СИСТЕМНЫЙ АНАЛИЗ И УПРАВЛЕНИЕ
SYSTEM ANALYSIS AND PROCESSING OF KNOWLEDGE

УДК 004:006.44
DOI: 10.18413/2411-3808-2018-45-2-322-332

ИНТЕГРИРОВАННАЯ МОДЕЛЬ ПОДДЕРЖКИ ЖИЗНЕННОГО ЦИКЛА ПРОЕКТОВ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

THE INTEGRATED MODEL OF AUTOMATIC SYSTEMS PROJECTS LIFE CYCLE SUPPORTING

Р.Г. Асадуллаев, В.В. Ломакин
R.G. Asadullaev, V.V. Lomakin

Белгородский государственный национальный исследовательский университет,
Россия, 308015, г. Белгород, ул. Победы, 85

Belgorod National Research University, Russia, 85 Pobeda St, Belgorod, 308015, Russia

E-mail: asadullaev@bsu.edu.ru, lomakin@bsu.edu.ru

Аннотация
Статья посвящена проблеме управления жизненным циклом автоматизированных систем. Проведен анализ моделей жизненного цикла, отечественных и зарубежных стандартов, а также индустриальных и корпоративных методологий разработки информационных технологий. Выделены ключевые особенности крупных проектов разработки автоматизированных систем. На основании выявленных недостатков существующих моделей жизненного цикла предложена интегрированная модель управления жизненным циклом в виде алгоритма, сочетающая достоинства известных моделей с учетом выделенных особенностей реализации крупных проектов. Разработанная модель адаптирована под любой состав этапов и процессов жизненного цикла, который определит разработчик. С учетом описанной интегрированной модели жизненного цикла и требований к необходимости оценки рисков, разработана схема базы данных, содержащая временные и качественные параметров оценки как проекта в целом, так и отдельных составляющих.

Abstract
The article deals with the problem of life cycle management of automated systems. Life cycle models were analyzed, domestic and foreign standards and industrial and corporate methodologies of information technologies development were investigated. The key features of major projects for automated systems design were defined. Based on identified shortcomings of existing life cycle models an integrated model of life cycle management was proposed as an algorithm, which is combining strengths of known models, taking into account the allocated levels of major projects implementation. The developed model is adapted for any phases and processes composition, which can be determined by developer. In view of outlined integrated life cycle model and bearing in mind the requirements on the need for using risk-assessment tools, database schema containing time and quality evaluation parameters of project and its individual components has been developed. The approach of preliminary data preparation has been proposed in order to create incoming vectors for training the neural network predicting the target deadline of project's different stages.

Ключевые слова: системный анализ, модель жизненного цикла, автоматизированная система, моделирование систем, методология управления жизненным циклом.
Keywords: system analysis, life cycle model, automated system, system modeling, life cycle management methodology.

Введение

Процесс создания автоматизированной системы (АС) является сложной и многоэтапной задачей, требующей наличия механизмов, позволяющих систематизировать работу коллектива исполнителей с учетом выбранной методологии внедрения. Реализация проектов, направленных на автоматизацию деятельности крупных организаций посредством разработки информационно-технологического обеспечения, осложняется множеством взаимно пересекающихся бизнес-процессов, что, в свою очередь, предполагает длительный период выполнения всех этапов проекта. Длительность реализации подобных проектов измеряется годами. При этом заказчик может потребовать помимо отчетной документации о ходе реализации проекта и работоспособные промежуточные версии системы для формирования представления о реализованном функционале с целью постоянной корректировки требований к конечному продукту, а также для подтверждения целевого использования вложенных инвестиций. В свою очередь, это полезно для исполнителя, так как наглядное представление проделанной работы позволяет отслеживать ход практической реализации проекта и проводить своевременную корректировку реализуемого функционала системы.

Управление жизненным циклом систем регламентируется международным стандартом [ISO/IEC/IEEE 15288:2015, 2015], который распространяется на системы в целом, в том числе включая технические и программные средства. В международном стандарте [ISO/IEC 12207:2017, 2017] рассматривается понимание жизненного цикла лишь для программных средств, исключая из внимания аппаратную составляющую. Российским аналогом стандарта [ISO/IEC 12207:2017, 2017] является [ГОСТ Р ИСО/МЭК 12207-2010, 2012]. Данные стандарты не привязаны к какой-либо модели жизненного цикла. В них дается рекомендация по выбору модели с учетом имеющегося опыта у разработчика и специфики проекта. При этом стандарт [ГОСТ Р ИСО/МЭК 12207-2010, 2012] не регламентирует состав стадий жизненного цикла.

С целью упрощения совместного использования международных стандартов [ISO/IEC/IEEE 15288:2015, 2015; ISO/IEC 12207:2017, 2017] в процессе проектирования модели жизненного цикла разработана группа стандартов ИСО/МЭК 24748, представляющих собой руководства по применению стандартов. Международный стандарт [ISO/IEC TR 24748-2:2011, 2011] является руководством по применению стандарта ИСО/МЭК 15288, а стандарт [ISO/IEC TR 24748-3:2011, 2011] представляет собой руководство по применению ИСО/МЭК 12207. Для формирования у российских разработчиков представления о применении международных стандартов разработаны национальные стандарты [ГОСТ Р 57102-2016, 2017], идентичный стандарту [ISO/IEC TR 24748-2:2011, 2011], и [ГОСТ Р 56923-2016, 2017], идентичный стандарту [ISO/IEC TR 24748-3:2011, 2011].

