УДК 334.71:656:338.245

ПРИМЕНЕНИЕ ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ ДЛЯ РЕГЕНЕРАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Матчин Василий Тимофеевич,
ст. преподаватель кафедры инструментального и прикладного программного обеспечения,
e-mail: matchin.v@gmail.com,
Институт информационных технологий,
Российский технологический университет (РТУ МИРЭА), г. Москва

В статье рассматривается задача регенерации программного обеспечения информационных систем. Показано различие между обновлением и регенерацией, рефакторингом и регенерацией программного обеспечения. Дана систематизация эволюционных алгоритмов. Детально рассмотрены особенности генетических алгоритмов и алгоритмов роя пчел. Описаны условия применения алгоритма роя пчел для регенерации программного обеспечения. Рассмотрены алгоритмы использования ресурсов на основе логистического уравнения. Описаны особенности применения логистического уравнения как индикатора регенерации программного обеспечения. Выделены два варианта регенерации: в информационных системах и в программном обеспечении. Сформулированы условия регенерации для этих вариантов на системном уровне. Регенерация в информационных системах применима при условии тринитарной связи программного обеспечения, технологического обеспечения и системы данных. Регенерация программного обеспечения применима при условии, что совокупность программ и алгоритмов образует связанную целостную систему. Технология регенерации представляется современной технологией модернизации программного комплекса.

Ключевые слова: программное обеспечение, кибернетика, вычисление, обновление информации, регенерация программных компонент

APPLICATION OF EVOLUTIONARY MODELING FOR SOFTWARE REGENERATION

Matchin V.T.,
senior lecturer department of instrumental and applied software,
e-mail: matchin.v@gmail.com,
Institute of Information Technology, Russian Technological University (RTU MIREA), Moscow

The article deals with the problem regeneration of information systems software. The difference between updating and regeneration, refactoring and software regeneration is shown. The systematization of evolutionary algorithms is given. The features of genetic algorithms and algorithms for swarming bees are considered in detail. The conditions for using the swarm of bees algorithm for software regeneration are described. Algorithms for using resources based on the logistic equation are considered. The features of using the logistic equation as an indicator of software regeneration are described. Two variants of regeneration are identified: in information systems and in software. Regeneration conditions are formulated for these variants at the system level. Regeneration in information systems is applicable if the software, technology, and data system are connected in a Trinitarian way. Software regeneration is applicable provided that the combination of programs and algorithms forms a related complete system. The regeneration technology is represented as a modern technology for upgrading the software system.

Keywords: software, cybernetics, computing, updating information, regenerating software components

DOI 10.21777/2500-2112-2019-4-42-52

Введение

Регенерация программного обеспечения – новый процесс, который обусловлен взаимосвязью программных компонент технологии и данных в информационных системах. Технология регенерации возникла как ответ на требование реальности. Длительное время в информационных системах проводилось независимое обновление программного обеспечения и модернизация технологического обеспечения. Данные подстраивались под программное обеспечение. В то же время для любых информационных систем (ИС) важным фактором является качество и оценка качества такой системы [3]. ИС характеризуется следующими компонентами: применяемыми данными, технологией функционирования (включая обработку информации), программными компонентами. Для информационных систем необходимо применять методы стандартизации программного обеспечения (ПО) [9] и учитывать соответствующие ГОСТы. Перечисленные компоненты характеризуют качество функционирования системы. При обновлении компонент может изменяться качество функционирования. Основой оценки качества программного обеспечения в информационных системах являются два стандарта: отечественный стандарт ГОСТ Р ИСО/МЭК 25010-2015 «Информационные технологии (ИТ). Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов»; зарубежный стандарт ISO/IEC 25010:2011 «Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программного обеспечения» [14]. Указанные стандарты отражают современную концепцию развития ПО на основе системной и программной инженерии[1] [1]. Такой интегрированный подход требует проводить разработку информационных систем и программного обеспечения к ним с позиций качества. В соответствии со стандартом ГОСТ Р ИСО/МЭК 25010-2015 в системах выделяют три модели качества: модель самого алгоритма и модель применения алгоритма, модель качества данных. Поэтому разработка любой системы связана с разработкой программного обеспечения и с разработкой качественной модели данных. Появление и применение этих стандартов зафиксировало важное обстоятельство. Программное обеспечение, технологическое обеспечение (ТО) и структуры данных (СД) взаимосвязаны. Их независимое обновление нарушает комплементарность [7; 10] между ними. Регенерация – это процесс взаимосвязанного обновления ПО, ТО, СД. Для регенерации пригодны различные методы. Одним из перспективных следует считать совокупность эволюционных методов.

