ISSN 2500-316X (Online)

https://doi.org/10.32362/2500-316X-2020-8-5-19-33

УДК 334.71: 656: 338.245

Поддержка жизненного цикла программных компонент

С.А. Кудж,
В.Я. Цветков@,
И.Е. Рогов

МИРЭА – Российский технологический университет, Москва 119454, Россия
@Автор для переписки, e-mail: cvj2@mail.ru

Современная разработка программного обеспечения основана на системном подходе, при котором программа или программный комплекс рассматривается как система взаимодействующих программных компонент. Модели программных компонент являются аналогами подсистем сложной системы. В силу этого сложную программу рассматривают как систему программных компонент. Организация структуры программных компонент влияет на качество и результат действия программы. Организация взаимодействия программных компонент влияет на эффективность работы программы. Важным фактором системы программных компонент является жизненный цикл, который определяет эффективность и целесообразность применения данной программы. Программное обеспечение отличается от многих сложных систем и информационных систем тем, что обладает возможностью увеличения своего жизненного цикла. При этом необходимость увеличения жизненного цикла характеризуется двумя факторами: внешним и внутренним. Внутренний фактор возникает при моральном старении программы. В этом случае она не соответствует новым условиям, например, новой операционной системе. Внешний фактор возникает при внешнем воздействии в виде помех или целенаправленных действий, типа компьютерных вирусов. Проблема создания структуры программных компонент вычислительных комплексов и информационных систем, обеспечивающих длительность жизненного цикла при наличии внешних воздействий, является актуальной. Исследование данной проблемы способствует совершенствованию технологической базы вычислительных комплексов и информационных систем, решающих прикладные задачи. В статье представлена новая модель жизненного цикла, основанная на двух моделях роста и деградации. Предложен

ресурсный подход для оценки жизненного цикла. В качестве аналитического решения предлагается использовать логистическое уравнение, которое достаточно хорошо описывает механизмы процесса формирования жизненного цикла. В статье рассматриваются три вида ресурса при вычислениях: физический, технологический и коммуникационный. Общим решением резервирования предлагается создание сети с включением модели мультиграфа.

Ключевые слова: программное обеспечение, программные компоненты, жизненный цикл, резервирование, составная модель, информационные ресурсы, логистическое уравнение.

Для цитирования: Кудж С.А., Цветков В.Я., Рогов И.Е. Поддержка жизненного цикла программных компонент. *Российский технологический журнал*. 2020;8(5):19-33. https://doi.org/10.32362/2500-316X-2020-8-5-19-33

Life cycle support software components

Stanislav A. Kudzh,
Viktor Ya. Tsvetkov@,
Igor E. Rogov

MIREA – Russian Technological University, Moscow 119454, Russia
@Corresponding author, e-mail: cvj2@mail.ru

Modern software development is based on a systems approach, in which a program or software complex is considered as a system of interacting software components. Models of software components are analogs of complex system subsystems. Therefore, a complex program is considered as a system of software components. The organization of the structure of software components affects the quality and result of the program. The organization of interaction between software components affects the efficiency of the program. An important factor in the system of software components is the life cycle, which determines the effectiveness and feasibility of using this program. Software differs from many complex systems and information systems in that it has the ability to increase its life cycle. Moreover, the need to increase the life cycle is characterized by two factors: external and internal. The internal factor arises due to the obsolescence of the program. In this case, it does not meet the new conditions, for example, a new operating system. The external factor arises from external influences in the form of interference or purposeful actions, such as computer viruses. The problem of creating the structure of software components of computing systems and information systems that ensure the duration of the life cycle in the presence of external influences is topical. The study of this problem contributes to the improvement of the technological base of computing systems and information systems that solve applied problems. The article presents a new life cycle model based on two models of growth and degradation. The article recommends a resource-based approach for life cycle assessment. As an analytical solution, it is proposed to use a logistic equation, which describes the mechanisms of the life cycle formation process quite well. The article discusses three types of resource in calculations: physical, technological and communicative. A general redundancy solution is proposed to create a network with the inclusion of a multigraph model.

Keywords: software, software components, lifecycle, redundancy, composite model, information resources, logistic equation.

For citation: Kudzh S.A., Tsvetkov V.Y., Rogov I.E. Life cycle support software components**.** *Rossiiskii tekhnologicheskii zhurnal = Russian Technological Journal.* 2020;8(5):19-33 (in Russ.). https://doi.org/10.32362/2500-316X-2020-8-5-19-33

Введение

Отличительными особенностями разработки современных программных компонент и программного обеспечения являются вычислительная интеграция, распределенная коммуникация, допустимый параллелизм процессов, наличие внешних и внутренних помех вычислительному процессу, зависимость жизненного цикла (ЖЦ) от внешних и внутренних условий, резервирование вычислительного процесса и другое. Программные компоненты (ПК) можно рассматривать как относительно независимые системы, решающие одну или несколько вычислительных задач, которые являются частью программного обеспечения вычислительных систем или информационных систем (ИС). Они могут функционировать автономно и в составе комплексов программных компонент. Аналогом ПК может являться искусственный нейрон или мультиагент. При организации в комплекс программных компонентов они могут создавать синергетический эффект, несвойственный отдельному ПК. Важным этапом создания программного компонента является разработка концептуального описания информационной услуги или информационной конструкции [1], которая включает проект модели жизненного цикла.