В области управления жизненным циклом основу отечественной нормативной базы документирования информационных систем составляет комплекс стандартов единой системы программной документации (ЕСПД), представляющий собой систему межгосударственных стандартов стран СНГ (ГОСТ). В процессе управления проектом по созданию информационно-технологического обеспечения могут возникнуть проблемы с выбором рационального стандарта для разработки технической документации. В документе [ГОСТ 34.003-90, 1992] дано определение автоматизированной системы, где под АС подразумевается система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций. Здесь же приводятся разновидности АС в зависимости от вида деятельности предприятия и вида объекта управления. Вне зависимости от типа АС в технической документации должны быть описаны следующие положения: цель АС, необходимый обслуживающий и эксплуатационный персонал, комплекс необходимых и достаточных программных и технических средств, обеспечивающих корректную работу АС. Документ [ГОСТ 34.602-89, 1990] устанавливает содержание, состав и правила оформления технического задания на создание, развитие или модернизацию АС. АС представляет собой комплекс технического

и программного обеспечения. Согласно стандарту [ГОСТ 19.101-77, 1980] программный комплекс – это программа, состоящая из двух или более компонентов и (или) комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса. Следовательно, программный комплекс может быть реализован как в составе АС, так и отдельно. Несмотря на тот факт, что часть стандартов ЕСПД морально устарела, они до сих пор успешно применяются разработчиками. Таким образом, в процессе реализации проекта по созданию АС необходимо использовать рекомендации группы стандартов серии ГОСТ 34.XXX, а при подготовке документации на программный комплекс необходимо использовать группу стандартов серии ГОСТ 19.XXX.

Все стандарты носят рекомендательный характер. Таким образом, руководитель проекта по разработке АС, реализуя функции по управлению жизненным циклом АС, должен самостоятельно выбирать модели управления жизненным циклом и методику разработки, основываясь на личном опыте и действующих стандартах. Для эффективной организации данного процесса необходимо уделять особое внимание следующим положениям:

– учет требований стандартов, регламентирующих процессы создания АС и управления их жизненным циклом;

– использование рациональной модели управления жизненным циклом АС;

– возможность оценки рисков несоблюдения сроков реализации проекта и этапов в частности;

– наличие средств оценки вклада каждого сотрудника, вовлеченного в реализацию проекта.

Объекты и методы исследования

Выделим следующие стадии жизненного цикла АС: формирование требований к АС, разработка концепции АС, техническое задание, эскизный проект, технический проект, рабочая документация, ввод в действие, сопровождение АС. Каждая стадия подразумевает выполнение нескольких этапов работ. В зависимости от специфики условий реализации проекта и личного опыта разработчика возможно исключение стадии эскизного проекта, а также возможность невыполнения некоторых этапов работ на всех стадиях, слияние стадий подготовки рабочей документации и технического проекта в одну стадию технорабочего проекта. С учетом индивидуальных особенностей проекта по созданию АС может быть реализовано выполнение отдельных этапов работ до завершения предшествующих стадий, а также параллельное выполнение этапов работ [ГОСТ 34.601-90, 1992].

Существующий стандарт [ГОСТ 19.102-77, 1980] регламентирует стадии разработки программ и программной документации для вычислительных машин, комплексов и систем. В качестве стадий выделяют техническое задание, эскизный проект, технический проект, рабочий проект, внедрение. Стадии разработки декомпозируются на группы этапов работ. Допускается исключать вторую и третью стадии, объединять, исключать этапы работ и (или) их содержание, а также вводить другие этапы работ по согласованию с заказчиком.

Жизненный цикл автоматизированной системы – совокупность взаимосвязанных процессов создания и последовательного изменения состояния АС от формирования исходных требований к ней до окончания эксплуатации и утилизации комплекса средств автоматизации [ГОСТ 34.003-90, 1992]. Из данного определения следует, что выбор модели управления жизненным циклом должен осуществляется исходя того факта, что поддержка АС будет реализована вплоть до момента ее утилизации. Модель жизненного цикла – структура процессов и действий, связанных с жизненным циклом, организуемых в стадии, которые также служат в качестве общей ссылки для установления связей и взаимопонимания сторон [ГОСТ Р ИСО/МЭК 12207-2010, 2012]. В настоящее время широко известны следующие классические модели жизненного цикла информационных систем:

– Каскадная (водопадная) модель. Особенностью данной модели является то, что выполнение этапов строго последовательно. Переход к следующему этапу осуществляется

только по завершении предыдущего этапа, и обратный переход между этапами невозможен. Основным достоинством каскадной модели является возможность планирования сроков реализации и стоимости работ по каждому этапу. Подобная жесткая структура делает модель неприменимой на практике, так как процесс реальной разработки АС почти никогда не проходит по строго заданной схеме, а требования заказчика определены лишь частично [Royce, 1970].

