УДК 004.9

Е.Г. Кириленко, Г.А. Фролова

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

МОДЕЛИ ОРГАНИЗАЦИИ КОЛЛЕКТИВОВ РАЗРАБОТЧИКОВ В ОБЛАСТИ ПРОГРАММНОЙ ИНЖЕНЕРИИ

В статье исследованы особенности организации коллективов разработчиков программного обеспечения (ПО). Выявлено, что важной задачей в проекте является распределение ролей и функций в группе, а также организация взаимодействия как внутри группы, так и между группами. Приведены характеристики традиционных и гибких моделей организации групп разработчиков ПО. Определено, что структура группы и взаимодействие в ней определяются, в основном, технологией процесса разработки ПО. Выявленные особенности организации коллективной разработки ПО могут стать основой для разработки содержания соответствующих модулей учебной дисциплины «Групповая динамика и коммуникация».

Ключевые слова: *программная инженерия, групповая динамика и коммуникация, модели организации коллективов разработчиков программного обеспечения, технологии разработки процессов программного обеспечения.*

Введение

Программная инженерия – область компьютерной науки и технологии, которая занимается построением программных систем, настолько больших и сложных, что для этого требуется участие слаженных команд разработчиков различных специальностей и квалификаций.

В ответ на требования времени к уровню подготовки специалистов в учебные программы в области программной инженерии по специальности 5.0501201 «Разработка программного обеспечения» и специальности 6.050103 «Программная инженерия» включена учебная дисциплина «Групповая динамика и коммуникация», которая относится к нетехническим обязательным дисциплинам и нацелена на развитие у будущих инженеров дополнительных профессиональных качеств гуманитарно-социального характера.

Целью изучения дисциплины является формирование у будущих специалистов в области программной инженерии системного представления о коллективной разработке программных продуктов, включая гуманитарные и инженерные аспекты.

Задачами дисциплины являются: изучение основополагающих теоретических положений о совокупности возникающих в группах разработчиков программного обеспечения процессов и явлений, затрагивающих групповую динамику; формирование навыков эффективной устной и письменной коммуникации, как в межличностном, так и в межгрупповом общении в процессе разработки программного обеспечения.

Основной проблемой разработки учебной дисциплины является ее междисциплинарный характер содержания, требующий интеграции профессиональных, гуманитарных, социальных знаний.

1. Анализ исследований и публикаций

Замечательные люди создают замечательные программы. Они формулируют требования, отлаживают технологию и придерживаются графиков. Они тестируют, документируют и сопровождают продукт. Их идеи, профессионализм и энтузиазм определяют успех или провал разработки [7].

В процессе разработки программного обеспечения (ПО) специалисты выполняют различные виды деятельности, связанные с анализом, проектированием, конструированием, реализацией, тестированием и сопровождением программного продукта. В группе они могут исполнять некоторые роли по отношению друг к другу.

Роль в этом случае рассматривается как временное назначение сотруднику набора функций в рамках конкретного проекта. Получение роли означает делегирование полномочий для выполнения определенных функций и принятие ответственности за результаты выполнения этих функций. Роль может требовать выполнения разных функций; некоторые функции могут быть присущи нескольким ролям.

Поскольку на судьбу проекта больше всего влияет «человеческий фактор», то очень важной задачей является распределение ролей и функций внутри группы, а также организация взаимодействия как внутри, так и между группами. Решение этой задачи зависит от размера, структуры, состава группы, рабочего окружения [5].

Многие технологии разработки ПО определяют роли, которые специалисты могут выполнить в проекте и типы задач, соответствующие той или иной роли, что показано в работах А. Якобсона, Г. Буч, Дж. Рамбо – RUP [10]; К. Бек, экстремальное программирование [2]; Стэплтон – DSDM [14]; К.

Швабер, Д. Сазерленд – SCRUM[13] и др. Использование той или иной технологии разработки ПО позволяет организовать эффективную коммуникацию между группами и внутри групп и добиться успеха в коллективной разработке ПО.

2. Постановка задачи

Для эффективной работы специалисту в области программной инженерии необходимо *знать* дифференциацию групп, структуру группы, дифференциацию ролей в группе, а также понимать, как осуществляется взаимодействие внутри групп и между группами в процессе разработки ПО.

Таким образом, актуальной является задача анализа особенностей моделей организации коллективов разработчиков ПО.

3. Традиционные модели организации коллективов разработчиков ПО

3.1 Модель бригада

В конце 60-хх начале 70-хх годов в связи с ростом сложности программных систем появилась необходимость организации отдельных программистов в бригады или в группы бригад. Оптимальная структура бригады определялась числом людей, их подготовкой, индивидуальными особенностями, самим проектом и организационными условиями [12].

К первой разновидности этой модели можно отнести модель «***Обычной бригады***», в которой старший программист руководит работой нескольких младших программистов. По мере накопления опыта старший программист может стать руководителем проекта, организующим работу нескольких бригад, а младшие программисты могут стать старшими.

Преимуществом этой модели является то, что компетентность вознаграждается продвижением по службе, явно выражены связи между ответственностью и полномочиями. Задания распределяются вышестоящими работниками, работа выполняется индивидуально, а результаты оцениваются руководителем. Конкуренция между членами бригады рассматривается как здоровый стимул к повышению уровня производительности труда.