Анализ причин, влияющих на качество информационных услуг, позволяет сделать вывод о том, что порой даже взгляды потребителя и производителя программного обеспечения на вопросы ценности соответствующих программно-аппаратный комплексов различны. Ценность информационной услуги необходимо рассматривать при формировании потребительной стоимости информационного компонента как с позиций производителя, так и потребителя.

И, наверное, самым важным является изучение преимуществ информационной услуги с позиций ее жизненного цикла. Здесь следует выделить три основных направления:

1) проектирование, разработка и формирование информационной услуги или информационной конструкции программно-аппаратного обеспечения;

2) предоставление информационной услуги;

3) последующий реинжиниринг или регенерация информационной услуги.

При этом программные компоненты могут быть активными и пассивными. Активные ПК выполняют функции вычисления, анализа, сравнения, идентификации. Важной характеристикой активных ПК является их жизненный цикл, связанный с живучестью и эффективностью ПК. Отдельные программные компоненты, комплексы программных компонент (КПК) и программное обеспечение (ПО) обладают возможностью увеличения жизненного цикла. Возможность увеличения жизненного цикла характеризует гибкие технологические системы и саморазвивающиеся системы. Например, применение новой операционной системы без изменения ПК и КПК может увеличивать их жизненный цикл. Это пример поддержки ЖЦ с помощью внешних ресурсов.

При этом уровень технологических процессов, обеспечивающих соответствующее качество и максимальную доступность программных компонент в рамках предоставления информационных услуг, состоит из последовательности технологических операций, направленных на реализацию информационных сервисов в пределах ЖЦ.

1. Методология исследования

Основой исследования являются системный анализ, структурный анализ, сравнительный анализ и качественный анализ. В качестве материалов в данной работе исполь-

зованы публикации в области разработки и управления программными компонентами, а также материалы по технологиям применения мультиагентных систем и искусственных нейронных сетей.

Первое направление исследования – анализ ценности программного обеспечения, который может включать разные наборы программных компонент, как «идеальный продукт», поэтому издержки на его производство и цена совпадают.

Второе направление предполагает, что с позиции производителя программного обеспечения на этапе разработки определяется ожидаемая ценность, в то же время издержки производства программных компонент должны включать затраты на планируемые внутренние и внешние «дефекты».

Третье направление исследования – это последующий реинжиниринг или регенерация ПК и КПК, что также увеличивает жизненный цикл. Однако увеличение ЖЦ возможно не всегда, а только при определенных условиях. Поэтому исследование ЖЦ и методов его поддержки является актуальной проблемой для ПК, ПО и ИС. За рубежом применяют термин Systematic Literature Review (SLR) [2] как основу для внесения изменений в программное обеспечение и обновление ПК. Целью настоящей работы является выявление и анализ факторов, которые влияют на жизненный цикл ПО и ПК в контексте управления программным обеспечением.

2. Результаты исследований

2.1. Концепции разработки программных компонент

Программное обеспечение и программные компоненты связаны между собой. Программное обеспечение является более общим объектом и может включать разные наборы программных компонент, но при этом решать одну главную задачу. Программный компонент может быть рассмотрен как система, в силу чего к нему могут быть применены системный анализ и системное проектирование. Как всякая система, ПК имеет жизненный цикл. Комплексы программных компонент, как более сложная система по отношению к ПК, имеют свой жизненный цикл. Комплексы программных компонент, решающих одну задачу, могут иметь разный уровень надежности и разные периоды ЖЦ. Поэтому анализ и комплексирование программных компонент направлен не только на решение вычислительной задачи, но и на обеспечение надежности и устойчивости вычислений. Устойчивость вычислений зависит от внешних и внутренних воздействий, которые изменяют жизненный цикл вычислительной системы и жизненный цикл совокупности программных компонент. Планирование и управление разработкой программных компонент является важной задачей и отражено в ряде глобальных проектов, одним из которых является проект разработки глобального программного обеспечения (Global Software Development – GSD). В ряде работ [2–4] отмечено, что в процессе разработки необходимо решать вопросы нормального функционирования ПК, распределения вычислительных задач и управления вычислительным процессом. Нормальное функционирование и управление вычислительным процессом зависят от жизненного цикла ПК.

На сегодняшний день существует несколько моделей, поддерживающих функционирование ПК, в том числе модели на основе критериев жизненного цикла [5]. Результаты

исследования SLR показывают, что жизненный цикл КПК и ПК существенно зависит от потребленных, потребляемых и резервируемых ресурсов. При формировании модели ЖЦ программных компонент авторы пришли к выводу, который согласуется с [6] и рядом других работ, что современные программные системы часто слишком сложны, чтобы быть представленными одной моделью. Признавая это, авторы предлагают диадную модель ЖЦ, позволяющую описывать ЖЦ с разных точек зрения. В этом контексте состав модели ЖЦ стал составным, поскольку сочетание разных оценок неизбежно. Авторами определен подход для формирования частей ЖЦ ПК и метод композиции частей в составную модель ЖЦ ПК.

2.2. Модели жизненного цикла

Модель жизненного цикла применяют в разных направлениях: биологии, экономике, моделировании, вычислениях, информатике, проектировании, строительстве и т.д. Жизненный цикл является интегральной характеристикой вычислительных систем. Жизненный цикл отражает последовательность временных периодов, на каждом из которых объект имеет разную эффективность и по-разному проявляет себя по отношению к внешнему окружению. В первую очередь это относится к эффективности эксплуатации объекта. Связь эффективности функционирования объекта с этапами жизненного цикла определяет важность поддержки этапов жизненного цикла, на которых объект обеспечивает максимальную эффективность.

