УДК 004.78

И.В. ШОСТАК, М.А. ДАНОВА

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

ПОДХОД К ФОРМИРОВАНИЮ СТРАТЕГИИ ПРИМЕНЕНИЯ МЕТОДОЛОГИЙ РАЗРАБОТКИ С УЧЕТОМ ОСОБЕННОСТЕЙ ПРОГРАММНЫХ ПРОЕКТОВ

Рассмотрены существующие методологии разработки программных проектов, а также приведены сравнительные характеристики наиболее эффективных и часто применяемых методологий, таких как RUP, MSF, XP, Crystal и других. Приведена классификация типов программных проектов по масштабу, надежности, влиянию на безопасность и многих других. Предложен подход к формированию стратегии применения методологий разработки программного обеспечения для различных типов программных проектов, который позволяет установить однозначное соответствие между конкретными типами программных проектов и рассматриваемыми методологиями.

***Ключевые слова**: методология разработки ПП, прикладные программы, программный проект, классификация ПО, стратегия, масштаб проекта.*

Введение

На сегодняшний день, как во всем мире, так и в Украине спрос на программное обеспечение (ПО) постоянно увеличивается. Как средство автоматизации и улучшения работ, оно используется практически во всех сферах человеческой деятельности (экономика, медицина, бизнес, коммерция, промышленность и т. д.).

Разработчики ПО ориентированы на реализацию разных предметных областей, начиная от операционных систем и заканчивая прикладными бизнес-системами [1, 7, 11]. Знания, используемые ими, как правило, отличаются разнообразием и являются неполными, несогласованными и разнородными. Зачастую общий проект представляет собой смесь краткосрочных решений, при этом единого плана не существует, то есть разработка ПО представляет собой деятельность, которую возможно охарактеризовать фразой "code and fix" (кодирования и исправления ошибок).

Описанная выше ситуация как правило возникает при создании небольшой программы, однако при разработке крупного ПО количество ошибок будет неизбежно расти, а исправлять их будет все труднее [3, 4]. При подобной организации тестирования и исправления ошибок адекватное планирование просто невозможно, и, как следствие, нарушение планов по созданию программы.

Как известно [1, 2, 4, 11], главной целью процесса проектирования и разработки ПО является создание программного продукта, обладающего высоким качеством, в приемлемые сроки в рамках прогнозируемого бюджета. Достичь этого можно только при правильной организации работ по созданию ПО, применяя различные методологии.

Любая методология организует взаимодействие всех вовлеченных в проект сторон в единый процесс с целью снижения риска получения результата (продукта), не удовлетворяющего поставленным требованиям [2, 5].

Методология превращает создание программного продукта в упорядоченный процесс [8 – 12], с помощью которого можно сделать работу программиста более прогнозируемой и эффективной.

На сегодняшний день задачи управления процессом разработки ПО относятся к классу слабоформализованных, слабоструктурированных задач с расплывчатыми ограничениями, неполными и нечеткими данными, сильно зависящими от изменений внешней среды и субъективных предпочтений лица, принимающего решение [2, 5 – 8] таким образом *цель данной статьи состоит в изложении подхода к применению методологий разработки для различных типов программных проектов.*

1. Постановка задачи

Известен портфель программных проектов, разрабатываемых в типовой IT-Компании, а также набор методологий разработки ПО.

Необходимо получить рациональную стратегию применения методологий для различных типов программных проектов.

Решение указанной задачи предполагает разработку классификации типов программных проектов, а также проведение сравнительного анализа особенностей наиболее часто применяемых методологий

проектирования. Реализация последней задачи послужит основанием к созданию классификации методологий разработки ПО. При этом будет установлено однозначное соответствие между конкретными типами программных проектов и рассматриваемыми методологиями. На заключительном этапе решения рассматриваемой задачи путем композиции этапов различных методологий будет создана эффективная процедура, формализующая стратегию разработки ПО.