К *недостаткам* данной модели относят: низкую компетентность управления проектами ведущими программистами, с одной стороны, и отсутствие возможности использовать их способности в программировании в проекте, с другой стороны, а так же большое влияние ошибок отдельного программиста на работу других членов в бригаде [9]

Следующей разновидностью модели стала «***Бригада без персонализации***» [9]. Программирование без персонализации – определенное состояние ума, при котором программисты отделяют себя от своей продукции. Такое разделение позволяет программистам спрашивать совета, не опасаясь показаться некомпетентным, без опаски воспринимать замечания относительно собственных программ и смеяться над ошибками в своих программах.

В «Бригадах без персонализации» поощряется скорее кооперация, а не конкуренция и успех и неудача отдельного человека рассматриваются как следствие работы всего коллектива. Уяснение целей проекта, распределение работ осуществляется всей бригадой. Если кто-то из членов бригады испытывает трудности при выполнении своей работы, то ему помогают в их преодолении.

По мнению Б. Шнейдермана большой успех имеют, вероятно, «Бригады без персонализации», состоящие из программистов равной квалификации и имеющих опыт совместной работы. В тоже время, при отсутствии авторитетной личности, которая поощряет и наказывает, без ясных перспектив продвижения в «Бригадах без персонализации» возможны усложнения при установлении высокого уровня кооперации, необходимого для стабильной работы [9].

Потеря квалификации программиста при выдвижении на руководящие посты и все возрастающая изолированность программистов привели к идее создания модели ***«Бригады главного программиста»*** [3].

В этой модели каждый выполняет определенную роль, а не получает определенную часть программы. Таким образом, можно повысить производительность, уменьшить число ошибок, успешнее справляться с планами и получать большее удовлетворение от работы.

В «Бригаде главного программиста» выделяют ниже перечисленные роли.

Главный хирург является главным конструктором и пишет наиболее важные куски программы. Он должен быть компетентным в данной области, иметь стаж работы свыше 10 лет, обладать существенными знаниями в системных и прикладных областях.

Второй пилот выполняет особо важные действия и может при необходимости заменить хирурга, знает хорошо код программы, исследует возможности альтернативных стратегий проектирования, может заниматься написанием кода, но не несет ответственность за него. Хирург испытывает на нем свои идеи, но не связан его предложениями.

Администратор исполняет роль хирурга – начальника, и ему принадлежит последнее слово в отношении персонала, прибавок к жалованью, обеспечения жизнедеятельности бригады и др. Один администратор может обслуживать две команды.

Редактор критически перерабатывает документацию, созданную хирургом, снабжает ее ссылками, библиографией, обеспечивает несколько версий и публикацию.

Два секретаря. Администратору и редактору нужны секретари. Секретарь администратора обрабатывает переписку, связанную с проектом, а также документы, не относящиеся к проекту.

Делопроизводитель отвечает за регистрацию всех технических данных бригады в библиотеке программного продукта. Он имеет секретарскую подготовку и несет ответственность за все файлы, предназначенные как для машины, так и для чтения.Когда проектные спецификации готовы к реализации, к работе подключаются другие программисты.

Инструментальщик обеспечивает необходимым или желательным инструментом своего хирурга, а не другие бригады. Он обычно пишет специализированные утилиты, каталогизированные процедуры, макробиблиотеки.

Отладчик, с одной стороны, планирует последовательность тестирования, создание среды для тестирования компонентов, разрабатывает примеры для системного тестирования, исходя из функциональных спецификаций, а с другой стороны, готовит данные для ежедневной отладки.

Языковед – член бригады, владеющий тонкостями языка программирования, находящий эффективные способы использования языка для решения сложных, неясных или хитроумных задач. Один языковед может работать с двумя или тремя хирургами.

По мнению автора модели ***«Бригада главного хирурга»*** позволяет превратить программирование из личного искусства в общественную деятельность.

Модель ***«Бригады главного программиста»*** имеет ряд модификаций, связанных с удалением или объединением нескольких ролей и имеет следующие достоинства:

– предсказуемость, в случае, если главный программист некомпетентен, то это выявляется на ранних стадиях проекта и проект может быть прекращен или реорганизован практически без убытков. Если главный программист компетентен, то вероятность успешного завершения проекта высока даже при наличии серьезных внешних факторов риска.

– автономность, которая позволяет успешно функционировать даже в изменяющейся и неблагоприятной внешней среде.

– гибкость, которая позволяет ориентировать бригаду на различные типы программных проектов.

– единоначалие, которое обеспечивает высокую степень надежности, устойчивости и управляемости группой.

К *недостаткам* ***«Бригады главного программиста»*** можно отнести: зависимость успеха проекта от главного программиста; неравномерная нагрузка членов бригады; отсутствие удовлетворения от работы у программиста-заместителя и программистов помощников, работа которых оценивается ниже, несмотря на то, что проект в целом является их собственным.

3.2 Иерархическая модель группы

Иерархическая модель, которая показана на рис. 1, является самой простой и распространенной и проверенной временем. В зависимости от масштаба проекта иерархическая модель может иметь разнообразное количество слоев в иерархии, а каждый горизонтальный слой может содержать разное количество элементов в слое.

В ***иерархической модели*** каждый член группы исполняет отведенную ему роль в иерархии. Для координации работы в такой структуре руководитель наделен полномочиями. Он управляет и контролирует работу сотрудников. Эта структура может быть расширена с помощью рекурсии. В результате появится иерархия руководителей, где высшие чины руководят низшими.