Поддержка жизненного цикла может быть внешней и внутренней. Внутренняя поддержка ЖЦ направлена на функционирование системы и исключение противоречий между ее компонентами. Одним из методов внутренней поддержки является обеспечение условий комплементарности [7] между программными компонентами, решающими общую задачу. Внешняя поддержка ЖЦ направлена на отражение внешних угроз среды или конкурентов.

Необходимо различать жизненные циклы разных систем и объектов. Продукция имеет иную модель ЖЦ по сравнению с проектом [8]. Проект имеет иной ЖЦ по сравнению со сложной технической системой. ЖЦ ИС [9] отличается от жизненного цикла технической системы. Анализ эффективного функционирования системы связан с исследованием таких важных параметров, как критерий длительности функционирования программ, причинно-следственная связь функционирования программы с внешними факторами, модели проектирования ЖЦ.

Рассмотрим несколько типовых моделей жизненного цикла. Распространенная модель ЖЦ – трапециевидная модель, включающая четыре этапа или четыре фазы. Более сложной является модель, известная как «петля качества» (ISO 9001), которая включает 11 фаз. Эти модели являются статическими. При проектировании и в динамических системах существуют каскадная и спиральная модели ЖЦ. Каскадная модель опирается на дополнительные внешние ресурсы, которые увеличивают длительность жизненного цикла. Спиральная – основана на выработке дополнительного ресурса в процессе функционирования системы или объекта. Эти модели ЖЦ являются динамическими. Фактически они выполняют функции реинтеграции и регенерации [10]. В данной работе предлагается ресурсная модель жизненного цикла [5], основанная на том, что объем ресурсов и скорость их расходования определяет жизненный цикл системы или объекта.

Уравнение, описывающее процесс изменения состояния системы за счет расхода ресурсов, называется логистическим уравнением. Логистическое уравнение, также известное как уравнение Ферхюльста (Pitrre Francois Verhulst), изначально было получено при рассмотрении модели роста [11]. Обозначим через P состояние системы в зависимости от времени t. Такая модель сводится к дифференциальному уравнению:

$$\delta\frac{dP}{dt} = rP\left(1-\frac{P}{K}\right). \quad (1)$$

Логистическое уравнение допускает разные трактовки. Применительно к цели данного исследования оно рассматривается с позиций расхода и потребления ресурсов. В выражении (1) параметр r характеризует скорость расхода ресурсов, а параметр K – максимально возможную ёмкость ресурсов системы. Показатель δ является индикатором процесса. Большинство записей данного уравнения не рассматривают показатель δ. В нашем исследовании этот показатель является индикатором, который имеет два оппозиционных целочисленных значения +1 и –1. Значение величины δ отражает два качественно разных процесса, но протекающих по одинаковой аналитической зависимости. При равенстве единице его опускают. Положительное значение индикатора δ соответствует развитию системы (ПК) за счет потребления ресурса и накопления собственного ресурса. Его отрицательное значение соответствует деградации системы за счет расхода собственного ресурса и потерю ресурса системой. Точным решением уравнения (1) является логистическая функция – *S*-образная кривая (логистическая кривая), для которой существует предел:

$$\lim_{t\to\infty} P(t) = K.$$

Можно упростить решение логистического уравнения до вида, которое используют в однопараметрической модели Раша:

$$P(t) = \frac{KP_0e^{rt}}{K + P_0(e^{rt}-1)}, \quad (2)$$

$$P(t) = \frac{P_0e^{t-a}}{1+P_0e^{t-a}}.$$

Соответствующая кривая приведена на рис. 1, она известна в математике и относится к классу сигмоид (sigmoid). Сигмоида – гладкая монотонная *S*-образная возрастающая функция, которая применяется для отражения процесса накопления и предела процесса. Величина a задает сдвиг вправо от начала координат. Для рис. 1 величина a соответствует середине сигмоиды. Сигмоида *S*1, как модель жизненного цикла, показывает рождение и рост. Насыщение или зрелость системы отражает горизонтальный отрезок P_{con}, который дополняет сигмоиду.

На рис. 1 показаны три начальные фазы жизненного цикла: рождение, рост, зрелость. Этот график соответствует δ = 1. Приведенная модель называется моделью роста. Если начинается деградация системы, то индикатор δ принимает значение –1. Это эквивалент-

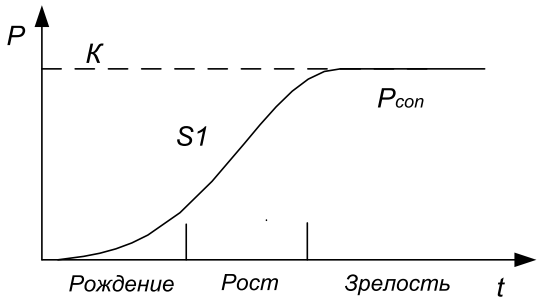

Рис. 1. Три фазы жизненного цикла при δ = 1.

но тому, что аргумент *t* изменит знак, и величина *a* заменится на величину *b*, которая задает сдвиг влево. В результате для процесса деградации получаем уравнение:

$$P(t) = \frac{P_0 e^{-t+b}}{1 + P_0 e^{-t+b}}. \quad (3)$$

В выражениях (2) и (3) скорости расхода ресурсов *r* условно равны 1. Это упрощение делается для того, чтобы понять суть процесса роста и деградации. Графическая зависимость, соответствующая уравнению (3), имеет вид, приведенный на рис. 2.