1. Переносимость данных при регенерации программного обеспечения информационной системы

Переносимость данных является одной из основных задач при регенерации программного обеспечения ИС. Затраты и сложность переноса данных в новую среду зависят от характеристик базы данных, которые отражают форматную, лингвистическую и физическую совместимость переносимой базы данных и рассматриваемых платформ[2]. Форматная совместимость характеризуется степенью соответствия данных в базе данных анализируемых платформ требованиям стандартов на форматы представления данных. Лингвистическая совместимость определяется степенью использования в рассмаПрименение логистического уравнения для исследования системыванных соответствующими стандартами. Физическая совместимость заключается в степени соответствия кодировки информации баз данных одинаковым стандартам.

Одной из задач регенерации программного обеспечения ИС является обеспечение переносимости данных, хранящихся во внешней памяти, на новые или модифицированные прикладные платформы. Наилучшим образом это достигается применением типовых моделей данных и стандартов, строго регламентирующих форматы и способы представления данных. Однако широкая и полная стандартиза-

[1] *Дешко И.П., Кряженков К.Г., Цветков В.Я.* Системная и программная инженерия: учебное пособие. – М.: МАКС Пресс, 2018. – 80 с.

[2] *Липаев В.В.* Сопровождение и управление конфигурацией сложных программных средств. – М.: СИНТЕГ, 2006. – 372 с.

ция, как правило, не достигается, поэтому используется локальная унификация номенклатуры и функций, применяемых для обработки определенных классов данных.

Интероперабельность ПО обеспечивает возможность обмена данными между модулями, в том числе, реализуемыми на разнородных прикладных платформах, а также возможность совместного использования ими обмениваемых данных. Данное свойство обеспечивается построением стандартизованных коммуникационных интерфейсов.

2. Методы эволюционного моделирования

Методы эволюционного моделирования направлены на решение оптимизационных задач большой размерности через процесс комбинирования случайных и детерминированных факторов подобно тому, как это происходит в живой природе в процессе эволюции биологических видов, адаптации колоний живых организмов к условиям обитания, выборе стратегии группового поведения животных и т.п.

Взаимосвязь ПО, ТО, СД в информационных системах представляет собой совокупность случайных и детерминированных факторов, похожую на связь факторов в процессе эволюции биологических видов. Эволюционное компьютерное моделирование применяется в задачах оптимизации программного обеспечения [13], генерации оптимальных тестовых наборов данных [12], оценке надежности программного обеспечения [11], кластеризации программного обеспечения на компоненты [16], выделении функциональности в виде программной и аппаратной компонент [19].

Следует отметить процедуру, направленную на улучшение программного обеспечения которую называют рефакторинг (software refactoring). Отдаленно она отражает идеи регенерации. Рефакторинг является довольно узкой операцией. Он состоит в улучшении внутренней структуры исходного кода программы при сохранении ее внешнего поведения. Рефакторинг не осуществляет: а) переписывание кода; б) исправление ошибок; в) улучшение интерфейса. Регенерация допускает все три пункта – а, б, в, – то есть является более широкой технологией, поддерживающей идеи рефакторинга. При рефакторинге программного обеспечения применяется эволюционное моделирование [17], что делает его удобным инструментом при регенерации ПО.

Использование эволюционного моделирования для композиции сервисов ПО с целью обеспечения качества обслуживания позволяет сократить перебор стратегий реагирования на отклонения от заданного уровня качества. Использование эволюционного моделирования при управлении потоком сервисного обслуживания позволяет поддерживать гибкий поток работ. В этих методах композиция технологий сервисов осуществляется динамически. Применение эволюционных алгоритмов на этапе проектирования информационной системы для эффективной регенерации компонент позволяет исследовать значительно большие подмножества компонент, которые могут явно казаться не связанными, но в ходе анализа выявляется их связь и комплементарность отношений [8].

3. Систематизация эволюционного моделирования

Эволюционное моделирование связано с методами моделирования и алгоритмами, которые эти методы реализуют. Общим является изменение методов и алгоритмов, которое включает саморазвитие и адаптацию. Общей характеристикой этих методов и алгоритмов можно определить развитие и саморазвитие алгоритмов. Общей характеристикой методов и алгоритмов эволюционного моделирования является стремление к оптимизации, а в отдельных случаях, решение этой задачи.

Выделяют три группы алгоритмов эволюционного моделирования [5]:

− узко эволюционные,

− синергетические;

− системные.

Указанные три группы объединяют эволюционные программирование и стратегия. Следует отметить, что в современном понимании алгоритм рассматривают не только как последовательность вычислений, но и как метод развития, как сложную систему и как характеристику закономерностей развития в информационном поле.

