

Можно выделить по крайней мере следующие группы метатактик:

– все уровни CPS отрабатываются параллельно;

– все уровни CPS отрабатываются последовательно сверху вниз;

– все уровни CPS отрабатываются последовательно снизу вверх;

– уровни CPS отрабатываются в соответствии со сложностью достижения некоторого архитектурно-значимого параметра.

Типовые механизмы управления вариабельностью. Понятие вариабельности не определяется в стандартах, но влияет на ряд стандартизованных характеристик качества.

На рис. 5 показаны связи вариабельности с другими характеристиками, которые можно рассматривать как архитектурно значимые применительно к AA CPS. Очевидно, что повышение уровня вариабельности приводит к повышению уровня сложности. Целесообразно ввести дополнительные механизмы вариабельности посредством аналитического метода, известного как метод «Стоимость/Выигрыш» (Cost Benefit Analysis Method, CBAM) [15]. Идея CBAM состоит в том, что с помощью экспертных оценок определяются последствия принимаемых архитектурных решений. В результате оценок экспертов получаются графики, которые описывают значение каждого параметра и его ценность, причем ценность может определяться не только в экономических параметрах. В большинстве случаев для каждого уровня требуются собственные оценки.

Использование механизмов вариабельности позволяет повысить уровень готовности, которая важна для CPS как характеристика надежности. Эффект может достигаться за счет реконфигурации. Для оценки эффективности в этом случае также можно использовать подход CBAM.

Использование механизмов вариабельности позволяет в ряде случаев повысить производительность. Это связано прежде всего с реализацией механизмов балансировки нагрузки и механизмов контентной и контекстной адаптации.

Вариабельность можно рассматривать как средство реализации механизмов адаптивности, что относится также к масштабируемости и модифицируемости.

Влияние вариабельности на пропускную способность каналов связи напрямую крайне ограничено. Однако вариабельность через адаптивность может влиять на характер трафика в сторону его уменьшения и сглаживания.

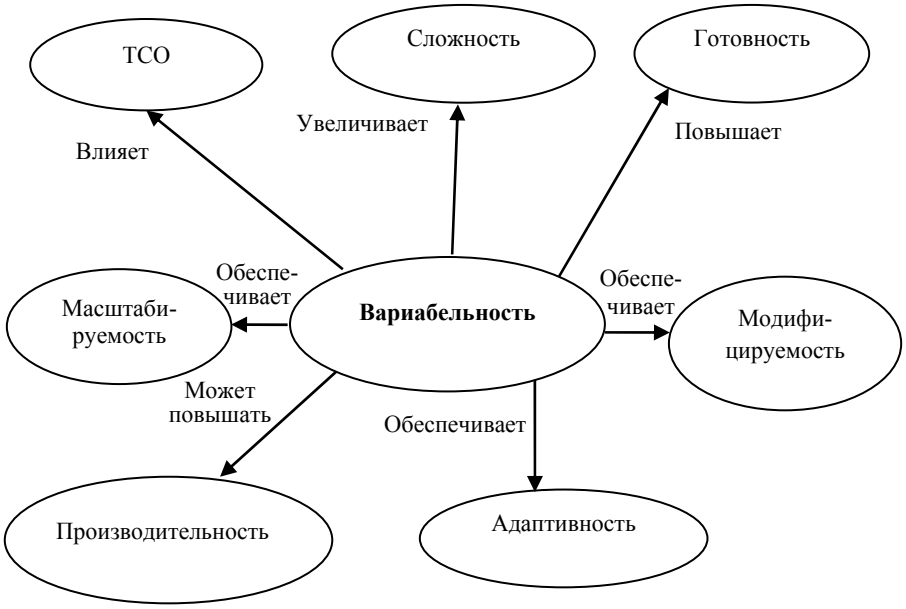

Рис. 5

Влияние вариабельности на TCO неоднозначно. С одной стороны, реализация вариабельности приводит к усложнению системы, т. е. к увеличению ее стоимости, а с другой, позволяет снизить стоимость за счет увеличения срока эксплуатации и удешевления процедуры модернизации. TCO можно определить как

$$\mathrm{TCO} = C_{\mathrm{cr}} + C_{\mathrm{mt}} + C_{\mathrm{mod}},$$

где C_{cr}, C_{mt}, C_{mod} – стоимости создания, поддержки, модификации системы. Очевидно, что в плане стоимости создания, системы, в которых не реализуются механизмы вариабельности, будут выигрывать. В отношении расходов на эксплуатацию, корректно спроектированные системы, реализующие механизмы вариабельности, не должны уступать обычным системам. Таким образом в качестве условий получения можно определить $\Delta C_{\mathrm{cr}} < \Delta C_{\mathrm{mod}},$ где Δ – разница в стоимости.

Предлагаемый подход ориентирован на проектирование достаточно крупных распределенных CPS и напрямую не связан ни с одним из предметных доменов. Следует заметить, что предлагаемый ход достаточно общий. Это касается, прежде всего моделей, которые применительно к специфике решаемой задачи можно существенно упростить, например автоматной модели верхнего уровня и многоуровневой структурно-поведенческой модели.

Описанный подход был использован авторами для построения реальных систем, в число которых входят производственная система автоматизированного сборочно-сварочного участка, система мониторинга состояния систем кабельного телевидения. В настоящее время авторы ведут работы по созданию системы управления образовательным контентом для одного из университетов, предназначенных для поддержки динамических образовательных траекторий.