В проектах по разработке ПО, в которых используется эта модель, в иерархии может и не быть много уровней, но это все равно иерархия. У каждого есть своя работа, своя роль, свои обязанности и свое место в иерархии. Корпоративные интересы, или интересы отдела, или интересы проекта стоят превыше всего, а работники получают вознаграждение за добросовестное исполнение своей части в своей работе [5].

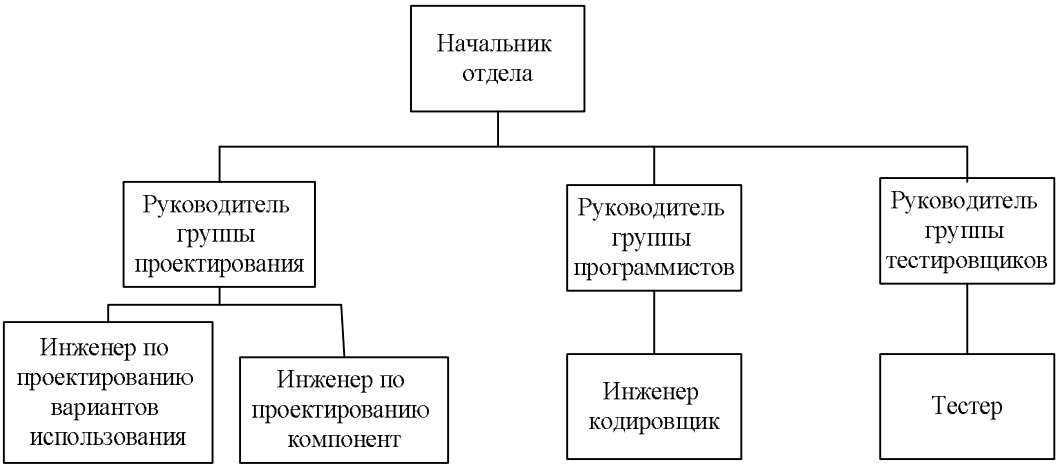

Рис.1. Модель иерархической группы

К основным *достоинствам* ***иерархической модели*** организации группы можно отнести:

– единоначалие, которое обеспечивает высокую степень надежности, устойчивости и управляемости группой;

– отсутствие тренингов по внедрению и обучению персонала, т.к. иерархическая модель привычна и проверена временем;

– предсказуемое и надежное исполнение достигается за счет стандартов, процедур и правил работы;

– высокая масштабируемость позволяет применять модель в масштабах от десятков до сотен тысяч исполнителей.

К *недостаткам* ***иерархической модели*** можно отнести:

– наращивание функциональности обеспечивается увеличением количества сотрудников;

– ориентация на выполнение строго определенных функций. Изменение функций требует изменения всей структуры иерархии;

– жесткое закрепление за каждым исполнителем его ролевой функции;

– плохая адаптация к быстрой смене технологий и парадигм;

– влияние отрицательных качеств руководителя на эффективность работы группы.

4. Гибкие модели организации коллективов разработчиков ПО

В начале 90-хх годов для организации производства в небольших и средних по размеру группах, занимающихся разработкой программного продукта в условиях неясных или быстро меняющихся требований стали применяться гибкие технологии разработки ПО.

В гибких технологиях основным приоритетом является итерационность процесса разработки ПО, в котором после каждой итерации появляется результат, доступный для оценки и просмотра, а основной акцент делается на организацию группы и эффективное взаимодействие в ней, ее циклов, требующих от членов группы планирования и выполнения задач проекта.

Основными принципами гибких технологий являются: работающая программа ценнее документа, перечисляющего ее функции; регулярное сотрудничество с клиентами ценнее подробного контракта, описывающего предполагаемое использование продукта; специалисты ценнее процессов и инструментария; прежде всего должна цениться восприимчивость к изменениям в установленном плане [15].

С. Амблер отмечает, что для создания эффективной команды, использующей гибкие технологии разработки программных продуктов необходимо: нанять несколько хороших разработчиков; усвоить, что в гибкой разработке нет «Я»; потребовать от всех активного участия; моделировать в группе [1].

4.1 Модель группы XP

Автор подхода экстремальное программирование (eXtreme Programming) К. Бек позиционирует его как упрощенный, эффективный, гибкий, предсказуемый, научно обоснованный и весьма приятный способ разработки программного обеспечения, предусматривающий низкий уровень риска.

К его особенностям К. Бек относит то, что в экстремальном программировании объединены все давно известные приемы, собранные под одной крышей; интенсивность, с которой эти приемы внедряются в повседневную работу, доведена до крайности; используемые методики поддерживают одна другую в наибольшей возможной степени [2].

Четыре ценности лежат в основе XP: коммуникация (communication); простота (simplicity); обратная связь (feedback); храбрость (courage); уважение (esteem). Непротиворечиво следовать этим ценностям позволяют методики: игра в планирование (planning game); непрерывные выпуски версий малыми порциями (small releases); простой дизайн (simple design); разработка через тестирование (testing); переработка кода (refactoring); программирование парами (pair programming); коллективное владение кодом (collective ownership); непрерывная интеграция системы (continuous integration); 40-часовая неделя (40-hour week); заказчик на месте разработки (on-site customer).

Технология разработки XP предназначена для работы над проектами, в которых работает от двух до десяти программистов, не зажатых в жесткие рамки существующего компьютерного окружения и в которых вся необходимая работа, связанная с тестированием, может быть выполнена в течение одного дня.