Эволюционное программирование использует адаптивные на воздействие среды универсальные конечные автоматы [6]. Эти конечные автоматы позволяют решать оптимизационные задачи. Эволюционное моделирование в области компьютерных программ может быть представлено моделью, в которой каждая программа является гипотезой A о форме зависимости целевой функции P от входных переменных x, то есть

$$\forall P(x) \rightarrow A.$$

Как только получена функция (программа) с заданной точностью, выражающая искомую зависимость, в нее алгоритмически начинаются вноситься небольшие изменения, повышающие точность решения. Одновременно могут развиваться несколько параллельных линий эволюции программ, конкурирующих между собой. Конкурирующим критерием является точность идентификации решения или уменьшение погрешности итерации. Такой подход позволяет использовать эволюционное программирование в области нейроэволюции – автоматической регенерации весовых коэффициентов и топологии искусственных нейронных сетей. *Эволюционная стратегия* основана на модификации решений или паттернов, когда речь идет о программах. Достаточно подробно эта стратегия представлена в диссертации автора Н.Б. Уральского [6].

Общим для трех групп алгоритмов эволюционного моделирования является стратегия, а тактика меняется в зависимости от решаемой задачи или проблемы. Стратегии являются близкими по алгоритмам их реализации, а тактические действия совпадают по принципам, но по механизмам реализации различаются. Общим для трех групп являются решения задач локальной и глобальной оптимизации. При этом характерно изменение локальной оптимизации в пользу решения задачи глобальной оптимизации.

4. Генетические алгоритмы

Генетические алгоритмы (ГА) основаны на механизме модификации, больше схожем с регенерацией, чем с обновлением компонент[3]. Достаточно подробно и развернуто модель этого алгоритма представлена в учебном материале, который размещен на сайте Уфимского государственного авиационного технического университета [2]. Эту базовую модель используют многие авторы статей и диссертаций.

Применение ГА к решению оптимизационных задач неуклонно растет от десятилетия к десятилетию по мере развития средств вычислительной техники, позволяющих исполнять всё более сложные алгоритмы для решения задач возрастающей размерности за приемлемое время. Блок-схема простого генетического алгоритма, модифицированного под регенерацию программных компонент, приведена на рисунке 1.

В качестве базовой основы был использован алгоритм дискретного дерева для решения задачи оптимизации (Discrete Tree-Seed algorithm – DTSA) [12]. Этот алгоритм базировался на аналоговой версии (TSA). Основная идея алгоритма – это решение задачи непрерывной оптимизации отношений между деревьями и их семенами.

Многие двоичные версии TSA представлены в литературе, включая отдельную версию TSA, в которой переменные решения представлены в виде целочисленных значений. В работе [12] базовый TSA модифицирован путем интеграции операторов подкачки, сдвига и преобразования симметрии для решения задач оптимизации с перестановочной кодировкой, и он называется DTSA. В данной работе использованы идеи, но не сам алгоритм.

Если перенести идеи работы алгоритма на регенерацию компонент, то можно рассматривать компоненты как семена (тактика), а дерево как основной алгоритм (стратегия). Компоненты создаются с использованием формул (1) или (2) для каждого дерева путем итераций

$$S(k)=T(i)+ a\ (B\text{-}Tr)), \quad (1)$$

[3] *Гладков Л.А., Курейчик В.В., Курейчик В.М.* Генетические алгоритмы: учебное пособие. – 2-е изд. – М.: Физматлит, 2006. – 320 с.

$$S(k)=T(i)+ a\ (T(i) -Tr)). \quad (2)$$

В выражениях (1), (2) приняты следующие обозначения: $T(i)$ – i-е дерево; $S(k)$ – k-ый компонент $T(i)$; α – равномерно распределенное случайное число между -1 и 1; B – лучшее дерево, полученное до настоящего времени; $T(r)$ – случайное дерево, которое отличается от $T(i)$. Два уравнения создания начальных чисел (уравнения 1 или 2) управляются параметром Search Tendency (ST), имеющим значение от 0 до 1.