– Каскадная модель с промежуточным контролем. Представляет собой вариацию каскадной модели с возможностью возвращения к предыдущим этапам с целью корректировки уже проработанных вопросов с учетом новых реалий. Промежуточные корректировки приводят к увеличению регламентированного времени, отведенного на разработку, что приводит к снижению риска получения на выходе некачественной АС. Общим недостатком каскадной модели и ее вариаций является тот факт, что оценка результирующей системы производится только по окончательным результатам внедрения. Следовательно, проявляется риск, заключающийся в том, что полученная на выходе система может морально устареть и не будет востребована на рынке [Зараменских, 2014].

– Модель разработки через тестирование (V-образная модель). Суть модели заключается в постепенном повышении степени детализации проекта во времени и параллельном проведении горизонтальных фаз и итераций. Полученные результаты каждой фазы в левой стороне оказывают влияние на итерации тестирования и компоновки проекта с правой стороны. Модель снижает проектные риски за счет проверки с участием заказчика промежуточных результатов на ранних стадиях реализации проекта. При этом V модель не учитывает этапы сопровождения и утилизации [Зараменских, 2014].

– Инкрементная модель. Управление жизненным циклом основывается на последовательности инкрементов (увеличение на 1), каждый из которых реализует линейную последовательность выполнения всех этапов разработки и формирует инкремент АС. Таким образом, с каждым инкрементом реализуются новые возможности базового продукта, полученного в первом инкременте. При этом каждый инкремент представляет собой работающий продукт [Сенник, Гребенников, 2015].

– Спиральная модель. В данной модели переходы от одного этапа реализации системы к другому не являются жестко последовательными, то есть допускается возможность начала работ над следующим этапом до завершения предыдущего. Основная идея заключается в прохождении всех этапов жизненного цикла в несколько итераций, создавая каждый раз новую версию системы (прототип) и сверяя актуальность требований. В моделе добавлен этап анализа рисков, отсутствующий в моделях каскадного типа. Недостатком модели выступают повышенные требования к заказчику и сложности управления временем разработки [Boehm, 1988].

– Компонентно-ориентированная модель. Представляет собой развитие спиральной модели и основывается на концепции повторного использования существующих разработок при создании новой АС, в частности программных компонентов. Таким образом, все компоненты, созданные в процессе реализации предыдущих проектов, записываются в библиотеку. При создании нового проекта в библиотеке реализуется поиск подходящих компонентов. Если подходящие компоненты находятся, то они используются в новом проекте повторно, в противном случае создаются новые компоненты и добавляются в библиотеку. Заявленный эффект уменьшения затрат и увеличения производительности достигается лишь в том случае, когда имеется значительный опыт реализаций проектов [Брауде, 2004].

На основании классических моделей жизненного цикла разработаны индустриальные стандарты и методологии, а также методологии крупных компаний:

– Корпоративная методология Oracle Unified Method (OUM), руководство по настройке и типовой структуре работ. OUM описывает управление программами и проектами, поддержку ИТ-архитектуры и подход к реализации проектов. Методология выделяет пять стадий с возможностью итераций внутри каждой из них [OUM, 2014].

– Корпоративная методология Rational Unified Process (RUP), выделяющая четыре стадии жизненного цикла и девять процессов. В основе лежит поэтапное моделирование разрабатываемого продукта. В RUP заложены такие принципы, как идентификация и непрерывное устранение рисков, постоянное ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки, непрерывное обеспечение качества на всех этапах разработки проекта, ключевая роль в команде над проектом закреплена архитекторами [RUP, 1998].

– Корпоративная методология Microsoft Solutions Framework (MSF). В состав входят модель проектной группы (формирование команды, адаптируемой для выполнения любого проекта), модель процессов (основывается на интеграции каскадной и спиральной моделях), дисциплина управления проектами (инструментарий для достижения целей проектов с учетом заданных параметров качества, бюджета, сроков и иных ограничений), дисциплина управления рисками (постоянный мониторинг рисков для своевременного принятия решений на протяжении всего жизненного цикла проекта), дисциплина управления подготовкой (формализуются основополагающие принципы методологии и формируются рекомендации по практическому применению MSF к управлению знаниями на протяжении всего жизненного цикла системы) [FSDM, 2008].

– Индустриальная методология экстремального программирования Extreme Programming (XP). В основу методологии заложена итерационно-инкрементная модель быстрого создания прототипов системы с учетом поступивших требований. Основными стадиями каждого цикла методологии являются вброс архитектуры (формализуется видение проекта и принимаются решения по используемым технологиям и архитектуре), история использования (сбор требований в виде сценариев работы отдельных функций), планирование версии (формирование функционального состава версии), разработка (реализуются только отобранные функции), тестирование, выпуск релиза. Методология основывается на таких принципах, как: сотрудники важнее процессов и инструментов, работающая программа важнее документации, сотрудничество с заказчиком важнее обсуждения деталей, отработка изменений важней соблюдения плана [Зараменских, 2014].