В технологии XP выделены две обобщенных роли: менеджер и разработчик. Одним из основополагающих правил стратегии XP является то, что технари концентрируются на решении технических проблем, а бизнесмены – на решении бизнес-проблем. В XP менеджмент исполняет две роли: *Инструктор* (coach) и *Ревизор* (tracker). Обе эти роли могут исполняться одним и тем же членом команды, однако, это могут быть также разные люди.

Инструктор отвечает за весь процесс разработки и концентрирует свое внимание на общей производительности команды. К его основным обязанностям относятся: быть доступным в качестве партнера по разработке; видеть долгосрочные цели переработки кода и стимулировать мелкомасштабную переработку; помогать программистам индивидуальными техническими навыками; предоставлять информацию о ходе процесса разработки менеджерам высшего звена.

Ревизор отслеживает (tracking) ход процесса и обеспечивает обратную связь между заказчиком и разработчиками. В его обязанности входит сбор и документирование сведений о состоянии метрик, которые отслеживаются в данный момент и напоминание об изменениях, вносимых в план. К задаче менеджмента также относится контроль и принятие даже самых непопулярных решений в экстремальных случаях.

Консультант. В процессе экстремального программирования время от времени члены группы нуждаются в глубоких технических знаниях. Задача консультанта обеспечить членов группы необходимыми знаниями для того, чтобы они самостоятельно смогли решить проблему.

В XP метафора программирования используется для всех видов деятельности (проектирование, кодирование, тестирование) благодаря чему все, чем занимаются разработчики, выглядит как программирование.

Особенностью *роли программиста* в XP является то, что он должен иметь сформированные навыки общения и координации своей деятельности с деятельностью других членов группы; иметь привычку к поиску простого решения; уметь хорошо программировать; иметь сформированные навыки коллективного владения кодом; не бояться принимать новые технические решения.

Заказчик в группе XP является полноправным ее членом, к основным обязанностям которого входят: принятие решения о готовности продукта; документирование историй; написание функциональных тестов; участие в принятии некоторых важных решений вместе с программистами;.

Большая часть обязанностей, связанных с тестированием, лежит на плечах программистов, поэтому роль человека, выполняющего тестирование, в команде XP фокусируется на заказчике.

Тестер помогает заказчику в подборе и написании функциональных тестов. Если функциональные тесты не являются частью интеграционного пакета, тестер отвечает за регулярный запуск функциональных тестов и публикацию результатов в обозримом для всех месте.

Архитектор. Особенностью работы архитектора в группе XP является то, что вместо того, чтобы разрабатывать идеальную архитектуру системы и пытаться ей следовать, архитекторы создают и перерабатывают архитектуру по необходимости.

По мнению К. Бека и других экспертов в области программной инженерии точные пределы использования XP еще не до конца исследованы. Основными факторами, которые ограничивают применение XP являются: большое количество участников в группе разработчиков, большое количество групп, обязательное наличие заказчика в группе, одинаковый профессиональный уровень и высокие коммуникативные навыки всех разработчиков.

4.2 Модель группы Scrum

В 1968 г. Х. Такеучи и И. Нонака описали модель производства, которая применялась в области машиностроения и электроники. В соответствие с этой моделью успешность и гибкость работы проектов обеспечивали небольшие сплоченные команды без жесткой специализации (пример – табл.. 1).

Таблица 1

Распределение ролей в группе *Scrum*

Роль	Ответственность
Владелец продукта (Product Owner) отвечает за формирование списка требований к функциональности продукта. Обычно владелец продукта является представителем или доверенным лицом заказчика.	Формирует журнал продукта; определяет приоритеты функциональности в соответствии с их коммерческой ценностью; определяет дату и содержание выпусков продукта; отвечает за коммерческую успешность продукта; уточняет функциональность и приоритеты после каждого спринта; принимает (или отклоняет) результаты проекта.
Команда (Scrum Team) – группа, состоящая из пяти–девяти (7±2 человека) самостоятельных, инициативных разработчиков, включая архитекторов, программистов, тестеров и др. специалистов, непосредственно работающих над реализацией проекта.	Активно участвует в выборе цели следующей итерации; имеет право предпринимать все возможные (в рамках проекта) действия ради достижения цели спринта; обеспечивает принятый уровень качества продукта; самостоятельно организует себя и свою работу; демонстрирует результаты работы владельу проекта.
Скрам-мастер (ScrumMaster) – посредник между всеми участниками проекта, основной обязанностью которого являются организация процесса и соблюдение технологии Scrum.	Гарантирует высокую производительность работы группы; отвечает за обеспечение продуктивного взаимодействия между всеми участниками проекта; защищает команду от всех нежелательных внешних воздействий; отвечает за соблюдение в процессе всех принципов технологии Scrum.

Термин Scrum был ими заимствован из игры регби, где scrum – это команда из восьми человек, каждый член которой в каждый момент играет конкретную роль и взаимодействует с другими членами команды для достижения общей цели. Джеф Сазерленд и Кен Швабер использовали этот подход в индустрии разработки программных продуктов [13]. Основным принципом ***Scrum*** группы является то, что

вся команда должна принимать участие в итеративном процессе разработки ПО. Последовательность итераций в процессе называются спринтами (Sprint).