СПИСОК ЛИТЕРАТУРЫ

1. Sanfelice R. G. Analysis and design of cyber-physical systems. a hybrid control systems approach // Cyber-Physical Systems: From Theory to Practice / D. Rawat, J. Rodrigues, I. Stojmenovic. Taylor & Francis Group, Boca Raton, FL, 2016. P. 3–31. URL: https://hybrid.soe.ucsc.edu/sites/default/files/preprints/105.pdf.
2. Perry L. Internet of Things for architects. Birmingham, UK: Packt Publishing, 2018.
3. Len Bass, Ingo Weber, Liming Zhu. DevOps A Software Architect's Perspective. NJ: Addison-Wesley Professional, 2015. URL: http://ptgmedia.pearsoncmg.com/images/9780134049847/samplepages/9780134049847.pdf (дата обращения 27.04.2021).
4. Орлов С. А., Цилькер Б. Я. Технологии разработки программного обеспечения. Учебник для вузов. 4-е изд. СПб.: Питер, 2012.
5. Ингланд Р. Овладевая ITIL. М.: Лайвбук, 2011.
6. The Open Group architecture framework. URL: http://www.opengroup.org/togaf (дата обращения 26.04.2021).
7. Muhammad Ali Babar, Brown A. W., Mistrik I. Agile software architecture aligning agile processes and software architectures. MA, USA: Morgan Kaufmann, 2014.
8. Principles behind the Agile Manifesto URL: http://agilemanifesto.org/principles.html (дата обращения 27.04.2021).
9. Richards M., Ford N. Fundamentals of software architecture an engineering approach. Sebastopol, CA: O'Reilly Media, 2020.
10. Standard ISO/IEC/IEEE 42010 Systems and software engineering – Architecture description. URL: http://www.iso.org (дата обращения 27.04.2021).
11. Bloomberg J. The Agile architecture revolution: How Cloud computing, REST-based SOA, and mobile computing are changing enterprise IT. New Jersey: Wiley & Sons, 2013.
12. Thomas A., Becker J. Dynamically reconfigurable systems: Architectures, design methods and applications / ed. by M. Platzner, J. Teich, N. Wehn. NY: Springer, 2010.
13. Rozanski N., Woods E. Software systems architecture: Working with stakeholders using viewpoints and perspectives // Viewpoints. 2005. Vol. 8, № 2. P. 576.
14. Fairbanks G. Just enough software architecture. Co, USA: Marshall&Brainerd, 2010.
15. Bass L., Clements P., Kazman R. Software architecture in practice. 3rd ed. / Upper Saddle River. NJ: Addison-Wesley. 2013.
16. Temdee P., Prasad R. Context-aware communication and computing: Applications for smart environment. Cham, Switzerland: Springer Intern. Publishing AG, 2018.
17. Distributed technical object model synthesis based on monitoring data / V. Yu. Osipov, A. I. Vodyaho, N. A. Zhukova, M. Tianxing, S. Lebedev // Intern. J. of Knowledge and Systems Science (IJKSS). 2019. № 3. Art. 3. P. 27–43.
18. ISO/IEC 25010, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. URL: https://docs.cntd.ru/document/1200121069 (дата обращения 27.04.2021).

S. A. Abbas, A. I. Vodyaho
Saint Petersburg Electrotechnical University

N. A. Zhukova
Saint Petersburg Federal Research Centre of the Russian Academy of Sciences (SPCRAS)

ARCHITECTURAL DESIGN OF CYBER-PHYSICAL SYSTEMS BUILT ON THE INTERNET OF THINGS PLATFORMS

Discusses the architectural stage of designing cyber-physical systems built on fog computing platforms. The main types of variability encountered in modern cyber-physical systems are analyzed, the architecturally significant characteristics are determined, and typical statements of architectural design problems for the class of systems under consideration are given. The sequence of architectural decision-making in the design of cyber-physical systems with a high level of variability is proposed. Typical mechanisms for managing variability are considered. The impact of variability on the main architecturally significant parameters, such as performance, availability, and total cost of ownership, is evaluated. The proposed approach focuses primarily on the design of large-scale distributed cyber-physical systems. The considered approach was used in the construction of a number of real cyber-physical systems belonging to various subject domains, such as production systems, a system for monitoring the state of cable television systems. currently, the authors are working on the creation of an educational content management system.

Architectural design, agility, cyber-physical systems, Internet of things, fog platforms

УДК 004.9, 004.42

А. С. Букунов
Санкт-Петербургский политехнический университет Петра Великого

Автоматизированная система для управления строительными проектами

Описывается автоматизированная система, которая позволяет существенно упростить процесс управления проектами в строительстве и может быть использована при информационном моделировании зданий (BIM – Building Information Modeling). В системе предусмотрены различные возможности для пользователей, обладающих разными правами доступа к информации, хранящейся в базе данных приложения. Для пользователей приложения были реализованы следующие роли: администратор, BIM-менеджер, коммерческий менеджер, менеджер планового отдела, специалист снабжения, архитектор, проектировщик-конструктор, инженер по сетям, строитель. Система реализована в виде веб-приложения, написанного на языке PHP с помощью фреймворка Yii. Для хранения информации об используемых при реализации строительного проекта чертежах, оборудовании, материалах и т. п. была спроектирована и реализована реляционная база данных (БД), работа с которой осуществляется с помощью системы управления базами данных (СУБД) MySQL. Для решения некоторых задач использовались языки HTML и CSS. Работа с системой не требует установки дополнительного программного обеспечения, для этого достаточно доступа к сети Интернет.

Автоматизация бизнес-процессов, автоматизированная информационная система, информационное моделирование зданий, веб-приложение, база данных

В настоящее время как в России, так и во всем мире происходит бурное развитие BIM-технологий [1]. Проблемы, с которыми столкнулись разработчики BIM-систем, оказались настолько глобальными, что несмотря на колоссальные усилия многочисленного числа разработчиков по всему миру, до сих пор не создано ни одной автоматизированной информационной системы, которую можно было бы назвать действительно полноценной BIM-системой, связываю-