ПРОГРЕСИВНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

ПРОГРЕССИВНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

PROGRESSIVE INFORMATION TECHNOLOGIES

УДК 004.03:004.051

Брагина Т. И.[1], Табунщик Г. В.[2]

[1]Студент Запорожского национального технического университета
[2]Канд. техн. наук, доцент Запорожского национального технического университета

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИТЕРАТИВНЫХ МОДЕЛЕЙ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В статье выполнен обзор моделей разработки программных проектов, их жизненного цикла, достоинств и недостатков. Проведен анализ наиболее распространенных моделей: Rational Unified Process, Microsoft Solutions Framework, Personal Software Process / Team Software Process и Agile (eXtreme Programming, Crystal, Feature Driven Development). Сделаны выводы относительно факторов, влияющих на выбор модели, и предложены рекомендации по выбору соответствующей модели программного обеспечения в зависимости от характеристик проекта.

Ключевые слова: программный проект, стандарты качества, факторы выбора модели.

Методология управления проектами включает совокупность моделей, методов и программных продуктов, применяемых при разработке и реализации проектов различных классов и типов. Специфика проектов по разработке программных продуктов заключается в том, что результат разработки нематериален – это коллективные ментальные модели, записанные на языке программирования. Это приводит к тому, что большая часть проектов разработки программного обеспечения (ПО) завершается со срывами сроков, перерасходом бюджета, а часть проектов не завершается в принципе. Вышесказанное подтверждают исследования Standish Group [1], которые показали, что только 35 % проектов завершились в срок.

Одной из главных задач, напрямую влияющих на эффективность разработки ПО, является выбор модели процесса разработки. Проблемой является то, что не существует единого оптимального решения. Модель должна определяться для каждого конкретного проекта и может меняться в широком диапазоне, в зависимости от масштаба, новизны и критичности проекта, распределения участников, требований заказчика. На выбор модели также влияет возможность дальнейшей сертификации, что позволяет компании-разработчику получить преимущество перед конкурентами, привлечь новых заказчиков, повысить имидж организации.

Поэтому выбор модели процесса разработки ПО является очень актуальной проблемой при управлении проектом и коллективом разработчиков.

СРЕДСТВА ОЦЕНКИ КАЧЕСТВА МОДЕЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Существуют десятки различных подходов к обеспечению качества ПО, некоторые из которых (ГОСТ 28195-89, ГОСТ 19.102, DO-178B, ISO 9000 и др.)

постепенно устарели и содержат различные недостатки, например:

– недостаточная подробность стандарта, возможность самых различных его толкований в зависимости от представлений аудитора;

– неточность оценки качества процессов, задействованных при создании и внедрении программного обеспечения;

– отсутствие в стандарте механизмов, способствующих улучшению существующих процессов.

В начале 90-х годов прошлого столетия эксперты разработали более совершенные решения в области обеспечения качества, что привело к созданию целого ряда новых стандартов и методологий. Далее опишем три широко используемых в США и Европе стандарта – ISO 9001:2008, Capability Maturity Model (CMM) и ISO/IEC 15504 (SPICE), и государственные стандарты, используемые на территории стран СНГ.

Стандарт ISO 9001:2008. Стандарт ISO 9001:2008 был принят Международной организацией по стандартизации 14 ноября 2008 года. Он направлен на применение «процессного подхода» при разработке, внедрении и улучшении результативности системы менеджмента качества с целью повышения удовлетворенности потребителей путем выполнения их требований [2]. Для успешного функционирования организация должна определить и осуществлять менеджмент многочисленных взаимосвязанных видов деятельности. Деятельность, использующая ресурсы и управляемая с целью преобразования входов в выходы, может рассматриваться как процесс. Преимущество процессного подхода состоит в непрерывности управления, которое он обеспечивает на стыке отдельных процессов в рамках их системы, а также при их комбинации и взаимодействии.

В отличие от других, стандарт ISO 9001 имеет самую большую общность и зону применения. Однако ISO9001 затрагивает узкую область, оставляя за рамками своего внимания другие проблемы (не описываются связь между организационной поддержкой и реализацией проекта, роль измерения в системе управления качеством, необходимость постоянной оптимизации управления) процесса производства ПО и менеджмента этого процесса.

При применении в системе менеджмента качества процессный подход подчеркивает важность понимания и выполнения требований; необходимости рассмотрения процессов с точки зрения добавляемой ими ценности; достижения запланированных результатов выполнения процессов и обеспечения их результативности; постоянного улучшения процессов, основанного на объективном измерении.

С точки зрения многих западных и японских компаний, соответствие требованиям ISO 9001 – это крайне низкий уровень гарантий качества, однако это тот минимальный уровень, который дает возможность вхождения в рынок.

В Украине приказом Госпотребстандарта Украины от 22.06.2009 № 225 утвержден национальный стандарт ДСТУ ISO 9001:2009 «Системы управления качеством. Требования» (ISO 9001:2008, IDT) с введением в действие 01.09.2009.