Четкое распределение ролей и обязанностей в группе ***Scrum***, представленное в табл. 1, позволяет эффективно принимать решения по отношению к достижению целей проекта. Все участвующие в проекте специалисты разделяются на две большие символические группы: «поросята» (pigs) и «цыплята» (chickens). Такое разделение прав и обязанностей между разработчиками («поросятами») и менеджментом («цыплятами») позволяет, с одной стороны, группе разработчиков (Scrum Team) получать возможность самостоятельно принимать часть решений, а с другой стороны, группе менеджмента (Product Owner, ScrumMaster, представители менеджмента организации) контролировать ход проекта.

Проект в Scrum имеет три основных фазы: подготовка (pregame); разработка (game) и завершение (postgame). В процессе выполнения проекта технология Scrum предполагает регулярное проведение формальных сессий или собраний, таких как сессия планирования спринта (Sprint Planning Meeting), ежедневная скрам-сессия (Daily Scrum) и демонстрационная сессия (Sprint Review Meeting). Все сессии имеют формальное ограничение по времени и четкое разграничение прав и обязанностей всех участников проекта, что предотвращает возникновение серий затяжных совещаний, которые могут снизить эффективность работы группы, а также внутригрупповые и межгрупповые конфликты.

Реализация проекта в технологии Scrum управляется тремя документами: журнал продукта (Product Backlog), журнал спринта (Sprint Backlog) и график спринта (Burndown Chart) [16].

4.3 Модель группы DSDM

Технология разработки ПО Dynamic Systems Development Method (**DSDM**) сфокусирована на организации процесса производства программного обеспечения. Технология **DSDM** определяет набор стандартных ролей и ответственностей, которые перечислены ниже. На рис. 2 приведена типичная структура команды разработчиков (участники проекта, имеющие полную занятость, изображены в закрашенных овалах)

Менеджер проекта (Project Manager) обеспечивает общее руководство проектом.

Провидец (Visionary) является движущей силой проекта. Он следит за соответствием проекта коммерческим целям и задачам. Провидцем часто является топ-менеджер, который инициировал проект. Данная роль предполагает возможность частичной занятости.

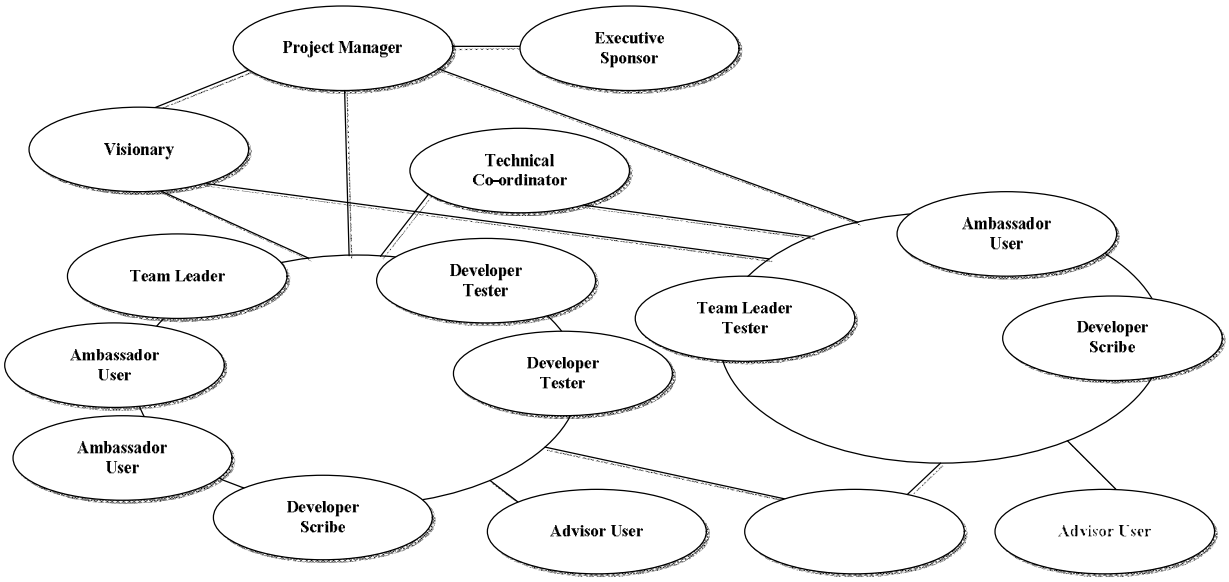

Рис. 2. Модель группы DSDM

Чемпион проекта (Project Champion или Executive Sponsor) обладает возможностями и обязанностями по распоряжению ресурсами и фондами, которые необходимы данному проекту. Он несет ответственность за принятие любых решений, связанных с проектом. Данная роль предполагает возможность частичной занятости.

Лидер команды (Team Leader) руководит командой разработчиков и обеспечивает эффективность ее работы.

Технический координатор (Technical Co-ordinator) отвечает за разработку архитектуры продукта. Технический координатор также отвечает за общее техническое состояние проекта.

Разработчик (Developer) участвует в анализе требований, моделировании, проектировании, основной обязанностью разработчика является программирование.

Тестировщик (Tester) отвечает за техническое тестирование продукта.

Представительный пользователь (Ambassador User) представляет пользователей продукта. Представительный пользователь отвечает за то, чтобы разработчики вовремя получали обратную связь со стороны пользователей.