– Международный стандарт проектного управления Project Management Body of Knowledge (PMBOK), содержащий свод знаний по управлению проектами, разделенными на 9 областей знаний, каждая из которых декомпозируется на подразделы управления: интеграция проекта, объем работ, время выполнения, стоимость, качество, персонал, коммуникации, риски, закупки и поставки [Guide PMBOK, 2018].

– Руководство к методическому справочнику по программной инженерии Guide to the Software Engineering Body of Know-ledge (SWEBOK). Представляет собой систематизированный справочник по десяти областям знаний разработки программного обеспечения. В SWEBOK выделяются следующие основные процессы: программные требования, дизайн и архитектура, конструирование программного обеспечения, тестирование, эксплуатация и поддержка программного обеспечения, конфигурационное управление, управление в программной инженерии, процессы программной инженерии, инструменты и методы, качество программного обеспечения [Зараменских, 2014].

– Методология управления ИТ-проектами с точки зрения продукта Projects In Controlled Environments (PRINCE2). Методология основывается на семи принципах (постоянное бизнес-обоснование, обучение на собственном опыте, формирование ролей и ответственности, управление по стадиям, управление исключениями, фокус на продуктах, адаптация к конкретному проекту), семи элементах и семи процессах (запуск, навигация, инициация, управление границами этапа, контроль этапа, управление созданием продукта, завершение проекта) [Зараменских, 2014].

В работе [Зараменских, 2014] рассматривается менеджмент жизненного цикла АС с позиции проектной деятельности. Современные стандарты и методологии объединяет наличие средств постоянного контроля процесса выполнения работ и достижения поставленных целей, а также наличие процессов управления рисками проекта, заключающихся в

планировании, идентификации, качественном и количественном анализе, планировании реагирования на риски и мониторинге в ходе реализации проекта.

Таким образом, независимо от выбранного подхода управления жизненным циклом АС, необходимо наличие инструментальных средств, позволяющих оценивать риски с учетом критериев и ограничений проекта разработки АС. При этом могут быть привлечены эксперты, что требует наличия средств автоматизации экспертных оценок [Ломакин, Лифиренко, 2014].

В процессе создания масштабных проектов АС можно выделить ряд ключевых особенностей, заключающихся в:

– большом временном диапазоне управления жизненным циклом продукта, требующем постоянного совершенствования, примером может послужить автоматизированная система управления наружным освещением, для которой время жизненного цикла составляет 25–50лет [Ломакин, Лифиренко, 2014];

– жестких требованиях к разработчику (опыт реализации проектов подобного масштаба, размер организации и так далее);

– необходимости соблюдения международных и российских стандартов;

– необходимости наличия средств оценки проектных рисков;

– высокой вероятности привлечения сторонних организаций для выполнения отдельных задач проекта.

Исходя из выделенных особенностей реализации масштабных проектов АС, сформируем интегрированную модель управления жизненным циклом, учитывающую достоинства существующих моделей:

1. В основу положена концепция спиральной модели с учетом возможности анализа рисков, при этом спиральная модель обеспечивает версионность проекта.

2. Внесенные элементы компонентно-ориентированной модели позволят сократить время реализации этапов проекта. Данное положение обосновывается тем, что для реализации масштабных проектов выбираются компании со значительным опытом, следовательно, у них должна быть обширная библиотека готовых решений.

3. Каждую версию реализации проекта предлагается выполнять в виде каскадной модели с промежуточным контролем, что позволит своевременно корректировать результат версии и учитывать реалии процесса разработки. Несмотря на тот факт, что жизненный цикл проекта длится годы, каждая версия проекта может быть декомпозирована на достаточно маленькие временные отрезки, позволяющие без значительных временных затрат производить возврат к завершенным стадиям жизненного цикла версии.

На рис. 1 представлен алгоритм жизненного цикла АС, разработанный с учетом описанной интегрированной модели. Опишем этапы исполнения алгоритма:

Итерация 1. Начало проекта. Инициализация множеств $S = \{S_h\}$ при $h = \overline{1,n}$ множество стадий для разрабатываемой версии АС, где n – количество стадий, определяемое индивидуально для каждой версии проекта V; $P = \{P_k\}$ при $k = \overline{1,m}$ множество процессов для каждой стадии, где m – количество процессов, определяемое индивидуально для каждой стадии S_h; $T = \{T_j\}$ при $j = \overline{1,z}$ множество задач для каждого процесса, где z – количество задач, определяемое индивидуально для каждого процесса P_k. Затем формируется первая версия проекта $V = 1$ и переход к итерации 2.

Итерация 2. Организуется цикл выполнения всех стадий текущей версии проекта $h = 1$, где h – счетчик цикла (номер стадии проекта). Если условие проверки окончания работ по стадиям версии $h \le S_h$ выполнено, то переход к итерации 3, в противном случае – переход к итерации 2.1.

Итерация 2.1. Если цель проекта достигнута, то переход к итерации 7, в противном случае осуществляется переход к реализации следующей версии проекта $V = V + 1$ (возврат к итерации 2).