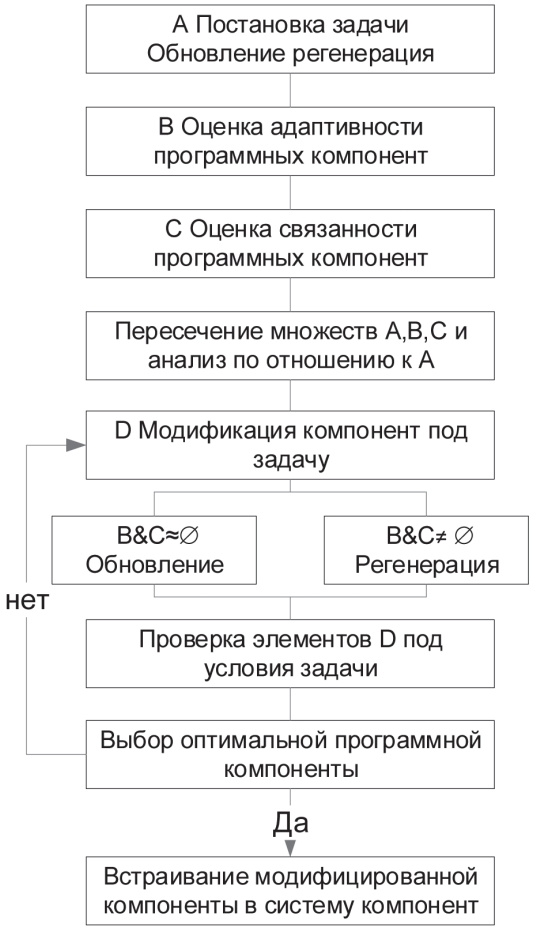

Рисунок 1 – Блок-схема простого модифицированного генетического алгоритма

В процессе поиска генерируется и сравнивается равномерно распределенное случайное число в диапазоне [0, 1] с параметром ST. Если ST меньше, чем это случайное число, уравнение (1) используется для создания семян, в противном случае уравнение (2) используется для создания семян. Уравнение (1) обеспечивает усиление, уравнение (2) обеспечивает диверсификацию в TSA. Блок-схема модифицированного генетического алгоритма приведена на рисунке 1. Схема на рисунке 1 характеризуется наличием множеств и соответственно возможностью применения теоретико-множественного подхода к ее описанию.

На первом этапе мотивацией применения алгоритмов является постановка задачи, обусловленная противоречием между требованиями реальности и возможностями программы. Это задает множество A – множество проблем и задач или проблемное множество. Для создания нового модифицированного компонента проводят анализ существующих компонент для выяснения, в какой степени возможно их использование или адаптация. Это задает множество B – множество адаптивных свойств или адаптивное множество. На следующем этапе, важном для регенерации, выявляют связанность и обусловленность компонент. Это задает множество C – множество отношений и связей. Множество D – является множеством процессов преобразований и модификации компонент как новых компонент.

Модификация программных компонент (ПК) может проходить путем обновления или модификации. Это выявляется в блоках сравнения множеств B и C. Если их пересечение дает пустое множество, то в этом случае целесообразно обновление.

На следующем этапе модифицированные компоненты проверяются под условия исходной задачи. Если они подходят, то решают задачу дискретной оптимизации для созданного набора компонент. В случае получения оптимального решения новый компонент встраивается в систему компонентов. В случае отрицательного решения происходит переход на этап модификации и выработки нового набора компонент.

5. Алгоритм искусственного пчелиного роя

Алгоритм искусственного пчелиного роя в наибольшей степени подходит для регенерации программного обеспечения. Алгоритм искусственного пчелиного роя (АПР) напрямую не подходит к регенерации ПК. Поэтому для этой цели используют алгоритм модифицированного пчелиного роя (МАПР). Он относится к эвристическим полиномиальным алгоритмам оптимизации, разработанным на основе моделирования поведения роя пчел, собирающих нектар. Модель поведения роя пчел, построенная человеком, то есть модель искусственного пчелиного роя, включает три основных параметра: источники питания, рабочие фуражиры, нерабочие фуражиры [15]. Основная идея подхода применительно к регенерации ПК в выделении групп мультиагентов, которые решают свои задачи. Здесь следует отметить принципиальное отличие алгоритма МАПР от муравьиного алгоритма. В муравьином алгоритме задача сводится к обработке одного эффективного источника или генерального алгоритма. В МАРП принципиально каждая группа агентов имеет свой алгоритм. Это говорит о том, что данный алгоритм является субсидиарным. Блок-схема МАПР приведена на рисунке 2.