2. Обзор существующих методологий разработки ПО

Рассмотрим наиболее известные и эффективные методологи разработки ПО.

Rational Unified Process (RUP) **–** методология разработки программного обеспечения, созданная компанией Rational Software.

RUP обеспечивает строгий подход к распределению задач и ответственности внутри организации-разработчика. Его предназначение заключается в том, чтобы гарантировать создание точно в срок и в рамках установленного бюджета качественного ПО, отвечающего нуждам конечных пользователей.

Особенностью RUP является то, что в результате работы над проектом создаются и совершенствуются модели. RUP опирается на разработку и развитие семантически обогащенных моделей, всесторонне представляющих разрабатываемую систему. В качестве языка моделирования в общей базе знаний используется Unified Modeling Language (UML), являющийся международным стандартом. Стандартный язык моделирования, используемый всеми членами группы, делает понятными для всех описания требований, проектирование и архитектуру системы.

RUP использует итеративную модель разработки. В конце каждой итерации проектная команда должна достичь запланированных на данную итерацию целей, создать или доработать проектные артефакты и получить промежуточную, но функциональную версию конечного продукта. Итеративная разработка позволяет быстро реагировать на меняющиеся требования, обнаруживать и устранять риски на ранних стадиях проекта, а также эффективно контролировать качество создаваемого продукта.

Структуру жизненного цикла проекта, выполняемого по технологии RUP [5] удобно рассматривать на координатной плоскости (рис.1).