Модель зрелости Capability Maturity Model for Software (SW-CMM). В середине 80-х годов XX столетия по заказу Министерства обороны США Институт программной инженерии (Software Engineering Institute – SEI), входящий в состав Университета Карнеги – Меллона, разработал стандарт SW-CMM [3] в качестве эталонной модели организации разработки ПО при реализации крупномасштабных программных проектов.

Данная модель определяет пять уровней зрелости процесса разработки ПО, каждый следующий уровень включает в себя все ключевые характеристики предыдущих: Начальный, Повторяемый, Определенный, Управляемый, Оптимизируемый.

В результате аттестации компании присваивается определенный уровень, который в дальнейшем может повышаться или (теоретически) понижаться. Данный стандарт предполагает возможность сертификации только одного процесса или подразделения организации.

Использование CMM затрудняют следующие проблемы: стандарт CMM является собственностью SEI и не является общедоступным; оценка качества процессов организаций может проводиться только специалистами, прошедшими специальное обучение и аккредитованными SEI; стандарт ориентирован на применение в относительно крупных компаниях; стандарт преувеличивает роль формальных описаний процессов; в стандарте не рассмотрены вопросы об отборе, повышении квалификации и сохранении компетентных сотрудников.

Модель зрелости Capability Maturity Model Integration (CMMI). После появления CMM стали разрабатываться специализированные модели зрелости для создания информационных систем, для процесса выбора поставщиков и некоторые другие. На их основе была разработана интегрированная модель CMMI.

CMMI уточняет и совершенствует предшествовавшие модели CMM, а также учитывает основные требования существующих международных стандартов в области менеджмента программных средств. С 2006 года действует версия CMMI – 1.2, которая

является модернизацией версии CMMI – 1.1. Предполагается выпустить в 2010 году новую, существенно улучшенную модель CMMI – 1.3.

Значительное внимание в CMMI уделяется процессам разработки и учету итераций при изменении требований заказчиков, их прослеживанию к функциям, компонентам, тестам и документам проекта. Существует два варианта модели CMMI – 1.1 для обеспечения непрерывного оценивания комплекса процессов в определенной области создания программных средств или для поэтапного оценивания и совершенствования зрелости предприятия, а также для организации жизненного цикла (ЖЦ) комплексов программ в целом. Стандарт регламентирует построение всех моделей по единой схеме [4].

Недостаток заключается в следующем. Поскольку модель CMMI предполагает строгое и формальное планирование, отслеживание проекта и полное документирование (иными словами – большие накладные расходы), эти модели вряд ли применимы к разработкам небольших систем в рамках маленьких проектов с небольшим количеством сотрудников, а также проектов с быстро меняющимися требованиями или с короткими циклами разработки.

Модели CMM/CMMI подходят для длинных проектов (программ) и больших организаций, которые создают сложные многокомпонентные системы с поставками новых версий каждые 9–12 месяцев и относительно большим числом сотрудников.

Стандарт SPICE. В 1991 году Международная организация по стандартизации инициировала работу по созданию единого стандарта оценки программных процессов. Стандарт получил имя SPICE (сокращение от **S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination – определение возможностей и улучшение процесса создания программного обеспечения). Официально стандарт называется «ISO/IEC 15504: Information Technology – Process Assessment» [5]. Международный стандарт версии ISO/IEC 15504 в настоящее время включает 6 частей, а 7-я часть в настоящее время находится в окончательной форме и начата работа по 8-й части.

ISO/IEC 15504 является эталонной моделью, которая определяет «Измерение процесса» и «Измерение зрелости (возможностей)» качества процесса.

«Измерение процесса» разделяет процессы на пять категорий: клиент – поставщик; инженерные; поддержки; управление; организация.

«Измерение зрелости (возможностей)» определяется с помощью девяти атрибутов процесса, сгруппированных на 5 уровнях зрелости:

1.1. Представление процесса (Process Performance);

2.1. Управление процессом (Performance Management);

2.2. Работа по управлению продуктами (Work Product Management);

3.1. Определение процесса (Process Definition);

3.2. Развертывание процесса (Process Deployment);

4.1. Измерение процесса (Process Measurement);

4.2. Контроль процесса (Process Control);

5.1. Инновация процесса (Process Innovation);

5.2. Оптимизация процесса (Process Optimization).

Каждый атрибут процесса оценивается по шкале: не достигнуто (0–15 %); частично достигнуто (15–50 %); в основном достигнуто (50–85 %); полностью достигнуто (85–100 %).

ISO/IEC 15504 может использоваться в двух контекстах: для совершенствования качества процессов разработки и для определения качества.

Достоинства ISO / IEC 15504:

– ISO/IEC 15504 является общедоступным национальным стандартом;

– данный стандарт пользуется поддержкой международного сообщества;

– SPICE возможно использовать и в небольших компаниях – об этом свидетельствуют результаты работы проекта SPIRE [6].

Недостатки SPICE [4]:

– ISO/IEC 15504 не является столь успешным, как CMM, т. к. CMM активнее спонсируется, был создан раньше и впоследствии был заменен CMMI, которая включает в себя множество идей ISO / IEC 15504, а также сохраняет преимущества CMM;

– по ISO/IEC 15504 не выдаются сертификаты соответствия. Он не является нормативным, хотя соответствует современным международным требованиям к качеству.