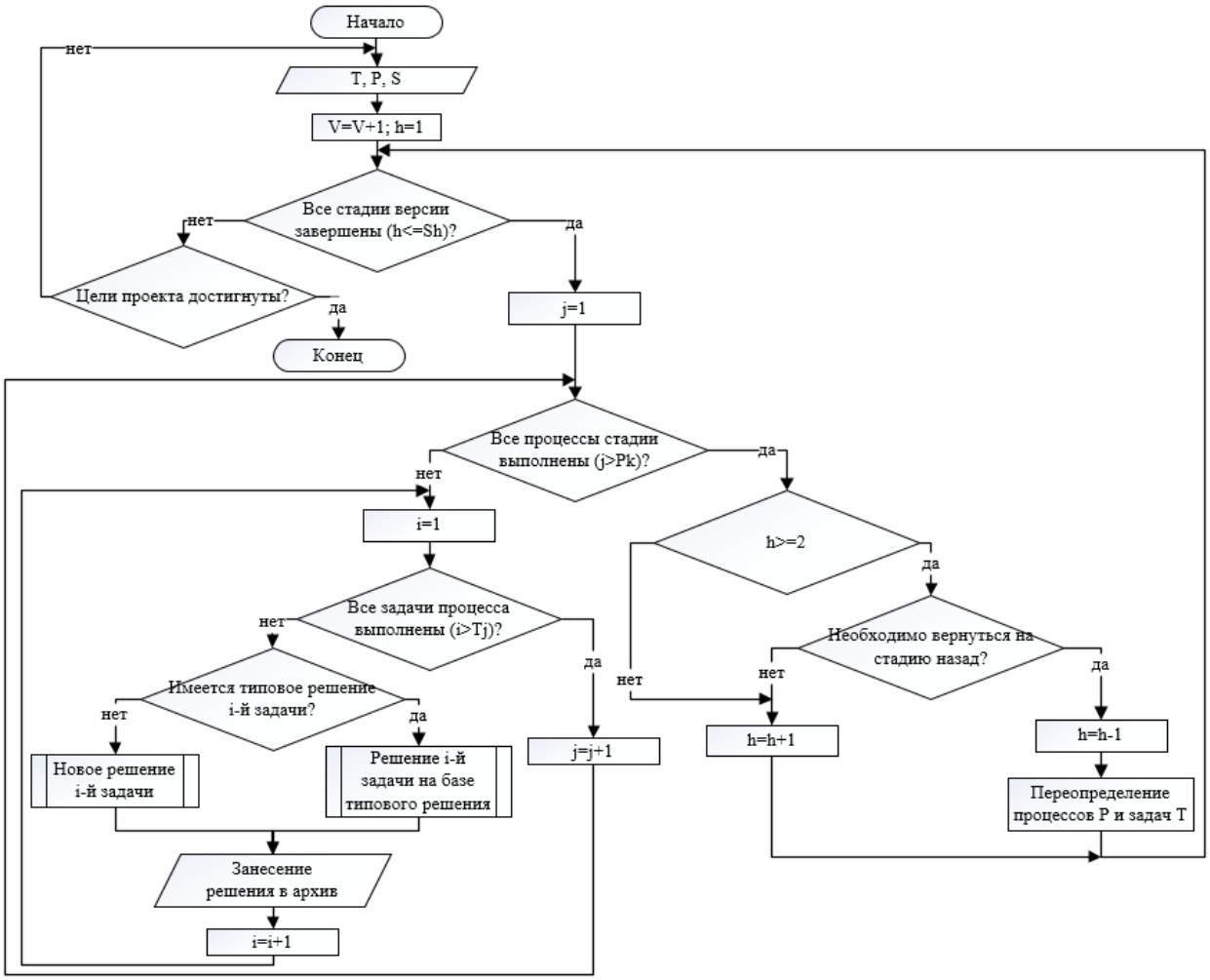

Рис. 1. Алгоритм жизненного цикла автоматизированной системы
Fig. 1. Algorithm of the life cycle of the automated system

Итерация 3. Запускается цикл выполнения процессов текущей стадии проекта $j = 1$, где j – счетчик цикла (номер процесса). Если условие $j > P_k$ не выполнено, то переход к итерации 4, иначе – к итерации 5.

Итерация 4. Если условие ($h \geq 2$) не выполнено (первая стадия), то переход к следующей стадии $h = h + 1$ (возврат к итерации 2), иначе переход к итерации 4.1.

Итерация 4.1. По результатам выполнения стадии обосновывается необходимость возврата к предыдущей стадии текущей версии проекта. Если таковая необходимость возникает, то ($h = h - 1$), переопределение множеств P и T и переход к итерации 2).

Итерация 5. Организуется цикл выполнения всех задач текущего процесса $i = 1$, где i – счетчик цикла (номер задачи процесса). Если условие проверки окончания работ по задачам процесса $i \leq T_j$ выполнено, то $j = j + 1$ и переход к итерации 3, в противном случае – переход к итерации 6.

Итерация 6. Если имеется типовое решение i-ой задачи, то применить найденное решение, занести новое полученное решение в архив и перейти к следующей задаче $i = i + 1$ (итерация 5), в противном случае получить новое решение, занести его в архив и перейти к итерации 5.

Итерация 7. Завершение проекта.