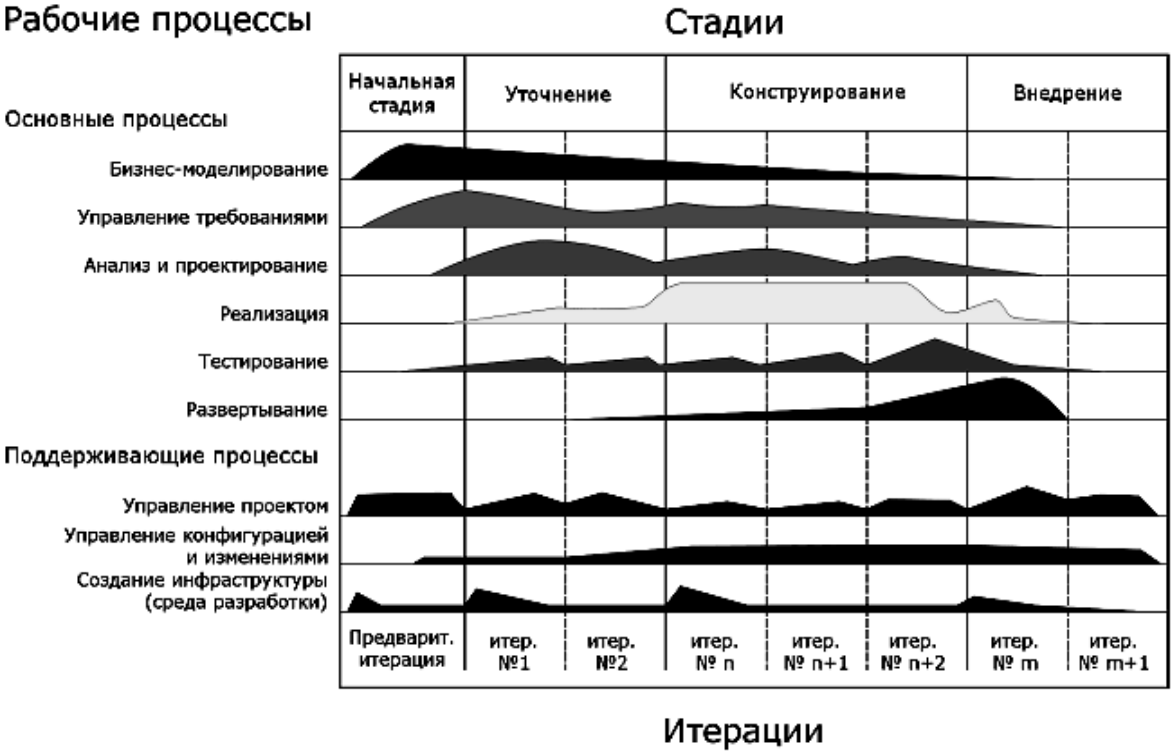

Рис. 1. Модель жизненного цикла RUP

Горизонтальное измерение представляет собой динамическую структуру или временное измерение процесса. Оно показывает, как процесс, выраженный в форме циклов, фаз, итераций и вех, развертывается в ходе жизненного цикла проекта.

Вертикальное измерение представляет собой статическую структуру процесса. Оно описывает, как элементы процесса – задачи, дисциплины, арте-

факты и роли – логически группируются в дисциплины или рабочие процессы (workflows).

Полный жизненный цикл разработки продукта состоит из четырех фаз, каждая из которых включает в себя одну или несколько итераций, причем переход с фазы на фазу возможен только после выполнения задач фазы и представляет собой контрольную точку (milestone) процесса:

1) начало (Inception) – основное внимание уделяется моделированию бизнес процессов и работе с требованиями. В этой фазе, как правило, осуществляется проектирование и реализация первого прототипа системы;

2) проектирование (Elaboration) – тщательно прорабатываются требования и выбираются основные проектные решения. В этой фазе концептуальный прототип превращается в реальную систему, позволяющую протестировать и оценить выбранные архитектурные решения;

3) построение (Construction) – быстрая и экономичная разработка кода системы. К концу фазы система должна быть готова к передаче заказчику для приемо-сдаточных испытаний.

4) внедрение (Transition) – посвящена подготовке разработанного продукта к передаче его заказчику, обучению пользователей, а также определению качества продукта.

RUP пригоден как для маленьких групп разработчиков, так и для больших организаций, занимающихся созданием ПО, т.к. в основе RUP лежит простая и понятная архитектура процесса, которая обеспечивает общность для целого семейства процессов.

Методология Microsoft Solutions Framework (MSF) – это методология ведения проектов и разработки решений, базирующаяся на принципах работы над продуктами самой фирмы Microsoft и предназначенная для использования в организациях, нуждающихся в концептуальной схеме для построения современных решений. MSF представляет собой согласованный набор концепций, моделей и правил.

MSF состоит из двух моделей:

−модель проектной группы;

−модель процессов,

и трех дисциплин:

−управление проектами;

−управление рисками;

−управление подготовкой.

В MSF нет роли "менеджер проекта" и иерархии руководства, управление разработкой распределено между руководителями отдельных проектных групп внутри коллектива, выполняющих следующие задачи (ролевые кластеры):

−управление программой (program manager) – разработка архитектуры решения, административные службы;

−разработку (developer) – разработка приложений и инфраструктуры, технологические консультации;

−тестирование (QAE) – планирование, разработка тестов и отчетность по тестам;

−управление выпуском (release manager) – инфраструктура, сопровождение, бизнес-процессы, выпуск готового продукта;

−удовлетворение заказчика (user experience) – обучение, эргономика, графический дизайн, техническая поддержка;

−управление продуктом (product manager) – бизнес-приоритеты, маркетинг, представительство интересов заказчика.

Жизненный цикл процессов в MSF сочетает водопадную и спиральную модели разработки: проект реализуется поэтапно, с наличием соответствующих контрольных точек, а сама последовательность этапов может повторяться по спирали.

При управлении проектом четко ставится цель, которую необходимо достичь в результате, и учитываются ограничения (ресурсов, времени и возможностей), накладываемые на проект. Эти три вида ограничений и приоритетность задач по их преодолению образуют треугольник приоритетов в MSF.

Методология eXtreme Programming (XP) – одна из гибких методологий разработки программного обеспечения, которая представляет собой небольшой набор конкретных правил, позволяющих максимально эффективно выполнять требования современной теории управления программными проектами.