Система государственных стандартов. В настоящее время на территории России и других стран СНГ действуют старые ГОСТы 19-й и 34-й серий и более новый ГОСТ Р ИСО МЭК 122207 (в России).

ГОСТ 19 «Единая система программной документации» и ГОСТ 34 «Стандарты на разработку и сопровождение автоматизированных систем» ориентированы на последовательный подход к разработке ПО. Разработка в соответствии с этими стандартами проводится по этапам, каждый из которых предполагает выполнение строго определенных работ, и завершается выпуском достаточно большого числа весьма формализованных и обширных документов. Таким образом, строгое следование этим гостам не только приводит к водопадному подходу, но и требует очень высокой степени формализованности разработки.

На основе этих стандартов разрабатываются программные системы по госзаказам в странах СНГ.

ГОСТ 12207 (перевод на русский язык ISO/IEC 12207), в отличие от стандартов 19-й и 34-й серий, описывает разработку ПО как набор основных и вспомогательных процессов, которые могут действовать от начала и до завершения проекта. Модель ЖЦ может выбираться исходя из особенностей проекта. Таким образом, этот ГОСТ явно не запрещает применение итеративного подхода, но и явно не рекомендует его использование. ГОСТ 12207 также более гибок в части требований к формальности процесса разработки. В нем содержатся только указания на необходимость документирования основных результатов процессов, но нет перечней требуемых документов и указаний относительно их содержания. Таким образом, ГОСТ 12207 допускает итеративную и менее формализованную разработку ПО.

ХАРАКТЕРИСТИКА МОДЕЛЕЙ С ИТЕРАТИВНЫМ ПРОЦЕССОМ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Под моделью ПО обычно понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении ЖЦ. Далее рассмотрены наиболее распространенные модели, ориентированные на итеративный процесс разработки: Rational Unified Process, Microsoft Solutions Framework, Personal Software Process / Team Software Process и Agile(eXtreme Programming, Crystal, Feature Driven Development).

Методология Rational Unified Process (RUP). Унифицированный процесс RUP [7] был разработан Филиппом Крачтеном (Philippe Kruchten), Иваром Якобсоном (Ivar Jacobson) и другими сотрудниками компании «Rational Software» в качестве дополнения к языку моделирования Unified Modeling Language (UML). Особенностью RUP является то, что в результате работы над проектом создаются и совершенствуются модели ПО. Вместо создания громадного количества бумажных документов RUP опирается на разработку и развитие семантически обогащенных моделей, всесторонне представляющих разрабатываемую систему.

RUP, в отличие от большинства других методологий, позволяет в широком диапазоне выбирать степень формализации и итеративности процесса разработки в зависимости от особенностей проектов и разрабатывающей организации. RUP наиболее часто используют компании с большим количеством разработчиков (свыше 40–50 человек).

ЖЦ проекта RUP состоит фаз и дисциплин. Фазы RUP: начало (Inception), проектирование (Elaboration), разработка (Construction) и внедрение (Transition). Последовательность этих фаз фиксирована, но число итераций, необходимых для завершения каждой фазы, определяется индивидуально для каждого конкретного проекта [8]. Каждая фаза состоит из дисциплин: «Бизнес-моделирование» (Business modeling); «Управление требованиями» (Requirements); «Анализ и Проектирование» (Analysis and Design); «Реализация» (Implementation); «Тестирование» (Test); «Развертывание» (Deployment), «Управление проектом» (Project management); «Управление изменениями» (Change management); «Среда» (Environment).

В RUP все дисциплины выполняются практически во всех фазах ЖЦ программных средств (ПС). Однако, в зависимости от фазы, меняются текущие цели проекта и соотношение между объемами работ, соответствующих различным дисциплинам.

Методология RUP использует программные решения Rational Application Developer, Rational Rose, Rational Asset Manager, Rational Software Architect, Rational Software Modeler, Rational Suite, Rational Requisitepro, Rational Clearquest, Rational Clearcase и другие [9].

Достоинства RUP:

– очень широкий диапазон решений в части формализации процесса разработки;

– можно использовать как основу для водопадного стиля разработки, так и в качестве гибкого процесса;

– снижение основных рисков заказчика и разработчика;

– экономия ресурсов за счет автоматизации регрессионного тестирования;

– улучшение качества ПО за счет многократных проверок изменений;

– улучшение качества тестирования за счет использования современных технологий.

Использование RUP гарантирует получение 3-го уровня CMM.

Недостатком является недостаточный уровень формализма, который может приводить к несогласованности решений, принимаемых участниками проекта, к непродуктивным затратам ресурсов на переработку кода и на повторное решение типовых проблем.

Модель Microsoft Solutions Framework (MSF). В 1994 году Microsoft выпустила MSF – набор концепций и рекомендуемых моделей, которые позволяют разрабатывать и внедрять распределенные информационные системы масштаба предприятия на основе технологий и инструментальных средств фирмы Microsoft. MSF базируется на практических результатах организации распределенных вычислений и применения

клиент-серверных технологий, полученных как фирмой Microsoft, так и ее партнерами и заказчиками [10].