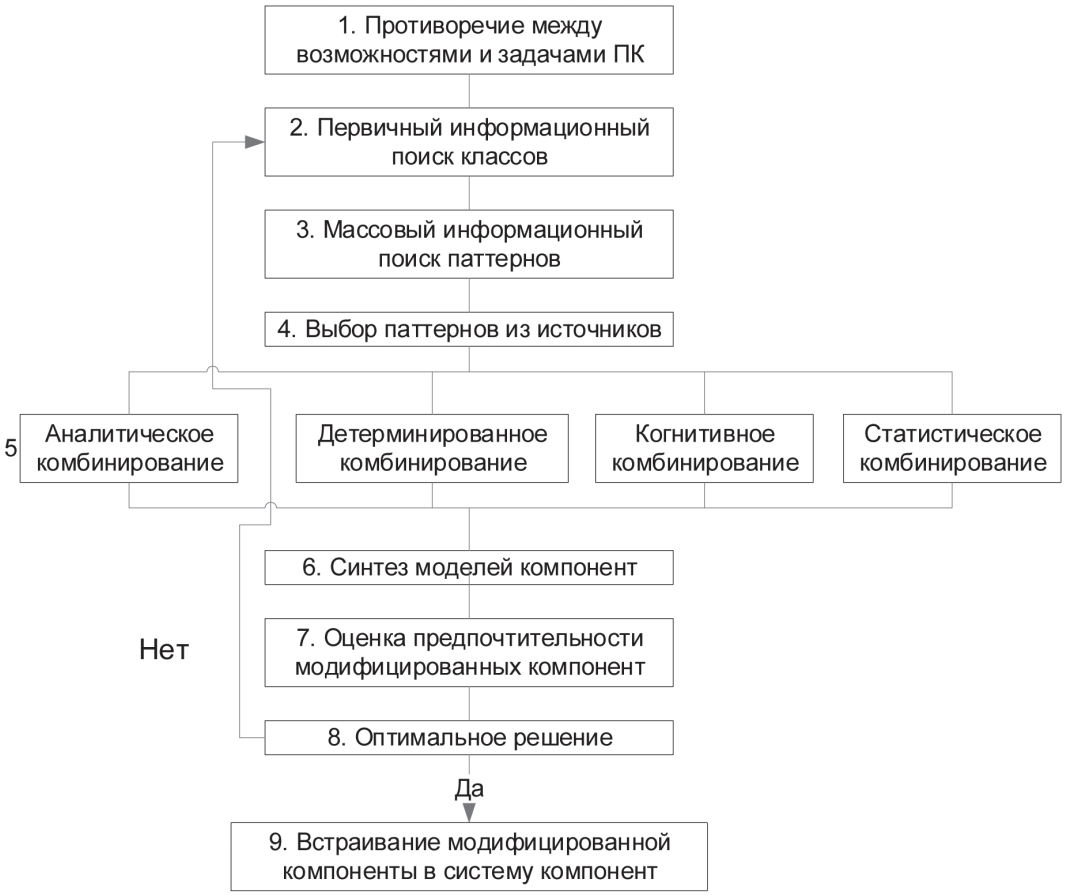

Рисунок 2 – Блок-схема регенерации программных компонент на основе модифицированного МАПР

Если схема на рисунке 1 ориентирована на множества, то схема на рисунке 2 ориентирована на группы мультиагентов. Причем их больше, чем в АПР. Цифрами обозначены следующие группы: 1 – высшее руководство и аналитики; 2, 3 – специалисты в области информационного поиска; 4, 5, 6 – программисты; 7, 8 – эксперты и специалисты в области оптимизации; 9 – специалисты в области системной инженерии. Таким образом, количество групп 5 и это задает 5 субсидиарных алгоритмов в дополнение к главному алгоритму создания новой системы ПК. Применительно к регенерации ПК наличие решения оптимизационной задачи означает возможность регенерации программного компонента или его модернизации.

6. Применение логистического уравнения для исследования системы

Совокупные действия пчел (агентов) можно описать также с помощью логистического уравнения. Исследование сложных систем, потребляющих ресурсы, приводит к необходимости моделирования потребления ресурсов. Для этой цели подходит модель, которая описывает процессы самореплика-ции и их подавление в среде с ограниченными ресурсами [4; 20].

Уравнение, описывающее процесс потребления ресурсов, называется логистическим уравнением. Логистическое уравнение, также известное как уравнение Ферхюльста (Pitrre Francois Verhulst), изначально появилось при рассмотрении модели экономического роста [21].

Обозначим через P потребление ресурсов или результативность системы через t время. Такая модель сводится к дифференциальному уравнению:

$$\frac{dP}{dt} = rP\left(1 - \frac{P}{K}\right). \tag{3}$$

В этой записи параметр r характеризует скорость расхода ресурсов, а K – ресурсную емкость среды. На основе (3) определяется две стратегии поведения системы: стратегия r предполагает интенсивную деятельность и короткий жизненный цикл; стратегия K предполагает медленное потребление ресурсов и длительный жизненный цикл. Решением уравнения (3) является логистическая функция, S-образная кривая (логистическая кривая):

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

для которой существует предел

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

Уравнение (3) не приемлемо для многих систем. Это обусловлено тем, что только в некоторых биологических системах популяция начинается с нулевого значения (пара особей) или нулевого ресурса. Для сложных систем существует нижний предел ресурсов, отличный от нуля. Нижний предел означает количество ресурсов, ниже которого система не функционирует. Верхний предел обусловлен ограниченными возможностями системы переработки ресурсов. Поэтому для сложных систем уравнение, моделирующее процесс функционирования системы на основе потребления ресурсов, имеет вид (4) [4]

$$\frac{dy}{dt} = a\left(y - k_1\right)\left(k_2 - y\right). \tag{4}$$

В выражении (4) y – величина, связанная с потреблением ресурсов; k_1 – нижняя граница потребляемых ресурсов; k_2 – верхняя граница потребляемых ресурсов; a – постоянная величина, характеризующая интенсивность потребления ресурсов.