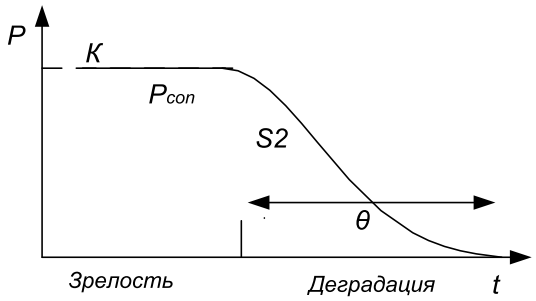

Рис. 2. Две фазы жизненного цикла при δ = –1.

Здесь приведена другая сигмоида *S*2. Она отличатся от *S*1 наклоном и деталями. Рис. 1 показывает три фазы жизненного цикла. Рис. 2 показывает две фазы жизненного цикла. Фаза «зрелость» является общей на обоих рисунках, что дает основание рассматривать данную модель жизненного цикла (рис. 1, 2) как составную трапециевидную модель, имеющую четыре фазы. Раздельный показ жизненного цикла на рис. 1 и рис. 2 свидетельствует о том, что рост и рождение объекта или системы не зависят от ее возможной последующей диссипации или деградации. И наоборот, деградация системы не зависит от предыдущих фаз рождения и роста.

Модель, приведенную на рис. 2, называют моделью деградации. Она включает часть процесса зрелости (P_{con}) и процесс деградации *S*2. Процесс *S*2 может быть также процессом диссипации. Время деградации (диссипации) обозначено символом θ. Суммарный жизненный цикл программных компонент ЖЦ ПК определится как:

$$\text{ЖЦ ПК} = S1 + P_{con} + S2. \quad (4)$$

Наступление деградации или диссипации часто происходит под воздействием внешних деструктивных сил (рис. 3). В нормальном состоянии на вход комплекса программных компонент поступает входная информация, которая в ходе вычислений преобразуется в выходную.

Рис. 3. Модель воздействия деструктивных сил
на комплекс программных компонент.

Деструктивное воздействие приводит к расходу ресурсов комплекса ПК и создает процесс, который изображен кривой $S2$ на рис. 2. Для того чтобы поддержать жизненный цикл ПК, необходимо предотвратить деградацию $S2$. Для этого вводят дополнительный внешний ресурс $S1^*$ из резерва, соответствующий условию $S1^* = -S2$. В результате новый ресурс нейтрализует деградацию $S2$ и период зрелости возрастет:

$$P^*_{con} = P_{con} + \theta. \tag{5}$$

Деградация происходит за счет внутренних отказов или внешних воздействий. При появлении новых причин, вызывающих деградацию $S2^*_i$, необходимо использование новых ресурсов $S1^*_{i+1}$. Рекурсивное использование выражения (5) позволяет увеличивать жизненный цикл ПК, пока имеются ресурсы подавления деградации или диссипации.

Также существуют и принципиальные математические подходы для системы типовых моделей программных компонент жизненного цикла (рис. 4).

Информация или документы в виде информационного сообщения проверяются при поступлении на соответствие предметной области.

Информационный массив образует множество проблемно-ориентированных документов, каждый из которых в системе представляется в виде поискового образа.

Документы классифицируются в программных компонентах в соответствии с информационным профилем документа.

Структурирование и классификация информации, в том числе деление на крупные и мелкие части в соответствии с критериями, происходит на последующих этапах.

В случае необходимости сбора дополнительной информации к первичному информационному массиву могут добавляться дополнительные информационные сообщения, и процедура повторяется.

2.3. Составная модель ЖЦ ПК

Как уже отмечалось [6], для сложных объектов или процессов невозможно описание с помощью одной модели, что вынуждает использовать две и более модели. Жизненный цикл может быть фиксированным и адаптивным. Адаптивный или меняющийся ЖЦ це-