Процесс разработки в соответствии с MSF является итеративным и включает в себя следующие основные фазы: разработка концепции (единого видения), планирование, разработка, стабилизация (тестирование), внедрение. Каждая фаза цикла заканчивается главной вехой (контрольной точкой). Соответственно главные вехи будут иметь названия: концепция продукта утверждена, планы продукта утверждены, разработка завершена, готовность решения утверждена, внедрение завершено. Веха является точкой синхронизации достигнутых результатов и ожиданий заказчика, а также анализа проектной среды. В решении о закрытии очередной фазы должны принимать участие ответственные представители всех ролевых кластеров (разработка, тестирование, внедрение, управление проектом и пр.) [10].

MSF предлагает достаточно нестандартные подходы к организационной структуре, распределению ответственности и принципам взаимодействия внутри команды. MSF предлагает разбиение больших команд (более 10 человек) на малые многопрофильные группы направлений (feature teams). Эти малые коллективы работают параллельно, регулярно синхронизируя свои усилия [11].

Microsoft предлагает инструменты для реализации MSF: MSF for Agile Software Development, MSF for CMMI Process Improvement, Visual Studio Team System (Team Suite; Team Edition for Database Professionals; Team Suite for Software Architects; Team Suite for Software Developers; Team Suite for Software Testers; Team Foundation Server; Team Test Load Agent).

Применение MSF позволит сертифицировать организацию на 2–3-й уровень CMM.

Достоинствами MSF являются:

– систематизация и структуризация информации в форме базы знаний;

– нестандартные подходы к организационной структуре, распределению ответственности и принципам взаимодействия внутри команды;

– MSF легко масштабируется – минимальный размер проектной группы в MSF-проекте – 3 человека, но применять данную методологию можно для коллективов и в десятки, сотни, тысячи человек;

– MSF не навязывает использование каких-либо конкретных инструментов и программных средств.

Недостатки MSF: не описывает детально важнейшие роли заказчика и пользователя; не рассматриваются методы управления группой проектов.

Personal Software Process / Team Software Process (PSP/TSP). В конце 90-х годов SEI разработал модель PSP/TSP, основанную на модели зрелости CMM [12]. PSP определяет требования к компетенциям разработчика. Согласно этой модели каждый программист должен уметь: учитывать время, затраченное на работу над проектом; учитывать найденные дефекты; классифицировать типы дефектов; оценивать размер задачи; осуществлять систематический подход к описанию результатов тестирования; планировать программные задачи; распределять их по времени и составлять график работы; выполнять индивидуальную проверку проекта и архитектуры; осуществлять индивидуальную проверку кода; выполнять регрессионное тестирование.

PSP включает 4 процесса, которые последовательно добавляются разработчиком: PSP0 – базовый собственный процесс (The Baseline Personal Process); PSP1 – планирование (Personal Planning Process); PSP2 – управление качеством (Personal Quality Management); PSP3 – циклический процесс (Cyclic Personal Process).

TSP делает ставку на самоуправляемые команды численностью 3–20 разработчиков. Команды должны установить собственные цели, составить свой процесс и планы, отслеживать работу, поддерживать мотивацию и максимальную производительность.

Методология является эффективной, может обеспечить существенные преимущества и стать моделью для любой организации, заинтересованной в совершенствовании своих процессов разработки. Совместное использование организациями технологий PSP и TSP способно сформировать процесс совершенствования в рамках модели зрелости процессов разработки ПО и сделать нормой в организации пятый уровень CMM.

PSP/TSP – это чистая методология по обеспечению качества проектов, рассчитанная на использование UML при проектировании, и любые CASE-средства, в частности они ориентированы на использование продуктов компании Microsoft.

Достоинства: лучшее планирование времени и бюджета; управление качеством продукта; уменьшение времени разработки; аккуратный сбор метрик; оценка времени работы через оценку объема артефактов; использование исторических данных в планировании; раннее обнаружение дефектов.

Недостатки: сложность в учете рисков, учете недооценки сложности.

Гибкая методология разработки ПС (Agile). В феврале 2001 года 17 разработчиков различных методологий собрались для того, чтобы попытаться обнаружить что-нибудь общее в своих подходах. Результатом стал Манифест гибкой разработки [13]. Тогда же появился термин Agile («гибкий, шустрый»), объединяющий все методологии.

Agile – гибкая методология разработки, это концептуальный каркас, в рамках которого выполняется разработка программного обеспечения. Основная идея всех гибких моделей заключается в том, что применяемый в разработке ПО процесс должен быть адаптивным. Они декларируют своей высшей ценностью ориентированность на людей и их взаимодействие, а не на процессы и средства [7].

CASE-средства, которые могут помочь внедрить и улучшить процесс создания ПО на базе Agile: XPArchitect; Borland ALM(в частности Silk).

Рассмотрим более детально несколько гибких методологий: eXtreme Programming, Crystal и Feature Driven Development.