XP ориентирована на:

−командную работу с тесными связями внутри команды и с заказчиком;

−разработку наиболее простых работающих решений;

−гибкое адаптивное планирование;

−оперативную обратную связь (путем модульного и функционального тестирования).

Основными принципами XP является разработка небольшими итерациями на основании порции требований заказчика (т.н. пользовательских историй), написание функциональных тестов до написания программного кода, постоянное общение и постоянный рефакторинг кода.

Обычно XP характеризуют набором из 12 правил (практик) [8], которые необходимо выполнять для достижения хорошего результата. Ни одна из практик не является принципиально новой, но в XP они собраны вместе и их можно объединить в 4 группы:

1) короткий цикл обратной связи (разработка через тестирование, игра в планирование, заказчик всегда рядом, парное программирование);

2) непрерывный, а не пакетный процесс (непрерывная интеграция, рефакторинг, частые небольшие релизы);

3) понимание, разделяемое всеми (простота, метафора системы, коллективное владение кодом/шаблонами проектирования, стандарт кодирования);

4) социальная защищенность программиста (40-часовая рабочая неделя).

Методология SCRUM **–** устанавливает правила управления процессом разработки и позволяет использовать уже существующие практики кодирования, корректируя требования или внося тактические изменения. Использование этой методологии дает возможность выявлять и устранять отклонения от желаемого результата на более ранних этапах разработки программного продукта.

Основа Scrum – итеративная разработка. Весь проект делится на итерации (спринты) продолжительностью 30 дней каждый. Методология, предназначенная для небольших команд (до 10 человек). Выбирается список функций системы, которые планируется реализовать в течение следующего спринта. Самые важные условия – неизменность выбранных функций во время выполнения одной итерации и строгое соблюдение сроков выпуска очередного релиза, даже если к его выпуску не удастся реализовать весь запланированный функционал. Руководитель разработки проводит ежедневные 20 минутные совещания, которые так и называют – scrum, результатом которых является определение какие функции системы были реализованы за предыдущий день, с какими сложностями столкнулись и что планируется сделать за следующий день.

Методология позволяет команде выбрать задачи, которые должны быть выполнены, учитывая бизнес-приоритеты и технические возможности, а также решить, как их эффективно реализовать. Это позволяет создать условия, при которых команда работает с удовольствием и максимально продуктивно. Основной упор методология Scrum делает на управление проектами и не задает никаких технических практик.

Методология Crystal Clear **–** методология, позволяющая менять степень формализации процесса разработки в зависимости от критичности задач и количества участников разработки.

Основные особенности: итеративная инкрементная разработка, автоматическое регрессионное тестирование, пользователи привлекаются к активному участию в проекте, состав документации определяется участниками проекта, как правило, используются средства контроля версий кода. Ориентирована на поддержание естественных привычек разработчиков.

Помимо Crystal Clear в семейство Crystal входит еще несколько методологий, предназначенных для выполнения более крупных или более критических проектов. Они отличаются несколько более жесткими требованиями к объему документации и вспомогательным процедурам, таким как управление изменениями и версиями.

Методология разработки динамических систем (DSDM) **–** позволяет разрабатывать проект в короткие сроки с использованием ограниченного количества ресурсов, предусмотренного бюджетом.

DSDM стремиться сократить связующие звенья между заказчиком и разработчиком, аналитиком и дизайнером, между членами команды и между разными уровнями управления, чтобы обеспечить более эффективные процесс разработки программного обеспечения. Фиксируя время разработки и устанавливая ограничения на используемые ресурсы, намного проще создать процесс разработки, отвечающий требованиям заказчиков.

Основные принципы DSDM:

– Активное вмешательство пользователя (в идеале, пользователи и разработчики разделяют рабочее пространство, поэтому решения могут приниматься на месте).