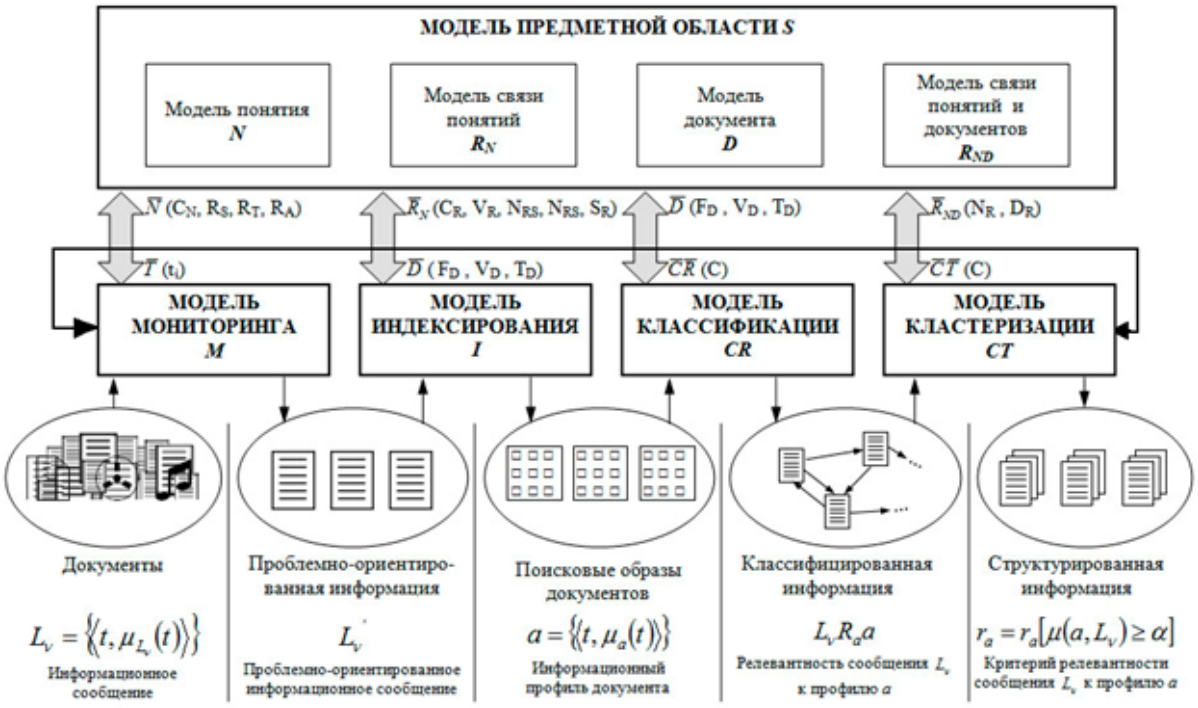

Рис. 4. Математическая модель жизненного цикла программных компонент.

лесообразно описывать набором моделей, каждая из которых характеризует определенный процесс и зависит от разных факторов. Составная модель жизненного цикла, состоящая из двух частных моделей, приведена на рис. 5. Эти две модели характеризуют рост и деградацию, и соответственно характеризуют модель роста и модель деградации ЖЦ.

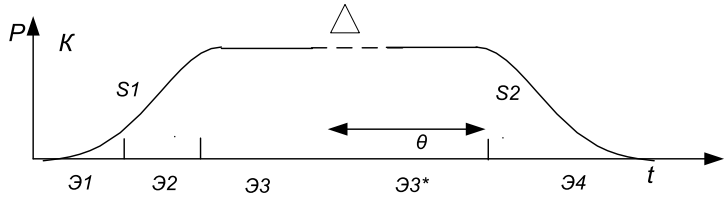

Рис. 5. Составная модель жизненного цикла программных компонент.

На рис. 5 условный разрыв между моделями роста и деградации показан пунктиром и обозначен символом Δ. Символом θ обозначено увеличение жизненного цикла программных компонент, обусловленное применением внешних информационных ресурсов (5). В общем случае величина Δ будет зависеть от количества актов информационного взаимодействия «помеха-ресурс»:

$$\Delta = P_{con} + \theta_1 + \theta_2 + \theta_3 + \ldots + \theta_n. \quad (6)$$

Выражение (6) показывает, что увеличение ЖЦ ПК возможно за счет подключения внешнего ресурса, подавляющего внешнее деструктивное воздействие. Условно обозначено *n* таких взаимодействий. Такими воздействиями могут быть, например, атаки вирусов.

При информационном взаимодействии может быть две ситуации – запаздывание и синхронность. Следует отметить важность информационной модели, называемой ин-

формационной ситуацией [12]. Информационная ситуация описывает модель объекта и все наиболее важные факторы, которые на него воздействуют. На рис. 6 показан результат информационного взаимодействия внешнего ресурса ($S1^*$) и деструктивного воздействия ($S2$).

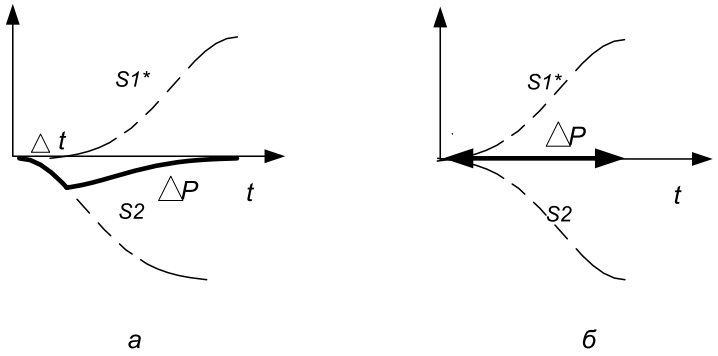

Рис. 6. Использование ресурса для ликвидации деструктивного воздействия на ЖЦ ПК: *а* – асинхронное; *б* – синхронное.

На рис. 6 показаны асинхронное (рис. 6а) и синхронное (рис. 6б) информационное взаимодействие между ресурсом и деструктивным воздействием. Результат взаимодействия обозначен как ΔP. Значение ΔP показано жирной линией. Для асинхронного взаимодействия (рис. 6а) деструктивная сила начинает действовать раньше, и между $S2$ и $S1^*$ существует период задержки Δt. В этом случае ресурс или резерв не успевают вовремя погасить деструктивную силу, и результат взаимодействия ΔP в течение некоторого времени отличен от нуля, он показан ломаной линией. График ΔP соответственно меняет состояние P.

Для синхронного взаимодействия (рис. 6б) деструктивная сила сразу встречает противодействие. В этой ситуации между $S2$ и $S1^*$ не существует задержки Δt, и они фактически симметричны. Такая ситуация имеет место при атаке компьютерного вируса на систему, в которой установлена антивирусная защита. В этом случае ресурс сразу погашает деструктивную силу и результат взаимодействия ΔP равен нулю, он показан прямой линией с нулевым значением.