В рамках модели (4) можно задавать пределы ресурсов и, зная скорость их потребления, получить решение для динамической модели. На рисунке 3 приведен результат экспериментального расчета в условных единицах решения уравнения (4) для сложной системы, потребляющей ресурсы, согласно модели (1). Можно упростить решение логистического уравнения до простого вида, которое используют в однопараметрической модели Раша [18]

$$P(t) = \frac{P_0 e^{t-a}}{1 + P_0 e^{t-a}}. \quad (5)$$

Такая кривая приведена на рисунке 3, она известна в математике и относится к классу сигмоид (sigmoid). Сигмоида – гладкая монотонная нелинейная S-образная возрастающая функция, которая применяется для отражения процесса накопления и его предела. Величина *a* задает сдвиг от начала координат вправо.

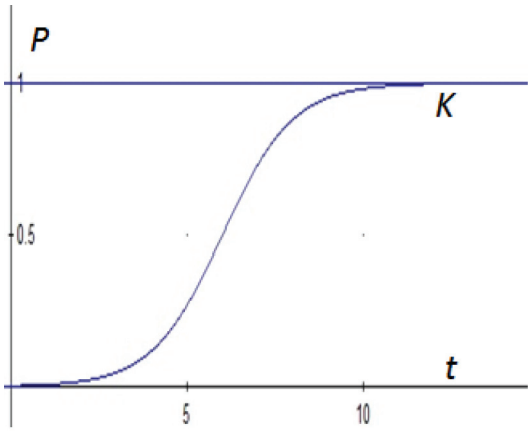

Рисунок 3 – Логистическая кривая потребления ресурсов сложной системой

Логистическая кривая потребления ресурсов на рисунке говорит о том, что есть предел потребления ресурсов каждой системой. В теории связи – это предел пропускной способности канала связи. В вычислительных технологиях – это предел скорости вычислений.

Если в качестве аргумента выбрать параметр *a*, то получаем другое решение уравнения и другую кривую. Решение (6) в этом случае имеет вид, приведенный на рисунке 4.

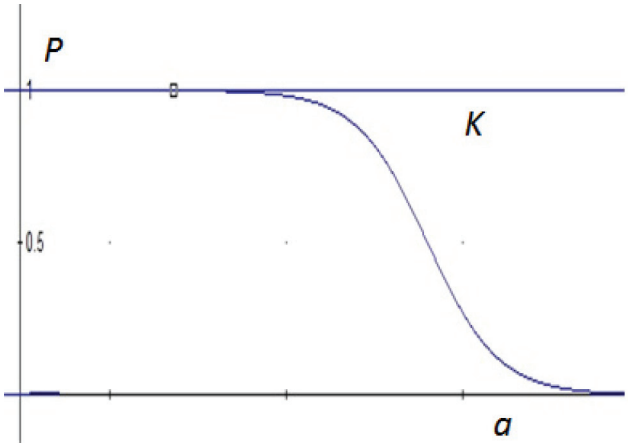

Рисунок 4 – Логистическая кривая потребления ресурсов

Оба вида кривых применяют в модели Раша [18]. Рисунок 4 характеризует производительность системы, а рисунок 3 характеризует расход ресурсов. Применительно к регенерации ПО рисунок 3 говорит о наступлении предела производительности системы или алгоритма. Это требует регенерации системы. Применительно к регенерации ПО рисунок 4 говорит об исчерпании ресурсов для работы системы и необходимости их регенерации.