Структура алгоритма предполагает последовательное выполнения каждой стадии проекта и соответствующих процессов. Однако если оформить блоки в виде предопределенных процессов, возможна организация параллельного выполнения стадий и процессов.

При необходимости отражения параллельно протекающих процессов возможно использование в алгоритме (рис. 1) формальных средств сетей Петри.

В процессе разработки АС ограничениями выступают качество выходного продукта и цели проекта. Критериями успешной реализации проекта являются конечная стоимость и сроки реализации. С учетом описанной интегрированной модели жизненного цикла АС (рис. 1) и требований управления проектами, заключающихся в необходимости оценки рисков, разработана схема базы данных (рис. 2).

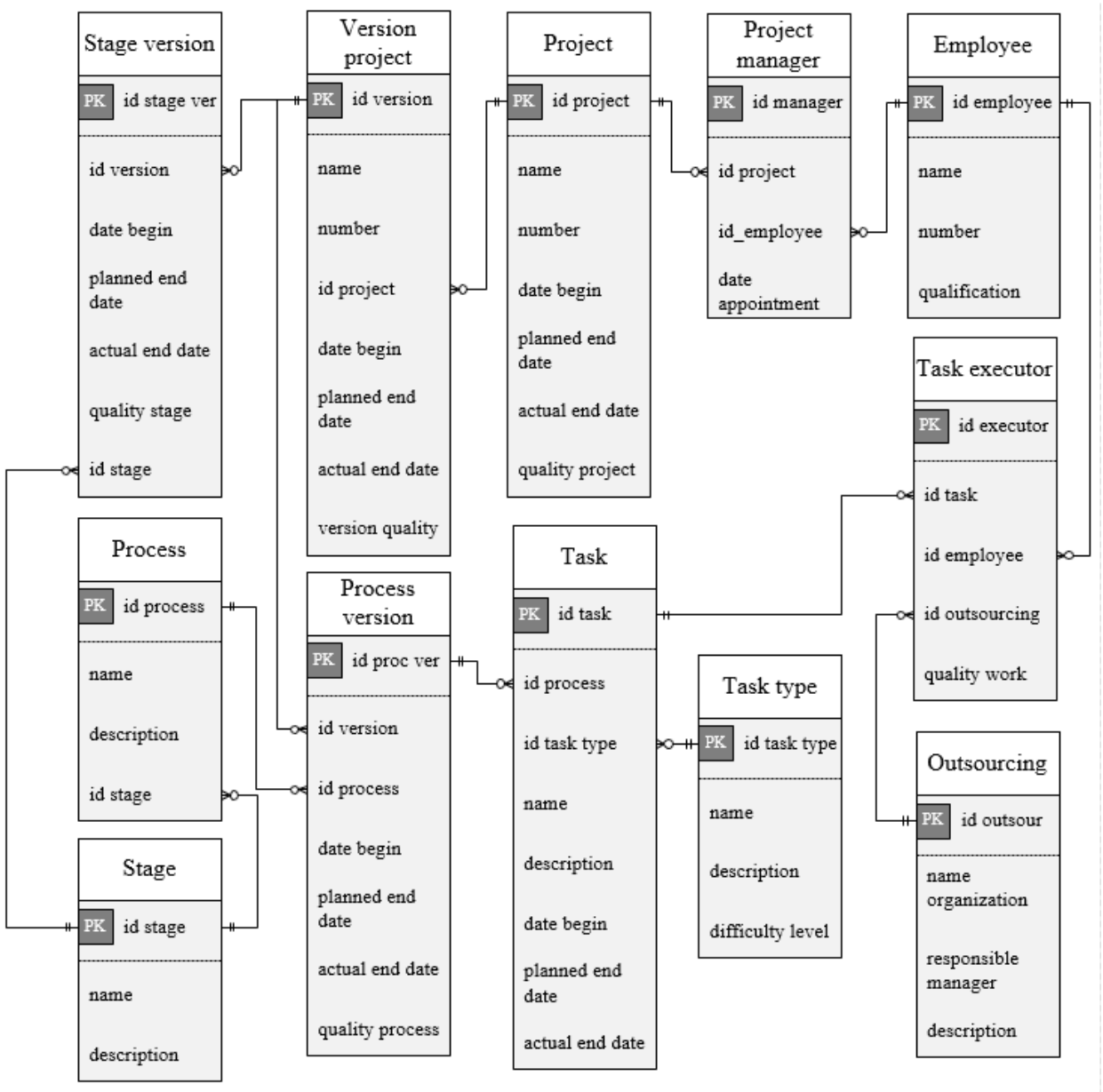

Рис. 2. Схема базы данных системы поддержки жизненного цикла АС
Fig. 2. Database schema of the life cycle support system of the AS

Сущность Project содержит сведения о проекте. Модель данных построена с возможностью учета версий разработки АС, сведения о которых содержатся в сущности Version project. Связь проекта с руководителем реализуется посредством сущности Project manager, которая также связана с сущностью Employee, содержащей сведения о сотрудниках организации, выполняющей проект. Жизненный цикл состоит из стадий (сущность Stage), которые в свою очередь подразделяются на процессы (сущность Process). Сущность Task содержит сведения о задачах проекта, которые относятся к одному из типов задач (сущность)Task type. Типы задач вводятся с целью регламентирования работ, так как названия задач одного типа