1. Экстремальное программирование (eXtreme Programming, или XP) – методология, разработанная Кентом Беком (Kent Beck) [12], Уордом Каннингемом (Ward Cunningham) и Роном Джеффрисом (Ron Jeffries), является сегодня наиболее известной из гибких методологий. XP проповедует коммуникабельность, простоту, обратную связь. Она описывается как набор практик: игра в планирование, короткие релизы, метафоры, простой дизайн, переработки кода (refactoring), разработка «тестами вперед», парное программирование, коллективное владение кодом, 40-часовая рабочая неделя, постоянное присутствие заказчика и стандарты кода. При использовании XP тщательное предварительное проектирование ПО заменяется, с одной стороны, постоянным присутствием в команде заказчика, а с другой – регулярными переработками кода (так называемый рефакторинг). Основой проектной документации считается тщательно прокомментированный код. Очень большое внимание в методологии уделяется тестированию.

ЖЦ проекта в XP [8] состоит из последовательности релизов. Каждый релиз – это полноценная версия продукта, которую может использовать заказчик, и содержащая дополнительную функциональность по сравнению с предыдущим релизом. Релиз появляется в результате одной или нескольких итераций, длящихся от одной до четырех недель.

В XP не рекомендуется тратить много времени на планирование; сам процесс планирования называется игрой (planning game). Подробный план составляется только на очередную итерацию и ближайшие один-два релиза.

Недостатки:

– XP могут эффективно использоваться только в команде опытных разработчиков;

– команду нельзя разбивать на несколько частей, возможный размер команды ограничен числом в 10–15 человек, т. к. большую роль в XP играет прямое общение;

– не всегда можно обеспечить постоянное присутствие представителя заказчика в проектной команде;

– нежелательно использовать XP в проекте с фиксированной ценой.

Несмотря на все перечисленные ограничения, XP может показать исключительную эффективность благодаря крайне низким накладным расходам.

2. Crystal – семейство методологий, определяющих необходимую степень формализации процесса разработки в зависимости от количества участников и критичности задач. Алистэр Коуберн (Alistair Cockburn) в начале 90-х по заказу от компании IBM написал работу на тему методологии разработки ПО [14]. При этом его подход существенно отличался от подхода большинства других методологов. Его теории основаны не только на личном опыте, но и на постоянных исследованиях других проектов и процессов.

Степень важности нарастает по вертикальной оси. Величина команды нарастает по горизонтальной оси. В результате получается семейство методологий. Чем ниже критичность и чем меньше команда, тем более «легкую» методологию нужно использовать. Самой легкой из всего семейства является методология Crystal Clear.

Главные принципы данной методологии:

– вся команда разработчиков (до 6 человек) находится в одном помещении;

– частые поставки продукта позволяют выработать «ритм» проекта;

– обмен информацией с реальными пользователями позволяет быстро получать обратную связь и ликвидировать недостатки на ранних стадиях;

– средства контроля версий кода обеспечивают коллективное владение кодом.

Семейство методологий Crystal построено на итеративной разработке. Продолжительность итерации может изменяться в пределах от 1 до 4 месяцев. Чем меньше итерация, тем лучше.

Важной особенностью Crystal является непрерывная настройка методологии. Это позволяет постепенно адаптировать ее для конкретной команды и конкретного проекта. Ни в одной другой методологии настройке не уделяется такого внимания.

Достоинством Crystal Clear является то, что она максимально проста в использовании, требует минимальных усилий для внедрения, ориентирована на человеческие привычки, описывает естественный порядок разработки ПО.

Недостатки:

– уступает XP по производительности;

– очень сложно заранее предсказать, какие промежуточные продукты необходимы.

3. Feature Driven Development(FDD). Эта методология была разработана Джеффом Де Люка (Jeff De Luca) и Питером Коадом (Peter Coad) в 1997 году [15]. Это функционально-ориентированная разработка, которая оперирует понятием функции или свойства (feature) системы, достаточно близким к понятию сценария использования, применяемому в RUP. Отличие – это дополнительное ограничение: «каждая функция должна допускать реализацию не более чем за две недели».

FDD включает пять процессов, причем последние два повторяются для каждой функции [15]: разработка общей модели; составление списка необходимых функций системы; планирование работы над каждой функцией; проектирование функции; конструирование функции. Работа над проектом предполагает частые сборки и делится на итерации, каждая из которых реализуется с помощью определенного набора функций.

Достоинства FDD: делает процесс легче; дает возможность планирования и предварительного проектирования, создания детального дизайна, есть управление приоритетами и возможна трассировка требований; рассчитан на работу в больших командах.

Но в тоже время, недостаток документации может значительно увеличивать стоимость последующего сопровождения продукта, поскольку внесение каких-либо изменений в него потребует очень больших усилий.