Заключение

Регенерация в информационных системах применима не во всех случаях, а при условии тринитарной связи ПО, ТО, СД. Регенерация программного обеспечения или программных компонент применима не во всех случаях, а только при условии того, что совокупность программ и алгоритмов образует связанную целостную систему. Регенерация – это процесс взаимосвязанного обновления ПО, ТО, СД. Для регенерации пригодны различные методы. Одним из перспективных методов регенерации следует считать совокупность эволюционных методов. Технология регенерации является современной технологией модернизации комплекса: технологии вычисления, технологии сбора и обработки данных. В отличие от разрозненного обновления данных, программ и технологий, данная технология регенерации принимает во внимание взаимосвязь параметров алгоритмической или информационной системы. Эта взаимосвязь требует учитывать условие комплементарности и решения задачи локальной оптимизации для нахождения комплементарного решения. В область эволюционного моделирования входят не только алгоритмы, работающие по этому принципу, но и другие уравнения развития популяций. Для регенерации программного обеспечения не все методы эволюционного программирования и моделирования подходят в равной степени. В наибольшей степени подходит модель искусственного роя пчел и модели, связанные с решением логистического уравнения. Принципиальным отличием регенерации от обновления является необходимость учета и соблюдения отношений комплементарности и оптимизация на этой основе.

Список литературы

1. *Буравцев А.В., Щенников А.Е.* Информационный подход в системной и программной инженерии // Славянский форум. – 2018. – № 1 (19). – С. 17–23.
2. Искусственный интеллект. Генетические алгоритмы [Электронный ресурс]. – URL: https://www.studfile.net/preview/986657/page:13 (дата обращения: 11.09.2019).
3. *Монахов С.В., Савиных В.П., Цветков В.Я.* Методология анализа и проектирования сложных информационных систем. – М.: Просвещение, 2005. – 264 с.
4. *Пригожин И., Стенгерс И.* Порядок из хаоса: новый диалог человека с природой. – М.: Прогресс, 1986. – 432 с.
5. *Снитюк В.Е.* Эволюционное моделирование и программирование жизненного цикла технических систем в детерминированных условиях // Искусственный интеллект. – 2006. – № 4. – С. 10–15.
6. *Уральский Н.Б.* Разработка моделей и алгоритмов составления оптимальных расписаний выполнения программных модулей в вычислительной сети на основе эволюционного подхода: дис. ... канд. техн. наук: 05.13.11. – М., 2017. – 137 с.
7. *Цветков В.Я.* Комплементарность информационных ресурсов // Международный журнал прикладных и фундаментальных исследований. – 2016. – № 2. – С. 182–185.
8. *Цветков В.Я.* Комплементарные отношения // Научный вестник Новосибирского государственного технического университета. – 2019. – № 2 (75). – С. 101–114.
9. *Цветков В.Я.* Стандартизация информационных программных средств и программных продуктов. – М.: МГУГиК, 2000. – 116 с.
10. *Щенников А.Н.* Комплементарность сложных вычислений // Славянский форум. – 2018. – № 2 (20). – С. 118–123.
11. *Aljahdali S.H., El-Telbany M.E.* Software reliability prediction using multi-objective genetic algorithm // IEEE/ACS International Conference on Computer Systems and Applications. – Rabat, 2009. – P. 293–300.
12. *Cinar A.C., Korkmaz S., Kiran M.S.* A discrete tree-seed algorithm for solving symmetric traveling salesman problem [Электронный ресурс] // Engineering Science and Technology, an International Journal. – 2019. – URL: https://wwwdoi.org/10.1016/j.jestch.2019.11.005 (дата обращения: 11.04.2019).
13. *Huang S.-J., Chiu N.-H.* Optimization of analogy weights by genetic algorithm for software effort estimation // Information and Software Technology. – 2006. – Vol. 48, No. 11. – P. 1034–1045.
14. ISO/IEC 25010:2011 “Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models” [Электронный ресурс]. – URL: https://www.iso.org/standard/35733.html (дата обращения: 12.12.2019).

15. *Karaboga D., Basturk B.* A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm // Journal of Global Optimization. – 2007. – Vol. 39, is. 3. – P. 459–471.
16. *Kumari A.C., Srinivas K., Gupta M.P.* Software module clustering using a hyper-heuristic based multi-objective genetic algorithm // 3rd IEEE International Advance Computing Conference (IACC). – IEEE, 2013. – P. 813–818.
17. *Ouni A., Kessentini M., Sahraoui H., Hamdi M.S.* The use of development history in software refactoring using a multi-objective evolutionary algorithm // Proceedings of the 15th annual conference on Genetic and evolutionary computation (GECCO '13) / ed. C. Blum. – ACM, 2013. – P. 1461–1468.
18. *Rasch G.* Probabilistic models for some intelligence and attainment tests. – Chicago: MESA Press, 1980. – 199 s.
19. *Saha D., Mitra R.S., Basu A.* Hardware software partitioning using genetic algorithm // Proceedings Tenth International Conference on VLSI Design. – IEEE, 1997. – P. 155–160.
20. *Tsvetkov V.Ya.* Resource Method of Information System Life Cycle Estimation // European Journal of Technology and Design. – 2014. – № 2 (4). – P. 86–91.
21. *Verhulst P.F.* Notice sur la loi que la population poursuit dans son accroissement // Correspondance mathematique et physique. – 1838. – Vol. 10. – P. 113–121.