могут значительно разниться, подчеркивая специфику проекта. При этом задачи относятся к одному из процессов (сущность Process). Процесс может быть представлен в виде одной задачи или декомпозирован на несколько отдельных задач. Сведения о команде, реализующей задачу, содержатся в сущности Task executor, которая сопоставляет сотрудников и задачи. В том случае, когда задача отдана на аутсорсинг, сущность Task executor связывает выполнение задачи со сторонней компанией (сущность Outsourcing). С целью сопоставления задач с процессами, которые в свою очередь сопоставляются с версией реализации проекта, введена сущность Process version. Сущность Stage version необходима для сопоставления этапов жизненного цикла с версией продукта.

Информация, представленная в разработанной структуре данных, предоставляет менеджеру проекта возможность осуществления мониторинга временных и качественных параметров как проекта в целом, так и отдельных его составляющих. Для этого в основные сущности введены поля date begin (дата начала выполнения работ), planned end date (плановая дата окончания работ), actual end date (фактическая дата окончания работ), quality (качество выполненных работ). При этом качество работ оценивается менеджером проекта, либо привлеченным экспертом.

Построение механизма прогноза рисков срыва сроков выполнения проекта возможно на основании средств искусственных нейронных сетей. Для этого необходимо из имеющихся реализованных проектов в базе данных (рис. 2) сформировать обучающую выборку в векторной форме, в которой входной вектор содержит информацию о датах начала работ, плановых сроках и фактических датах окончания работ, качестве полученного продукта декомпозированных по стадиям, процессам и версиям продукта. Выходной вектор содержит оценку прогнозируемого показателя. Так как даты входного вектора для версий проекта, стадий и процессов могут измеряться днями, неделями, месяцами и годами, то необходимо привести их к единому формату. При этом каждый обучающий пример – это отдельно реализованный проект, и нейронная сеть должна обобщить результаты. Примем за минимальную единицу времени 1 день. Если принять, что реализация проекта запланирована на N лет и в году 365 дней, то N×365 – общая длительность проекта, выраженная в днях. Следовательно, нормированное значение длительности одного дня составляет 1/(N×365). Таким образом, определив количество дней на реализацию каждого обособленного элемента процесса жизненного цикла и умножив его на нормированное значение одного дня формируется вектор, подающийся на вход нейронной сети.

Заключение

Проведен анализ моделей жизненного цикла, отечественных и зарубежных стандартов и методологий разработки информационных технологий. Рекомендательный характер данных документов, с одной стороны, осложняет процесс выбора подходящей методики управления жизненным циклом АС, с другой – позволяет выбрать наиболее приемлемый для конкретных условий вариант. Согласно большинству методологий, руководитель проекта создания АС должен иметь возможность оценивать риски несоблюдения сроков реализации проекта и этапов в частности, а также иметь средства оценки вклада каждого сотрудника, вовлеченного в реализацию проекта. Систематизированы основополагающие показатели крупных проектов разработки АС, формирующие требования к средствам управления жизненным циклом АС. На основании анализа существующих моделей жизненного цикла предложена интегрированная модель управления жизненным циклом АС, представленная в форме алгоритма, сочетающая достоинства известных моделей с учетом выделенных особенностей реализации крупных проектов. В основу интегрированной модели положены концепции спиральной модели, предоставляющей возможности учета версий проекта и анализа рисков, компонентно-ориентированной модели, предоставляющей возможность сокращения времени реализации этапов версий проекта посредством библиотека готовых решений, каскадной модели с промежуточным контролем, предоставляющей возможности своевременной корректировки результата версии и учета реалий процесса разработки. Интегрированная модель позволяет раз-

работчику организовать процесс управления жизненным циклом с учетом индивидуально заданного состава стадий жизненного цикла, декомпозированных на процессы с учетом уникальных особенностей проекта. На основании построенной интегрированной модели жизненного цикла и требований управления проектами, заключающихся в необходимости оценки рисков, разработана схема базы данных, содержащая временные и качественные параметры оценки как проекта в целом, так и отдельных составляющих.

Благодарности

Выполнено в рамках реализации комплексного проекта по созданию высокотехнологичного производства «Разработка методологии и инструментальных средств создания прикладных приложений, поддержки жизненного цикла информационно-технологического обеспечения и принятия решений для эффективного осуществления административно-управленческих процессов в рамках установленных полномочий», шифр «2017-218-09-187»; постановление Правительства Российской Федерации от 9 апреля 2010 г. № 218.

Список литературы
References

1. ISO/IEC/IEEE 15288:2015. Systems and software engineering – System life cycle processes. Date of introduction 15.05.2015.

2. ISO/IEC 12207:2017. Systems and software engineering – Software life cycle processes. Date of introduction 11.2017.

3. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. Дата введения 01.03.2012.

GOST R ISO/ IEC 12207-210. Information technology. System and software engineering. Software life cycle processes. Date of introduction 01.03.2012. (in Russian)

4. ISO/IEC TR 24748-2:2011. Systems and software engineering – Life cycle management. Part 2. Guide to the application of ISO/IEC 15288 (System life cycle processes), IDT). Date of introduction 09.2011.