СРАВНЕНИЕ МОДЕЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Поскольку каждая из рассмотренных моделей обладает своими достоинствами и недостатками, не существует единой оптимальной модели ПО. Она должна определяться для каждого конкретного проекта и может меняться в очень широком диапазоне. Основными факторами, влияющими на выбор модели и степень формализации процесса, являются [16]:

– масштаб проекта – чем больше людей участвует в проекте, тем более формально он должен вестись;

– распределение участников – чем компактнее расположены участники, чем легче им общаться между собой, тем менее формализованным может быть проект;

– критичность проекта – проекты, в которых ошибки могут приводить к тяжелым последствиям, вплоть до гибели людей, должны проводиться гораздо более формально, чем те проекты, в которых ошибки приводят только к временным неудобствам;

– новизна проекта – если разработчики используют новые технологии и мало знакомы с предметной областью, то следует более тщательно, более формально прорабатывать проектные решения. Однако если над проектом работает небольшая группа высококвалифицированных программистов, более предпочтительным будет тщательное оформление уже отработанных решений перед их тиражированием;

– требования заказчика – существенно различаются в зависимости от отрасли и статуса организации-заказчика;

– ожидаемая долговечность проекта – если разрабатываемое для конкретного заказчика ПО планируется через пару лет заменить новым, то вряд ли имеет смысл тратить много сил на документацию. Если же срок жизни проекта предполагается достаточно длинным, то без хорошей документации не обойтись.

Опираясь на данные факторы, будем классифицировать модели итеративной разработки ПО по «весу» – количеству формализованных процессов (большинство процессов или только основные) и детальности их регламентации. Чем больше процессов документировано, чем более детально они описаны, тем больше «вес» модели. Достоинства и недостатки модели в зависимости от ее веса рассмотрены в табл. 1.

Таблица 1. Характеристика моделей процессов разработки ПО

Вес	Модели	Достоинства	Недостатки
Тяжелый	RUP, MSF (для CMMI)	Процессы рассчитаны на среднюю квалификацию исполнителей. Невысокие требования к стабильности команды. Нет ограничений на распределение участников. Отсутствуют ограничения по объему и сложности выполняемых проектов. Используются для проектов с высокими рисками. Легче получить высокий уровень сертификации стандартов качества	Требуют существенной управленческой надстройки. Более длительные стадии анализа и проектирования. Более формализованные коммуникации. Большие затраты на документацию длительного процесса сопровождения ПО
Легкий	MSF (для Agile), Agile, PSP/ TSP	Меньше непроизводительных расходов, связанных с управлением проектом, рисками, изменениями, конфигурациями, сопровождением ПО. Упрощенные стадии анализа и проектирования, основной упор на разработку функциональности, совмещение ролей. Неформальные коммуникации	Используется для проектов с невысокими рисками и небольшого масштаба и сложности. Участники проекта должны располагаться компактно. Плохо подходят для новых проектов. Требуют более квалифицированной и стабильной команды. Сложнее получить высокий уровень сертификации стандартов качества

Таблица 2. Сравнение моделей разработки программных проектов

Модели / Параметры	RUP	MSF	PSP/TSP	Agile		
				XP	Crystal Clear	FDD
Содержание	Абстрактный общий процесс, на основе которого организация должна создать конкретный специализированный процесс, ориентированный на ее потребности	Согласованный набор концепций, моделей и правил, описывающих управление людьми и процессами в процессе разработки решения; опирается на практический опыт Microsoft	Набор требований к компетенциям разработчика, делает ставку на самоуправляемые команды	Набор практик: короткие релизы, простой дизайн, переработки кода, разработка «тестами вперед», коллективное владение кодом, постоянное присутствие заказчика и стандарты кода	Итеративная инкрементная разработка; автоматическое регрессионное тестирование; состав документации определяется участниками проекта	Функционально-ориентированная разработка, которая оперирует понятием функции или свойства системы
Команда	Свыше 40–50 человек	Группы по 3–10 человек	3–20 разработчиков	До 10–15 человек	До 6 человек	20–30 человек
Вес	Тяжелый	Может быть и тяжелым, и легким	Легкий	Легкий		
Используемые стандарты	CMMI, ISO, SPICE	CMMI, ISO, SPICE	CMMI, SPICE	CMMI, SPICE	CMMI, SPICE	CMMI, SPICE
Достоинства	1. Снижение основных рисков заказчика, и разработчика. 2. Экономию ресурсов за счет автоматизации регрессионного тестирования. 3. Улучшение качества ПО за счет многократных проверок. 4. Улучшение качества тестирования за счет использования современных технологий.	1. Систематизация и структуризация информации в форме базы знаний. 2. Нестандартные подходы к организационной структуре, распределению ответственности и принципам взаимодействия внутри команды	1. Лучшее планирование времени и бюджета. 2. Управление качеством продукта. 3. Уменьшение времени разработки. 4. Аккуратный сбор метрик. 5. Оценка времени работы через оценку объема артефактов. 6. Раннее обнаружение дефектов	1. Крайне низкие накладным расходы. 2. Процесс может показать исключительную эффективность	1. Максимально проста в использовании; 2. Требует минимальных усилий для внедрения; 3. Ориентирована на человеческие привычки; 4. Описывает естественный порядок разработки ПО	1. Дает возможность планирования и предварительного проектирования, создания детального дизайна. 2. Есть управление приоритетами и возможна трассировка требований
Недостатки	Несогласованность решений, непродуктивные затраты ресурсов на переработку кода и на повторное решение типовых проблем в случае недостаточного уровня формализма	Не описывает детально важнейшие роли заказчика и пользователя; не рассматриваются методы управления группой проектов	Проблемы при учете рисков, сложности	Рассчитан только на команду опытных разработчиков, не разбитую на несколько частей; проблемы разрешения нетехнологических рисков	Уступает XP по производительности; очень сложно заранее предсказать, какие промежуточные продукты необходимы	Недостаток документации; значительно увеличивается стоимость последующего сопровождения продукта
CASE-средства	IBM Rational	Visual Studio Team System	Любые CASE-средства, в частности продукты компании Microsoft	XPArchitect; Borland ALM (в частности Silk)		