Пользователь-консультант (Advisor User) может быть любой пользователь, представляющий значительную точку зрения на продукт. Пользователь-консультант привносит в проект знание по некоторому аспекту использования разрабатываемого продукта. Данная роль предполагает возможность частичной занятости.

Секретарь (Scribe) отвечает за протоколирование всех соглашений и решений, принятых во время семинаров.

Посредник (Facilitator) отвечает за проведение семинаров. Посредник также отвечает за эффективность коммуникации между всеми членами команды.

Технология разработки ПО DSDM рекомендует создавать команды небольшого размера, до шести человек (без учета руководящих персон вроде провидца и чемпиона проекта). Над одним проектом одновременно может работать несколько команд. Известны примеры проектов, выполненных по технологии DSDM, в которых участвовало до 150 человек. Команды могут отвечать как за реализацию некоторого набора функциональности (например, отдельная команда может отвечать за реализацию системы базы данных), так и за некоторые виды деятельности (например, в проекте могут быть две команды, отвечающие за разработку, и одна команда, отвечающая за тестирование).

Для установления взаимодействия между всеми участниками проекта предусмотрено проведение семинаров, на которые приглашаются только действительно заинтересованные лица или их представители, что позволяет эффективно проводить семинары. Также предусмотрен вариант привлечения незаинтересованных арбитров в случае обсуждения острых тем [11].

4.4 Модель проектной группы MSF

Microsoft Solutions Framework (MSF) Team Model описывает подход Microsoft к организации работающего над проектом персонала и его деятельности в целях максимизации успешности проекта. Данная модель определяет ролевые кластеры, их области компетенции и зоны ответственности, а также рекомендации членам проектной группы, позволяющие им успешно осуществлять деятельность по воплощению проекта в жизнь.

Эффективное использование модели группы MSF основывается на следующих принципах организации команды соратников: сфокусированность на нуждах заказчика; нацеленность на конечный результат; установка на отсутствие дефектов; стремление к самосовершенствованию; заинтересованность в результате. Модель группы MSF основана на постулате о шести качественных целях, достижение которых определяет успешность проекта. Эти цели определяют модель проектной группы, которая представлена на рис. 3.

Рис. 3. Модель группы MSF

Каждый из ее ролевых кластеров (или ролей) ассоциирован с одной из шести целей и работает над ее достижением. Роли продуманы таким образом, что все главные сферы проекта оказываются в зоне внимания на протяжении всего процесса разработки. Роли и их ответственности приведены в табл.. 2.

Таблица 2

Ответственности ролевых кластеров

Роль	Ответственность
Управление программой (program manager)	За разработку архитектуры решения, административные службы.
Разработка (developer)	За разработку приложений и инфраструктуру, технологические консультации
Тестирование (QAE	За планирование, разработку тестов и отчетность по тестам
Управление выпуском (release manager)	За инфраструктуру, сопровождение, бизнесы-процессы, выпуск готового продукта
Удовлетворение заказчика (user experience)	За обучение, эргономику, графический дизайн, техническую поддержку
Управление продуктом (product manager)	За бизнес-приоритеты, маркетинг, представительство интересов заказчика

Количество людей в группе определяется объемом работ, предусмотренным в проекте. Лидеры (менеджеры) групп координируют работу всех членов группы и выступают в качестве консультантов и наставников группы, а не выполняют функции контроля над ней, что позволяет каждому члену группы иметь необходимые полномочия для выполнения своих обязанностей и эффективно взаимодействовать с остальными членами группы.

В малых проектных группах некоторые роли могут совмещаться, но совмещение роли команды разработчиков, ни с какой другой ролью не допускается. Также необходимо избегать совмещения ролей, имеющих предопределенные конфликты интересов.

Модель группы **MSF** предлагает разбиение больших команд (более 10 человек) на малые многопрофильные группы направлений (feature teams), что отображено на рис. 4. Эти малые коллективы работают параллельно, регулярно синхронизируя свои усилия. Кроме того, когда ролевому кластеру требуется много ресурсов, формируются т. н. функциональные группы (functional teams), которые затем объединяются в ролевые кластеры.