– Команда должна иметь возможность принимать решения без ожидания подтверждения вышестоящими уровнями (команда состоит как из разработчиков, так и из заказчиков).

– Требования заказчика - прежде всего.

– Все изменения обратимы (возвращение - основная черта DSDM).

Тестирование происходит на протяжении всего жизненного цикла разработки, а не перед самым выпуском, когда что либо изменить без больших затрат уже невозможно (тестирование проводится как разработчиками, так и пользователями). Сроки должны быть сжатыми и определяющими скорый выпуск продукта.

Оценка программы должна быть основана на требуемой функциональности, а не на строчках кода.

Оценка риска должна фокусироваться на функциях, которые должны быть получены, а не на процессе их построения.

Методология Feature Driven Development (FDD) **–** функционально-ориентированная разработка, оперирует понятием функции или свойства (feature). Разработка состоит из 5 основных этапов:

1) разработка общей модели,

2) составление списка необходимых функций системы,

3) планирование работы над каждой функцией,
4) проектирование функций,
5) конструирование функций.

Разработчики в FDD делятся на «хозяев классов» и «главных программистов».

Главные программисты привлекают хозяев задействованных классов к работе над очередным свойством.

Работа над проектом происходит итеративно, частый выпуск версий, каждая из которых реализует определенный набор функций.

Методология адаптивной разработки (ASD) – появилась как альтернатива традиционным ориентированным на процесс, методам управления разработкой ПО. Главную роль играют человеческий фактор, результаты работы и минимизация самого процесса при максимально увеличении взаимодействия между людьми.

ASD базируется на принципе непрерывной адаптации, благодаря которой возникает другой жизненный цикл проекта. Постоянные изменения в проекте становятся нормой.

В ASD обычный статический жизненный цикл "Планирование – Проектирование – Конструирование" заменен на динамический "Обдумывание – Взаимодействие – Обучение". Этот цикл ставит своей целью непрерывное обучение. Он связан с постоянными изменениями, повторными оценками попытками предугадать неизвестное на текущий момент будущее проекта и требует тесного взаимодействий между разработчиками, тестировщиками и заказчиками.

Методология ASD построена на концептуальной базе теории сложных адаптивных систем. Она рассчитана на использование в экстремальных проектах, в которых превалируют быстрый темп разработок, непредсказуемость и частые изменения. Есть проекты, которые не могут считаться экстремальными, однако для всех остальных ASD подходит гораздо лучше, чем любой традиционный подход к разработке ПО.

Опираясь на вышеприведенное описание методологий, составим таблицу их сравнительных характеристик (табл.1).

Под уровнем *формализма* будем понимать наличие стандартов и правил оформления, детализации и движения сопутствующей документации в процессе разработки ПО.

Таблица 1

Сравнительные характеристики различных методологий разработки ПО

	RUP	**«Гибкие» (XP, SCRUM, Crystal, DSDM, FDD, ASD)**	**MSF**
Основные принципы	Ориентация на процесс, детальное планирование	Ориентация на человека, переоценка и постоянное планирование	Совмещает RUP и «Гибкие»
Модель ЖЦ	Итеративная	Итеративная	Каскадная + спиральная
Уровень формализма	высокий	низкий	высокий, средний
Критерии успеха	Соответствие плану	Отклик на изменения, работающий код	Совмещает RUP и «Гибкие»
Культура управления	Командно-административная	Лидерство и сотрудничество	Лидерство и сотрудничество

3. Классификация типов программных проектов

В настоящее время большое развитие получило создание т.н. *прикладных программ,* непосредственно обеспечивающих выполнение необходимых пользователям работ. Характерными чертами данных программных проектов (ПП) являются их способность к адаптации, технологическая инновационность, креативность и открытость, а также способность решать широкий круг задач в интересах конечных пользователей.

По ступеням надёжности различают следующие типы программных проектов:

– разработанные впервые;

– существующие (разработанное ранее или покупное коммерческое);

– конфигурируемые из типовых модулей.