В табл. 2 приводится сравнение моделей, поддерживающих итеративную разработку, по содержанию, весу, ориентации на количество разработчиков, используемым стандартам для обеспечения качества, достоинствам и недостаткам.

Если организация достаточно велика (больше десятка разработчиков) и выполняет проекты для различных отраслей или для систем разного уровня критичности – желательно выбирать для каждого проекта свой оптимальный уровень формализации и модель ПО.

Организациям с большим количеством людей или организациям-компаньонам, проводящим разработку совместно, но разделенным географически, следует выбирать RUP. В случае управления проектом с большим количеством данных целесообразно использовать MSF, т. к. систематизация и структуризация информации в форме базы данных позволит упростить работу с ними.

Если команда разработчиков состоит из 1–20 человек, которые расположены компактно, следует выбрать любую из гибких методологий или PSP/TSP. Последнюю желательно использовать, если ошибки в проекте не являются критичными, а команда организации состоит из небольшого количества разработчиков и является самоуправляемой.

Для критических и малознакомых проектов необходимо выбирать MSF либо RUP, для некритических подойдут PSP/TSP и Agile. Гибкие методологии также можно успешно применить при разработке новых технологий.

Если заказчик настаивает на личном участии в процессе разработки, следует выбирать гибкие методологии, в частности XP. Если же для контроля используется строгая отчетность, выбор следует остановить на RUP, MSF.

В случае повторного использования ПО целесообразно использовать RUP.

ВЫВОДЫ

Учитывая вышеизложенное, можно сделать следующие выводы:

1. Каждая из рассмотренных моделей предназначена для разработки определенного класса ПО с разными требованиями к командам разработчиков. Модели ориентированы на соответствующее ПО, однако эффективны и при использовании в качестве независимых методологий.

2. Для обеспечения качества ПО широко используются стандарты ISO9001, CMM, ISO/IEC 15504. На использование стандартов CMM/CMMI и ISO/IEC не влияет используемая модель разработки ПО. Однако стандартизация процессов, ориентированных на Agile методологию, с использованием стандарта ISO9001 практически невозможна.

3. Для выбора модели итеративного процесса разработки ПО одним из наиболее эффективных является показатель «вес модели», который учитывает масштаб, новизну и критичность проекта, распределение участников и требования заказчика.

СПИСОК ЛИТЕРАТУРЫ

1. *Rubinstein D.* Standish Group Report: There‘s Less Development Chaos Today [Электрон. ресурс]. – Режим доступа: http://www. sdtimes. com/content/article. aspx? ArticleID=30247.
2. ISO 9001:2008 [Электрон. ресурс]. – Режим доступа: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail. htm? csnumber=46486.
3. Capability Maturity Model for Software, Version 1.1 (CMU/SEI-93-TR-24) [Текст] / Paulk, Mark C., and others, Software Engineering Institute, Carnegie Mellon University. – Pittsburgh, Pa. : SEI, CMU, 1993. – 88 с.
4. *Липаев В. В.* Модели зрелости программной инженерии – CMMI. Содержание и применение [Текст] / В. В. Липаев // Информационный бюллетень. – 2006. – № 6(157). – С. 1–15.
5. ISO/IEC 15504 [Электрон. ресурс]. – Режим доступа: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail. htm? csnumber=50519.
6. Software Process Improvement in Regions of Europe [Электрон. ресурс]. – Режим доступа: http://www.cse.dcu.ie/spire.
7. *Bragina T., Tabunshchik G.* Comparative Analysis of Software Development Models for Electro-technical Systems // Proc. of Int. Conf. on Modern Problem of Radio Engineering, Telecommunications and Computer Science TCSET’2010, February 19–23, 2010, Lviv–Slavsko, Ukraine. – P. 347.
8. *Рахимбердиев А.* Современные процессы разработки программного обеспечения [Текст] / А. Рахимбердиев // RSDN Magazine. – 2006 – № 4. – С. 3–10.
9. IBM [Электрон. ресурс]. – Режим доступа: http://www-142.ibm.com/software/products/ru/ru/category/rational/SW700.
10. *Keeton M.* Microsoft Solutions Framework (MSF) : A Pocket Guide (Paperback) [Текст] / Keeton M. – Wan Haren Publishing, 2005. – 112 p.
11. *Turner M. S. V.* Microsoft Solutions Framework Essentials: Building Successful Technology Solutions [Текст] / Turner M. S. V. – Microsoft Press 1 ed., – 2006. – 336 p.
12. *Брагина Т. И., Табунщик Г. В.* Классификация моделей итеративной разработки программного обеспечения // Материалы всеукраинской научно-практической конференции «Системный анализ. Информатика. Управление», САІУ-2010, Март 04–05, 2010, Запорожье, Украина. – С. 23–25.
13. Agile Manifesto [Электрон. ресурс]. – Режим доступа: http://agilemanifesto.org.
14. *Cockburn A.* Surviving Object-Oriented Projects [Текст] / A. Cockburn. – Addison Wesley, 1997. – 272 p.
15. *Палмер С. Р.* Практическое руководство по функционально-ориентированной разработке ПО [Текст] / Стивен Р. Палмер, Джон М. Фелсинг. – Вильямс, 2002. – 304 с.
16. *Архипенков С.* Руководство командой разработчиков программного обеспечения. Прикладные мысли [Текст] / С. Архипенков. – М. : Москва, 2008. – 80 с.