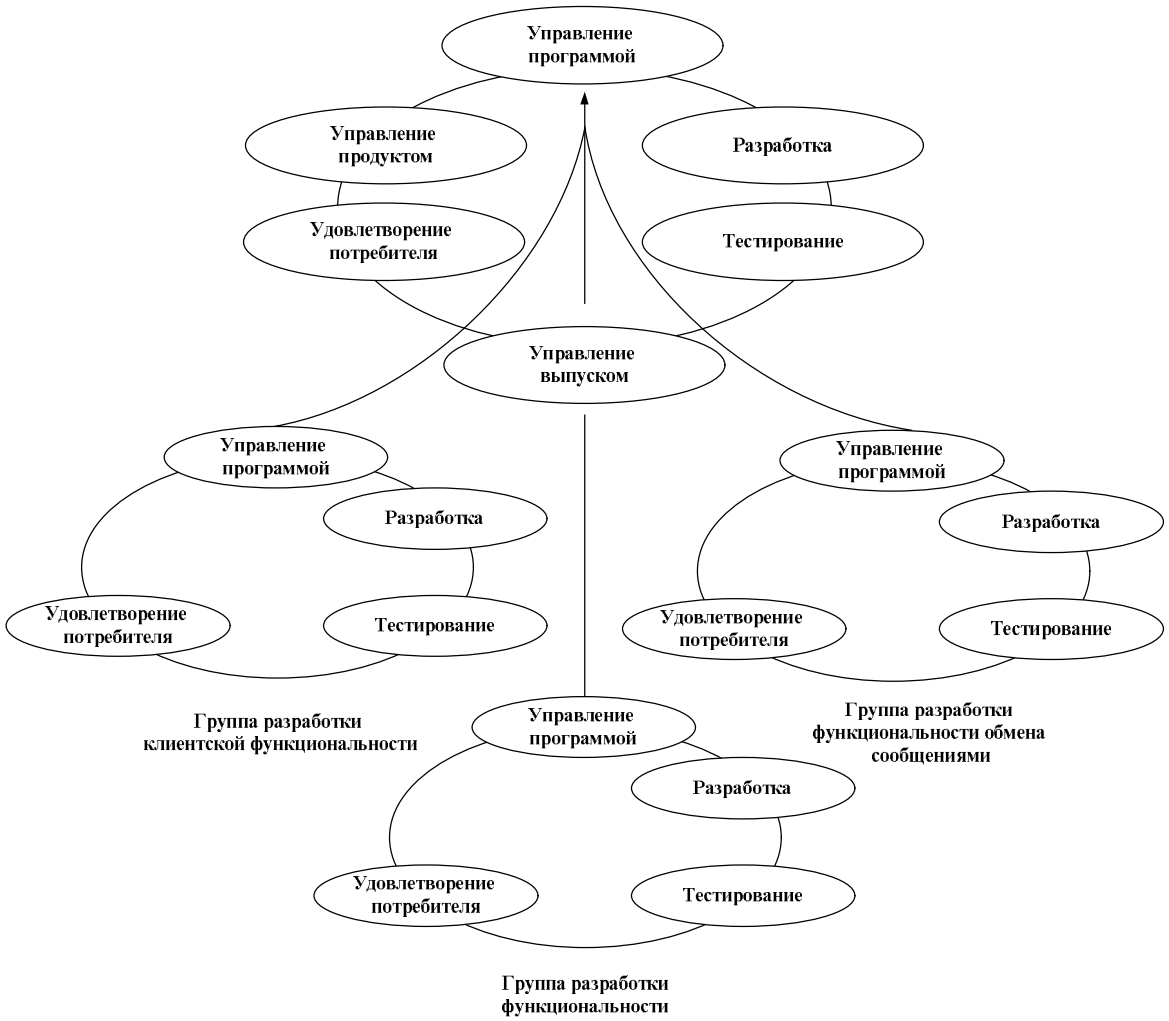

Рис. 4. Группы направлений

Использование ролевых кластеров не подразумевает и не навязывает никакой специальной структуры организации или обязательных должностей. Административный состав ролей может широко варьироваться в разных организациях и проектных группах. Чаще всего роли распределяются среди различных подразделений одной организации, но иногда часть их отводится сообществу потребителей или внешним по отношению к организации консультантам и партнерам. Ключевым моментом является четкое определение работников, ответственных за каждый ролевой кластер, их функций, ответственности и ожидаемого вклада в конечный результат.

Достоинства модели группы MSF:

– высокая производительность;

– сравнительно легкая масштабируемость;

– высокая мотивация труда и заинтересованность всех членов группы конечном успехе.

Недостатки модели MSF:

– для формирования команды нужны специалисты равной квалификации и одинаковой заинтересованности в успехе проекта;

– важное значение имеет коммуникабельность и умение работать в коллективе;

– демократичная модель команды MSF плохо сопрягается с жесткой иерархической структурой предприятия. [19]

4.5 Модель группы FDD

Функционально-ориентированная разработка (Feature Driven Development, FDD) оперирует понятием функции или свойства (feature) системы, достаточно близким к понятию сценария использования.

Технология разработки ПО FDD предполагает наличие шести основных ролей, но каждый участник проекта может выполнять одну или несколько ролей.

Менеджер проекта (Project Manager) является административным лидером проекта и отвечает за управление бюджетом проекта и распределение ресурсов. На плечах менеджера проекта также лежит представление отчетов о состоянии проекта руководству организации.

Главный архитектор (Chief Architect) отвечает за общий дизайн системы. Во время коллективных сессий, посвященных проектированию системы, главный архитектор выступает в ролях эксперта и посредника. Также главному архитектору принадлежит исключительное право принятия решений в спорных ситуациях.

Менеджер разработки (Development Manager) отвечает за управление процессами разработки. Обязанностью менеджера разработки является обеспечение эффективного использования всех ресурсов проекта. В частности, менеджер разработки выступает в роли арбитра в организационных спорах между главными программистами (например, споры о разделение ресурсов проекта между функциональными командами).

Ведущими программистами (Chief Programmers) назначаются опытные разработчики, которые хотя бы однажды прошли через все этапы жизненного цикла проекта. Ведущие программисты участвуют в процессах высокоуровневого анализа требований и проектирования. Ведущие программисты также возглавляют небольшие функциональные команды (от трёх до шести человек) и руководят локальными процессами анализа требований, проектирования и реализации. Они взаимодействуют друг с другом для решения повседневных локальных технических и организационных проблем.

Владельцем класса (Class Owners) является разработчик, который отвечает за проектирование и реализацию некоторого конкретного класса. В технологии разработки ПО FDD не используется распространенная практика коллективного владения кодом. При реализации функциональных возможностей, которые затрагивают несколько классов, соответствующие владельцы классов объединяются в функциональные команды и работают под началом ведущего программиста.