References

1. *Buravcev A.V., Shchennikov A.E.* Informacionnyj podhod v sistemnoj i programmnoj inzhenerii // Slavyanskij forum. – 2018. – № 1 (19). – S. 17–23.
2. Iskusstvennyj intellekt. Geneticheskie algoritmy [Elektronnyj resurs]. – URL: https://www.studfile.net/preview/986657/page:13 (data obrashcheniya: 11.09.2019).
3. *Monahov S.V., Savinyh V.P., Cvetkov V.Ya.* Metodologiya analiza i proektirovaniya slozhnyh informacionnyh sistem. – M.: Prosveshchenie, 2005. – 264 s.
4. *Prigozhin I., Stengers I.* Poryadok iz haosa: novyj dialog cheloveka s prirodoj. – M.: Progress, 1986. – 432 s.
5. *Snityuk V.E.* Evolyucionnoe modelirovanie i programmirovanie zhiznennogo cikla tekhnicheskih sistem v determinirovannyh usloviyah // Iskusstvennyj intellekt. – 2006. – № 4. – S. 10–15.
6. *Ural'skij N.B.* Razrabotka modelej i algoritmov sostavleniya optimal'nyh raspisanij vypolneniya programmnyh modulej v vychislitel'noj seti na osnove evolyucionnogo podhoda: dis. ... kand. tekhn. nauk: 05.13.11. – M., 2017. – 137 s.
7. *Cvetkov V.Ya.* Komplementarnost' informacionnyh resursov // Mezhdunarodnyj zhurnal prikladnyh i fundamental'nyh issledovanij. – 2016. – № 2. – S. 182–185.
8. *Cvetkov V.Ya.* Komplementarnye otnosheniya // Nauchnyj vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta. – 2019. – № 2 (75). – S. 101–114.
9. *Cvetkov V.Ya.* Standartizaciya informacionnyh programmnyh sredstv i programmnyh produktov. – M.: MGUGiK, 2000. – 116 s.
10. *Shchennikov A.N.* Komplementarnost' slozhnyh vychislenij // Slavyanskij forum. – 2018. – № 2 (20). – S. 118–123.
11. *Aljahdali S.H., El-Telbany M.E.* Software reliability prediction using multi-objective genetic algorithm // IEEE/ACS International Conference on Computer Systems and Applications. – Rabat, 2009. – P. 293–300.
12. *Cinar A.C., Korkmaz S., Kiran M.S.* A discrete tree-seed algorithm for solving symmetric traveling salesman problem [Elektronnyj resurs] // Engineering Science and Technology, an International Journal. – 2019. – URL: https://wwwdoi.org/10.1016/j.jestch.2019.11.005 (data obrashcheniya: 11.04.2019).
13. *Huang S.-J., Chiu N.-H.* Optimization of analogy weights by genetic algorithm for software effort estimation // Information and Software Technology. – 2006. – Vol. 48, No. 11. – P. 1034–1045.
14. ISO/IEC 25010:2011 "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models" [Elektronnyj resurs]. – URL: https:// www.iso.org/standard/35733.html (data obrashcheniya: 12.12.2019).
15. *Karaboga D., Basturk B.* A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm // Journal of Global Optimization. – 2007. – Vol. 39, is. 3. – P. 459–471.

16. *Kumari A.C., Srinivas K., Gupta M.P.* Software module clustering using a hyper-heuristic based multiobjective genetic algorithm // 3rd IEEE International Advance Computing Conference (IACC). – IEEE, 2013. – P. 813–818.
17. *Ouni A., Kessentini M., Sahraoui H., Hamdi M.S.* The use of development history in software refactoring using a multi-objective evolutionary algorithm // Proceedings of the 15th annual conference on Genetic and evolutionary computation (GECCO '13) / ed. C. Blum. – ACM, 2013. – P. 1461–1468.
18. *Rasch G.* Probabilistic models for some intelligence and attainment tests. – Chicago: MESA Press, 1980. – 199 s.
19. *Saha D., Mitra R.S., Basu A.* Hardware software partitioning using genetic algorithm // Proceedings Tenth International Conference on VLSI Design. – IEEE, 1997. – P. 155–160.
20. *Tsvetkov V.Ya.* Resource Method of Information System Life Cycle Estimation // European Journal of Technology and Design. – 2014. – № 2 (4). – P. 86–91.
21. *Verhulst P.F.* Notice sur la loi que la population poursuit dans son accroissement // Correspondance mathematique et physique. – 1838. – Vol. 10. – P. 113–121.