Надійшла 24.02.2010
Після доробки 22.03.2010

Брагіна Т. І., Табунщик Г. В.

ПОРІВНЯЛЬНИЙ АНАЛІЗ ІТЕРАТИВНИХ МОДЕЛЕЙ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті виконано огляд моделей розробки програмних проектів, їх життєвого циклу, переваг і недоліків. Проведено аналіз найбільш поширених моделей: Rational Unified Process, Microsoft Solutions Framework, Personal Software Process / Team Software Process і Agile (eXtreme Programming, Crystal, Feature Driven Development). Зроблено висновки щодо факторів, що впливають на вибір моделі, та запропоновано рекомендації з вибору відповідної моделі програмного забезпечення в залежності від характеристик проекту.

Ключові слова: програмний проект, стандарти якості, фактори вибору моделі.

Bragina T. I., Tabunshchyk G. V.

COMPARATIVE ANALYSIS OF ITERATIVE SOFTWARE DEVELOPMENT MODEL

The authors have made a review of software projects development models, their life cycle, their advantages and disadvantages. The most common models were analyzed, such as Rational Unified Process, Microsoft Solutions Framework, Personal Software Process / Team Software Process and Agile (eXtreme Programming, Crystal, Feature Driven Development). Conclusions were drawn on the factors which influence the model choice, and recommendations were proposed for appropriate software model selection depending on project characteristics.

Ключові слова: software project, quality standards, factors for model selection.

УДК 621.391

Высочина О. С.[1], Шматков С. И.[2], Салман Амер Мухсин[3]

[1]Аспирант Харьковского национального университета им. В.Н. Каразина
[2]Канд. техн. наук, заведующий кафедрой Харьковского национального университета им. В.Н. Каразина
[3]Аспирант Харьковского национального университета радиоэлектроники

АНАЛИЗ СИСТЕМ МОНИТОРИНГА ТЕЛЕКОММУНИКАЦИОННЫХ СЕТЕЙ

Рассматриваются наиболее известные системы мониторинга телекоммуникационной сети, представлен их сравнительный анализ, сформулированы общие требования и синтезирована общая архитектура подобных систем. Предлагаемые решения не способны прогнозировать состояние телекоммуникационной сети. Поэтому в такие системы необходимо включать дополнительные модули обработки статистической информации.

Ключевые слова: диагностика состояния телекоммуникационных сетей, нейронные сети, мониторинг состояния, адаптивное управление.

1. ПОСТАНОВКА ЗАДАЧИ

Телекоммуникационная отрасль переживает сегодня значительные преобразования, переходу от традиционных сетей с коммутацией каналов к пакетной передаче данных сопутствует лавинообразный рост предоставляемых абонентам услуг. При этом в условиях постоянного повышения сложности информационных и телекоммуникационных систем надежность телекоммуникационной сети и качество предоставляемых сервисов приобретают особую важность. Современная телекоммуникационная инфраструктура представляет собой сложную гетерогенную сеть, включающую телекоммуникационное, серверное и программное обеспечение различных производителей, работающее в различных стандартах и под управлением различного программного обеспечения. Сложность и масштабность сетевой инфраструктуры предопределяют высокий уровень автоматизированных средств мониторинга и управления, которые должны использоваться для обеспечения надежной работы сети.

Целью создания системы мониторинга телекоммуникационной инфраструктуры является [1–2]:

– обеспечение высокой скорости обработки запросов пользователей на предоставление требуемых информационных ресурсов и сервисов;

– предоставление программно-аппаратных средств по управлению информационными и телекоммуникационными ресурсами;

– создание эффективной службы диагностики и своевременного оповещения для предупреждения аварийных ситуаций и повышения отказоустойчивости телекоммуникационных систем;

– выполнение сбора, обработки, хранения и отображения полной информации о состоянии всех компонентов телекоммуникационной и информационной