2.4. Организация программных компонент как метод резервирования

В работе [13] обоснована актуальность использования проблемно-ориентированной конфигурации вычислительных комплексов для автоматизированных систем обработки информации и управления. На практике программные компоненты включают группы алгоритмов или группы задач. Каждый компонент решает несколько возможных задач. Системный термин «программный компонент» можно заменить эквивалентным термином из области вычислений – «специализированный вычислительный комплекс (СВК)». На каждом ПК или СВК находится как минимум один алгоритм. Если количество алгоритмов на СВК больше одного, то они объединяются в резервированную внутреннюю сеть. Объединение алгоритмов в сеть создает ресурс жизненного цикла ПК. Алгоритмы координируют свои действия между собой по сети. Такая модель ПК допускает возникновение отказов, заключающихся в потере связи между алгоритмами и полном или частичном выходе из строя алгоритмов.

Программные компоненты образуют КПК при решении сложной задачи. КПК, как правило, обладает синергетическим эффектом и это – его преимущество по отношению к отдельным компонентам. Программный комплекс имеет свой жизненный цикл. В комплексе ПК часто функционируют по принципу двойной алгоритмической модели. Примером являются роевые алгоритмы или алгоритмы муравьиной колонии. Главная задача такого алгоритма – достижение групповой цели, вторая задача — взаимодействие между собой для повышения надежности и увеличения жизненного цикла всего комплекса. При сетевой организации КПК отказы в одном ПК (СВК) перераспределяют его задачи на весь комплекс. Сетевое взаимодействие выполняет роль ресурса. Такой ресурс возможен при введении механизма реконфигурации вычислительных комплексов [14], учитывающего особенности программных компонент.

Обращает на себя внимание то, что триада объектов является минимальной моделью элемента сети ПК с резервированием. Наличие трех связей обеспечивает взаимодействие между тремя компонентами, если одна из связей прервется. В комплексе все ПК связаны между собой с помощью сети по схеме «каждая с каждой», в результате чего при выходе из строя одной связи между СВК за счет дополнительных соединений связь в системе будет сохраняться, и комплекс будет функционировать; при выходе из строя одной СВК ее функции будут перераспределяться между оставшимися СВК, и комплекс также будет функционировать. Три (и более) связанные ПК создают резерв для каждой из систем комплекса. Резерв по внешней связи создает резервная сеть. Основная идея резервирования – создание треугольников связанных систем. Линию можно прервать, а сеть прервать сложнее. Пример сети Интернет показывает это свойство.

Технологическая модель или метод поддержки ЖЦ комплексов ПК разделен на следующие части: резервирование физических ресурсов (вычислительных машин) путем объединения их в сеть; резервирование технологических ресурсов (программных компонент) путем объединения их в сеть; резервирование коммуникационных ресурсов (каналов связи) на физическом уровне путем создания в сети схем мультиграфа; введение правил перераспределения ресурсов при выходе ресурса из строя; поддержка жизненного цикла за счет дополнительных ресурсов.

Процессуальная модель поддержки ЖЦ ПК заключается в следующем. Гетерогенный комплекс ПК (КПК) может находиться во множестве состояний **Z**, каждое из которых может быть представлено процессами роста $S1_i$ (подключение ресурса) и деградации $S2_i$. Каждое состояние Z_i зависит от взаимодействия ресурса и процесса диссипации (рис. 5). В ходе функционирования комплекс ПК переходит от одного состояния к другому S_i. Переход между состояниями вычислительного комплекса ПК происходит как под воздействием внешних факторов, так и под воздействием внутренних событий. Предлагаемый метод адаптивного реагирования на процессы диссипации предполагает создание реактивной системы отклика на воздействия, которая инициирует подключение ресурса для нейтрализации очередной диссипации.

Сущность метода поддержки ЖЦ КПК заключается в создании резервированной архитектуры комплекса и использовании индикационного контроллера, реагирующего на внешнее воздействие или внутреннюю потерю комплементарности действий системы. Метод позволяет реинтегрировать комплекс ПК после отказа одного из компонентов и продолжить исполнение функционального процесса.

Общий алгоритм поддержки или реинтеграции комплексов ПК в случае отказа состоит из следующих шагов: индикация отказа, оценка последствий отказа, перераспределение задач на другие ПК, подавление помехи за счет внутренних ресурсов, исключение полностью отказавшего устройства из сети КПК, исключение связей с полностью отказавшим устройством, попытка восстановления отказавшего ПК. Предложенные методы отличаются от существующих тем, что учитывают особенности формирования жизненного цикла КПК за счет учета потребления и расхода ресурсов.

3. Обсуждение результатов

Оценка методов разработки программного обеспечения (SEE) является важной деятельностью в этой области. В работе [3] проведен анализ на основе подборки из 1178 публикаций с целью выявления явных и неявных тенденций в разработке и развитии программного обеспечения и программных компонент. Несмотря на такой большой объем исследований, в нем не нашлось места для анализа влияния ресурсов на жизненный цикл ПК и КПК. Предлагаемая работа восполняет этот пробел.

В модели деградации *S*2 следует различать диссипацию [15] и деградацию [16] ПК, хотя результаты от действия этих процессов одинаковые. Деградация означает нарушение внутренней согласованности и комплементарности компонентов ПК. Диссипация означает отклонение состояния ПК в сторону от целевого при внутренней согласованности компонентов системы. В первом случае ресурсы направляются на самовосстановление, потому что оно требуется всегда. Во втором случае они направляются на отражение угроз и самовосстановление, если оно требуется.