По влиянию на безопасность различают:

– программные проекты, которые влияют на безопасность (критическое ПО);

–программные проекты, которые не влияют на безопасность.

По масштабу (количеству участников) программных проектов:

−малые;

−средние;

−мега-проекты.

По принадлежности, данные ПП можно подразделить на следующие типы:

−офисные приложения (текстовый редактор или процессор и пр.);

−корпоративные информационные системы (аудиторская или бухгалтерская программа, система ERP и пр.);

−системы проектирования и производства (САПР, CAD и т.п.);

−научное ПО (система компьютерного моделирования и пр.);

−информационные системы;

−клиент для доступа к интернет -сервисам (электронная почта, банк клиент и пр.);

−мультимедиа (компьютерная игра, медиапроигрыватель и т.п.).

4. Основные этапы подхода к созданию предметноориентированной методологии разработки ПО

Все доступные модели и методологии хороши только для определённых задач и/или проектов, в которых [эти методологии] учитывают различные технические, организационные, проектные и командные особенности и поэтому выбор методологии – один из важнейших факторов, который существенно влияет на успех проекта.

Главная трудность состоит в том, что практически невозможно точно определить объемы задачи в самом начале проекта, а следовательно, и минимальное число людей, которые могут эту задачу решить. А если размеры проекта возрастают, может оказаться, что оптимальным решением будет применить другую методологию. Основными факторами, влияющими на выбор методологии управления проектом можно считать следующие:

− критичность проекта, при которой скрытые ошибки в программном проекте могут повлечь за собой значительный ущерб;

− цели проекта, приоритеты;

− масштаб проекта (необходимый размер команды, сложность задач);

− используемые средства разработки и внедряемые системы;

− компетентность участников команды разработки и внедрения;

− география разработки и внедрения;

− культура и традиции компании заказчика и компании-интегратора, ценности команды разработчиков.

Существующие методологии можно подразделить на два типа:

−«легкие» методологии – присуща низкая степень формализма;

− «тяжелые» методологии – классические методологии с высокой степенью формализма.

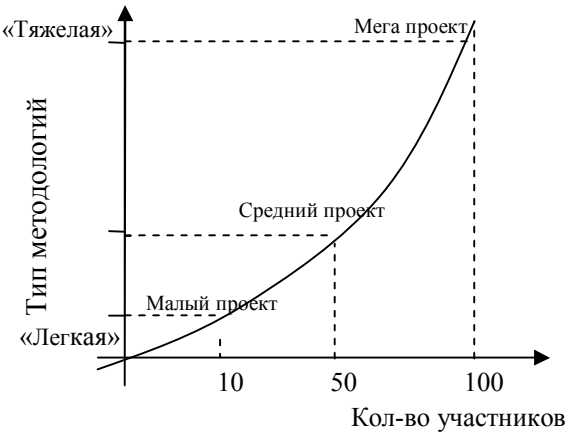

Рис. 2. Зависимость стратегии применения методологий от масштабов проекта

Другими словами, чем «легче» методология, тем продуктивнее работает команда, а беря во внимание приведенную классификацию типов ПП, можно утверждать что размеры проекта и методологии связаны между собой положительной обратной связью (рис. 2).

Следуя данному утверждению можно предположить, что один из принципов подхода к формированию стратегии разработки программного обеспечения, заключается в следующем: чем меньше масштаб проекта, тем более простой («легкий») тип методологи применяется.

Выводы

Предложенный подход к формированию стратегии применения методологий разработки программного обеспечения, основанный на композиции этапов различных методологий, который в отличии от существующих позволяет установить однозначное соответствие между конкретными типами программных проектов и рассматриваемыми методологиями, снизив тем самым уровень неопределенности при разработке стратегии создания программного обеспечения.

Литература

1. Ройс У. Управление проектами по созданию программного обеспечения / У. Ройс. – М.: Лори, 1998. – 424 с.