5. ISO/IEC TR 24748-3:2011. Systems and software engineering – Life cycle management. Part 3. Guide to the application of ISO/IEC 12207 (Software life cycle processes). Date of introduction 09.2011.

6. ГОСТ Р 57102-2016/ISO/IEC TR 24748-2:2011. Информационные технологии. Системная и программная инженерия. Управление жизненным циклом. Часть 2. Руководство по применению ИСО/МЭК 15288. Дата введения 01.09.2017.

GOST R 57102-2016/ISO/IEC TR 24748-2:2011. Information technology. System and software engineering. Life cycle management. Part 2. Guide to the application of ISO/IEC 15288. Date of introduction 01.09.2017. (in Russian)

7. ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011. Информационные технологии. Системная и программная инженерия. Управление жизненным циклом. Часть 3. Руководство по применению ИСО/МЭК 12207 (Процессы жизненного цикла программных средств). Дата введения 01.06.2017.

GOST R 56923-2016/ISO/IEC TR 24748-3:2011. Information technology. System and software engineering. Life cycle management. Part 3. Guide to the application of ISO/IEC 12207. Date of introduction 01.06.2017. (in Russian)

8. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения. Дата введения 01.01.1992.

GOST 34.003-90. Information technology. Set of standards for automated systems. Automated systems. Terms and definitions. Date of introduction 01.01.1992. (in Russian)

9. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. Дата введения 01.01.1990.

GOST 34.602-89. Information technology. Set of standards for automated systems. Technical directions for automated system making. Date of introduction 01.01.1990. (in Russian)

10. ГОСТ 19.101-77. Единая система программной документации. Виды программ и программных документов. Дата введения 01.01.1980.

GOST 19.101-77. Unified system for program documentation. Types of programs and program documents. Date of introduction 01.01.1980. (in Russian)

11. ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. Дата введения 01.01.1992.

GOST 34.601-90. Information technology. Set of standards for automated systems. Automated systems. Stages of development. Date of introduction 01.01.1992. (in Russian)

12. ГОСТ 19.102-77 Единая система программной документации. Стадии разработки. Дата введения 01.01.1980.

GOST 19.102-77. Unified system for program documentation. Development stages. Date of introduction. 01.01.1980. (in Russian)

13. Royce W.W. 1970. Managing the development of large software systems: concepts and techniques. Proc. IEEE WESTCON, 22: 1–9.

14. Зараменских Е.П. 2014. Управление жизненным циклом информационных систем. Н., Сибпринт, 270.

Zaramenskih E.P. 2014. Life-cycle management of information systems. N., Sibprint, 270. (in Russian)

15. Сенник Ю.С., Гребенников И.Р. 2015. Жизненный цикл информационных систем. Системный анализ и прикладная информатика, 2: 4–9.

Sennik U.S., Grebennikov I.R. 2015. Life cycle of information systems. System analysis and applied informatics, 2: 4–9. (in Russian)

16. Boehm B.W. 1988. A spiral model of software development and enhancement. IEEE Computer, 5: 61–72.

17. Брауде Э. 2004. Технология разработки программного обеспечения. СПб., Питер, 655.

Braude Je. 2004. Software development technology. SPb., Piter, 655. (in Russian)

18. Oracle unified method (OUM). Oracle's full lifecycle method for deploying oracle-based business solutions. Oracle and/or its affiliates. USA, World Headquarters, 2014.

19. Rational unified process (RUP). Best practices for software. Development teams. Rational Software Corporation, 1998.

20. Гибкая методология разработки программного обеспечения. Microsoft Corporation. М.: Русская Редакция, 2008.

Flexible software development methodology. Microsoft Corporation. M.: Russkaja Redakcija, 2008. (in Russian)

21. Руководство к своду знаний по управлению проектами (Руководство PMBOK). 5-е издание. М. Олимп-Бизнес, 2018.

A guide to the knowledge of project management (Guide PMBOK). 5-e izdanie. M. Olimp-Biznes, 2018. (in Russian)

22. Ломакин В.В., Лифиренко М.В. 2014. Система поддержки принятия решений с автоматизированными средствами корректировки суждений экспертов. Научные ведомости БелГУ. Сер. История. Политология. Экономика. Информатика. 1(172): 114–120.

Lomakin V.V., Lifirenko M.V. 2014. Decision support system with means of expert judgment correction. Nauchnye vedomosti BelGU. Istoriya. Politologiya. Ekonomika. Informatika. [Belgorod State University Scientific Bulletin. History Political science Economics Information technologies]. 1(172): 114–120. (in Russian)

23. Ломакин В.В., Лифиренко М.В. Инструментальные средства поддержки жизненного цикла автоматизированных систем управления наружным освещением на основе экспертных методов принятия решений. Вестник БГТУ им. В.Г. Шухова, 5: 196–200.

Lomakin V.V., Lifirenko M.V. 2014. Instrumental decision-making tools for lifecycle support of outdoor lighting automated control systems. Vestnik BGTU im. V.G. Shuhova, 5: 196–200. (in Russian)