Существует ряд ограничений применения данной технологии. Основное ограничение заключается в том, что каждый КПК должен иметь не менее трех СВК. Комплекс ПК должен иметь сетевую, а не линейную структуру. Базовым звеном технологии должна быть триада. С одной стороны, это обеспечивает резервирование, но с другой стороны, в системе появляются паразитные обратные связи. Эти связи уменьшают внутреннюю устойчивость работы КПК с большим числом СВК. Рост числа СВК влечет рост сложности системы в геометрической прогрессии, а также рост сложности проблемы синхронизации. Введение индикативного показателя в логистическое уравнение продиктовано необходимостью отражения характера двух противоположных процессов. В большинстве работ эти процессы рассматривают независимо. Введение индикативного показателя позволяет, при необходимости, сравнивать эти процессы и даже объединять для получения суммарного эффекта. Модель формирования жизненного цикла за счет логистической кривой удобна, но математически является разрывной, что следует расценить как недостаточно гибкое аналитическое решение.

Выводы

Широкое применение программных компонент в прикладных областях способствует формированию нового направления программной инженерии. Модели программных компонент широко используют при решении динамических задач, например, при управлении транспортом. Это привело к созданию специализированных транспортных киберфизических систем, которые приходят на смену интеллектуальным транспортным

системам. Основой киберфизических систем являются программные компоненты. Для увеличения жизненного цикла программных компонент и, главное, комплекса программных компонент, необходимо проводить резервирование технологических и физических ресурсов с использованием сетевых моделей, включая модели мультиграфа. Для обеспечения устойчивой и надежной работы отдельных ПК и комплекса ПК должны использоваться модели триад [17]. Для моделирования жизненного цикла целесообразно использовать логистические уравнения. Для моделирования жизненного цикла с помощью логистических уравнений целесообразно использовать две модели описания жизненного цикла. Первая часть модели ЖЦ – это модель роста. Вторая часть модели ЖЦ – это модель деградации. Предложен рекурсивный механизм поддержки ЖЦ ПК, основанный на противодействии внешним деструктивным воздействиям дополнительных ресурсов. Рассмотрены синхронное и асинхронное противодействие деструктивным воздействиям.

Еще одним фактором поддержки и увеличения жизненного цикла служит механизм саморазвития. Это является предметом дальнейших исследований.

Литература:

1. Tsvetkov V.Ya. Information Constructions. *European Journal of Technology and Design.* 2014;5(3):147-152. http://dx.doi.org/10.13187/ejtd.2014.5.147

2. Sajjad M., Sajjd A., Niazi M., Alshayeb M., Richardson I. Key factors that influence task allocation in global software development. *Inf. Soft. Technol.* 2017;91:102-122. https://doi.org/10.1016/j.infsof.2017.06.009

3. Sehra S.K., Brar Y.S., Kaur N., Sehra S.S. Research patterns and trends in software effort estimation. *Inf. Soft. Technol.* 2017; 91:1-21. https://doi.org/10.1016/j.infsof.2017.06.002

4. Tiwari S., Gupta A. Investigating comprehension and learnability aspects of use cases for software specification problems. *Inf. Soft. Technol.* 2017;91:22-43. https://doi.org/10.1016/j.infsof.2017.06.003

5. Tsvetkov V.Ya. Resource Method of Information System Life Cycle Estimation. *European Journal of Technology and Design.* 2014;2(4):86-91. https://doi.org/10.13187/ejtd.2014.5.147

6. Laghouaouta Y., Anwar A., Nassar M., Coulette B. A dedicated approach for model composition traceability. *Inf. Soft. Technol.* 2017;91:142-159. https://doi.org/10.1016/j.infsof.2017.07.002

7. Цветков В.Я. Комплементарность информационных ресурсов. *Международный журнал прикладных и фундаментальных исследований.* 2016;2:182-185. URL: https://applied-research.ru/ru/article/view?id=8546

8. Муравьева Я.И. Жизненный цикл проекта. *Экономика и социум*. 2016;3(22):1889-1894.

9. Дик В.В., Шайтура С.В. Жизненный цикл информационных систем. *Вестник МГТУ МИРЭА (Российский технологический журнал)*. 2014;3(4):116-129.

10. Матчин В.Т. Регенерация бортовых баз данных. *Наука и технологии железных дорог.* 2019;4(12):20-29.

11. Verhulst P.F. Notice sur la loi que la population poursuit dans son accroissement. *Corresp. Math. Phys.* 1838;10:113-126.