2. Демарко Т. Человеческий фактор: успешные проекты и команды / Т. Демарко, Т. Листер. – М.: Символ-Плюс, 2005. – 256 с.

3. Анализ требований и создание архитектуры решений на основе Microsoft .NET: учеб. курс MCSD: пер. с англ. – М.: ИД "Русская Редакция", 2004. – 416 с.

4. Вигерс К. Разработка требований к программному обеспечению: пер. с англ. / К. Вигерс. – М.: ИД "Русская Редакция", 2004. – 576с.

5. Крачтен Ф. Введение в Rational Unified Process: пер. с англ. / Ф. Кратчен. – М.: Вильямс, 2002. – 240 с.

6. Ньютон Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Бизнес Букс, 2007. – 192 с.

7. James P.L. The Project manager's pocket survival / P.L. James. – New York: John Wiley & Sons, Inc 2002. – 246 p.

8. Бек К. Экстремальное программирование: Планирование // К. Бек, М. Фаулер; Библиотека программиста. – СПб.: Питер, 2003. – 144 с.

9. Брукс Ф.П. Мифический человеко-месяц, или Как создаются программные системы / Ф.П. Брукс. – СПб.: Символ-Плюс, 1999. – 265 с.

10. Гагарина Л.Г. Технология разработки программного обеспечения / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул. – М.: ИД «ФОРУМ», 2008. – 400 с.

11. Кармайкл Э. Быстрая и качественная разработка программного обеспечения / Э. Кармайкл, Д. Хэйвуд . – М.: «Вильямс», 2003. – 400 с.

12. Амблер С. Agile Modelin/ С. Амблер. – СПб.: Изд-во "Питер", 2003. – 256 с.

Поступила в редакцию 1.06.2010

Рецензент: д-р техн. наук, проф., проф. кафедры ПО ЭВМ С.Ю. Шабанов-Кушнаренко, Харьковский национальный университет радиоэлектроники, Харьков.

ПІДХІД ДО ФОРМУВАННЯ СТРАТЕГІЇ ВЖИВАННЯ МЕТОДОЛОГІЇ РОЗРОБКИ З УРАХУВАННЯМ ОСОБЛИВОСТЕЙ ПРОГРАМНИХ ПРОЕКТІВ

І.В. Шостак, М.О. Данова

Розглянуто існуючі методології розробки програмних проектів, а також наведені порівняльні характеристики найбільш ефективних і часто вживаних методологій, таких як RUP, MSF, XP, Crystal та інших. Наведено класифікацію типів програмних проектів за масштабом, надійністю, впливом на безпеку та багатьох інших. Запропоновано підхід до формування стратегії вживання методологій розробки програмного забезпечення для різних типів програмних проектів, який дозволяє встановити однозначну відповідність між конкретними типами програмних проектів і розгляданими методологіями.

Ключові слова: методологія розробки ПП, прикладні програми, програмний проект, класифікація ПЗ, стратегія, масштаб проекту.

THE APPROACH TO FORMATION OF USAGE STRATEGY OF METODOLOGIES IN SOFTWARE DEVELOPMENT FOR VARIOUS TYPES OF SOFTWARE PRODUCTS

I.V. Shostak, M.A. Danova

Existing methodologies of software products are considered and comparative characteristics of the most effective and frequently used methodologies, such as RUP, MSF, XP, Crystal and others are resulted. Classification of types of software products on scale, reliability, influence on a security and by many other things is resulted. The approach to formation of usage strategy of methodologies in software development for various types of software products which allows to establish unequivocal conformity between specific types of program projects and considered methodologies is represent.

Key words: methodologies in software development, application software, program project, classification of software, strategy, scale of the project.

Шостак Игорь Владимирович – д-р техн. наук, доцент, проф. каф. инженерии ПО Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

Данова Мария Александровна – соискатель по каф. инженерия ПО Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: danovamariya@gmail.com.