12. Markelov V.M. Situational Modeling in Logistics. *European Journal of Economic Studies.* 2013;6(4):204-209.

13. Сорокин С.А., Бененсон М.З., Сорокин А.П. Методики оценки производительности гетерогенных вычислительных систем. *Российский технологический журнал.* 2017;5(6):11-19. https://doi.org/10.32362/2500-316X-2017-5-6-11-19

14. Riakiotakis I., Ciorba F.M., Andronikos T., Papakonstantionou G. Distributed dynamic load balancing for pipelined computations on heterogeneous systems. *Parallel Computing.* 2011;37(10-11):713-729. https://doi.org/10.1016/j.parco.2011.01.003

15. Kawai R., Parrondo J.M.R., Van den Broeck C. Dissipation: the phase-space perspective. *Phys. Rev. Lett.* 2007;98(8):080602. https://doi.org/10.1103/PhysRevLett.98.080602

16. Singh B., Sharma N. Mechanistic implications of plastic degradation. *Polym. Degrad. Stabil.* 2008;93(3):561-584. https://doi.org/10.1016/j.polymdegradstab.2007.11.008

17. Кудж С.А., Цветков В.Я. Тринитарные системы. *Российский технологический журнал.* 2019;7(6):74-88. https://doi.org/10.32362/2500-316X-2019-7-6-151-167

References:

1. Tsvetkov V.Ya. Information Constructions. *European Journal of Technology and Design.* 2014;5(3):147-152. http://dx.doi.org/10.13187/ejtd.2014.5.147

2. Sajjad M., Sajjd A., Niazi M., Alshayeb M., Richardson I. Key factors that influence task allocation in global software development. *Inf. Soft. Technol.* 2017;91:102-122. https://doi.org/10.1016/j.infsof.2017.06.009

3. Sehra S.K., Brar Y.S., Kaur N., Sehra S.S. Research patterns and trends in software effort estimation. I*nf. Soft. Technol.* 2017; 91:1-21. https://doi.org/10.1016/j.infsof.2017.06.002

4. Tiwari S., Gupta A. Investigating comprehension and learnability aspects of use cases for software specification problems. *Inf. Soft. Technol.* 2017;91:22-43. https://doi.org/10.1016/j.infsof.2017.06.003

5. Tsvetkov V.Ya. Resource Method of Information System Life Cycle Estimation. *European Journal of Technology and Design.* 2014;2(4):86-91. https://doi.org/10.13187/ejtd.2014.5.147

6. Laghouaouta Y., Anwar A., Nassar M., Coulette B. A dedicated approach for model composition traceability. *Inf. Soft. Technol.* 2017;91:142-159. https://doi.org/10.1016/j.infsof.2017.07.002

7. Tsvetkov V.Ya. Complementarity of information resources. *Mezhdunarodnyi zhurnal prikladnykh i fundamental'nykh issledovanii = International Journal of Applied and Fundamental Research.* 2016;2:182-185 (in Russ.). URL: https://applied-research.ru/ru/article/view?id=8546

8. Murav'eva Ya.I. Project life cycle. *Ekonomika i sotsium = Economy and society*. 2016;3(22):22-25 1888-1893 (in Russ.).

9. Dick V.V., Shaytura S.V. Life cycle of information systems. *Vestnik MGTU MIREA (Rossiiskii tekhnologicheskii zhurnal) = Russian Technological Journal.* 2014;3(4):116-129 (in Russ.).

10. Matchin V.T. Onboard database Regeneration. *Nauka i tekhnologii zheleznykh dorog = Science and technology of railways*. 2019;4(12):20-29 (in Russ.).

11. Verhulst P.F. Notice sur la loi que la population poursuit dans son accroissement. *Corresp. Math. Phys.* 1838;10:113-126.

12. Markelov V.M. Situational Modeling in Logistics. *European Journal of Economic Studies.* 2013;6(4):204-209.

13. Sorokin S.A., Benenson M.Z., Sorokin A.P. Methods for evaluating the performance of heterogeneous computer systems. *Rossiiskii tekhnologicheskii zhurnal = Russian Technological Journal.* 2017;5(6):11-19 (in Russ.). https://doi.org/10.32362/2500-316X-2017-5-6-11-19

14. Riakiotakis I., Ciorba F.M., Andronikos T., Papakonstantionou G. Distributed dynamic load balancing for pipelined computations on heterogeneous systems. *Parallel Computing.* 2011;37(10-11):713-729. https://doi.org/10.1016/j.parco.2011.01.003

15. Kawai R., Parrondo J.M.R., Van den Broeck C. Dissipation: the phase-space perspective. *Phys. Rev. Lett.* 2007;98(8):080602. https://doi.org/10.1103/PhysRevLett.98.080602

16. Singh B., Sharma N. Mechanistic implications of plastic degradation. *Polym. Degrad. Stabil.* 2008;93(3):561-584. https://doi.org/10.1016/j.polymdegradstab.2007.11.008

17. Kudzh S.A., Tsvetkov V.Y. Trinitarian systems. *Rossiiskii tekhnologicheskii zhurnal = Russian Technological Journal.* 2019;7(6):151-167 (in Russ.). https://doi.org/10.32362/2500-316X-2019-7-6-151-167

Об авторах:

Кудж Станислав Алексеевич, доктор технических наук, профессор, ректор ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). Scopus Author ID 56521711400, http://orcid.org/0000-0003-1407-2788

Цветков Виктор Яковлевич, доктор технических наук, доктор экономических наук, профессор, советник ректората ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). Scopus Author ID 56069916700

Рогов Игорь Евгеньевич, директор Института довузовской подготовки ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78).

About the authors:

Stanislav A. Kudzh, Dr. Sci. (Engineering), Professor, Rector of the MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia). Scopus Author ID 56521711400, http://orcid.org/0000-0003-1407-2788

Viktor Ya. Tsvetkov, Dr. Sci. (Engineering), Dr. Sci. (Economics), Professor, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia). Scopus Author ID 56069916700

Igor E. Rogov, Director of the Institute of Pre-University Training, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia).

Поступила: 18.12.2019; получена после доработки: 13.07.2020; принята к опубликованию: 25.07.2020.