Эксперты в предметной области (Domain Experts) должны обладать глубоким знанием предметной области. Эксперты играют роль основных источников подробной информации обо всех аспектах предметной области системы. Эксперты предметной области постоянно взаимодействуют с другими участниками проекта и активно участвуют в таких процессах как анализ требований, проектирование и тестирование разрабатываемого продукта [17].

Достоинства использования модели группы FDD:

– позволяет управлять проектами до 500 человек;

– привлекать к работе менее квалифицированных разработчиков;

– эффективно контролировать процесс разработки.

4.6 Модель группы OpenUp

Технология разработки ПО OpenUp позиционируется авторами как легкий гибкий вариант RUP, в основу которого положены следующие принципы:

– нахождение компромисса с целью максимального удовлетворения заинтересованных лиц;

– постоянное общение с целью сохранения общности интересов и понимания;

– постоянное развитие в процессе получения обратной связи и проведения улучшений;

– концентрация на четко сформулированной архитектуре [18].

Повышение мотивации сотрудников к работе в группе осуществляется за счет ее самоорганизации, что позволяет членам группы:

– коллективно планировать и распределять обязанности;

– самостоятельно, выдвигать свои кандидатуры на исполнение роли;

– давать видение своего участия в проекте, как в качестве роли, так и в качестве члена коллектива.

Для эффективной реализации самоорганизации и устранения преград в процессе разработки ПО членам группы помогает инструктор. Считается, что инструктором должен быть руководитель проекта, которому придется отказаться от директивно-контролирующего стиля управления и стать для членов группы наставником и помощником.

Участники группы выполняют ниже перечисленные роли.

Пользователи представляют интересы групп, заинтересованных в разработке программного обеспечения. Эту роль может исполнять любой заинтересованный в проекте человек (клиент, заказчик, инвестор, пользователи ПО и др.).

Менеджер проектов ведет планирование проекта в сотрудничестве с пользователями и группой, координирует их взаимодействие, контролирует процесс разработки с точки зрения достижения цели проекта.

Аналитик взаимодействует со всеми заинтересованными лицами (заказчиком, пользователями, разработчиками и др.) в реализации проекта, собирает данные от них, расставляет приоритеты для требований пользователей.

Архитектор отвечает за проектирование архитектуры программного обеспечения, принимает ключевые технические решения.

Разработчик отвечает за проектирование, реализацию и тестирование компонент, объединение компонент системы.

Тестер планирует и осуществляет тестирование системы, регистрирует и анализирует результаты тестирования.

Любая Роль исполняется любым человеком в группе, который может выполнять общие задачи проекта.

Использование технологии разработки ПО OpenUp предполагает разделение проекта на итерации – планируемые и ограниченные во времени интервалы, длительность которых измеряется неделями. Каждая итерация начинается с совещания, посвященного планированию итерации. На этом совещании коллектив самостоятельно решает вопросы определения и выполнения задач итераций и передачи результата.

Большая часть времени в процессе итерации отводится на выполнение микрошагов. Микрошаг представляет собой результат работы (продолжительностью от нескольких часов до нескольких дней) одного человека или нескольких человек, участвующих в совместной деятельности по достижению целей итерации.

Концепция микрошагов помогает отдельным членам коллектива разбить свою работу на малые единицы, чтобы каждый мог внести измеряемый вклад в работу коллектива. Микрошаги обеспечивают исключительно короткие циклы обратной связи, которые способствуют принятию адаптивных решений в ходе каждой итерации.

Для обсуждения хода выполнения микрошагов и состояния дел команда проводит ежедневное «собрание стоя» с участием всех членов. Проблемы на ежедневном собрании не обсуждаются. На выходе каждого микрошага должен генерироваться код для сборки, а также проверенные артефакты, а результатом итерации является версия, работу которой можно продемонстрировать или передать для ознакомления всем заинтересованным лицам

Вся работа назначается, записывается и отслеживается с помощью Списка работ (Work Item List). Участники команды используют его в качестве единого репозитория для всех задач, которые необходимо записать и отслеживать, включая все запросы на изменения, дефекты и требования пользователей.

Технология предназначена для небольших команд, не распределенных географически. Необходимым условием эффективной работы является возможность ежедневного общения лицом к лицу. Достоинства модели группы OpenUp: прозрачность процесса разработки программного обеспечения; понимание членами коллектива работы других [21].

Выводы

Обзор существующих моделей организации коллективной разработки ПО позволил выявить разнообразные модели организации групп, структура и взаимодействие в которых определяются, в основном, технологией процесса разработки ПО.

Выявленные особенности организации коллективной разработки ПО могут стать основой для разработки содержания соответствующих тем учебной дисциплины «Групповая динамика и коммуникация», т.к. знание функциональных ролей групп и отдельных членов в группе позволит сформировать у будущих инженеров системное представление о разрабатываемом продукте; условиях разработки в целом; понять разделение функций между сотрудниками; осмыслить должностную иерархию; подчиненность; осознать характер деятельности в группе.

Изучение особенностей моделей организации коллективной разработки ПО в рамках данной дисциплины позволит студентам на начальных курсах ознакомиться с профессиональной терминологией технологий разработки ПО, развить у студентов понимание значимости технологий до начала их углубленного изучения на старших курсах.

Список литературы

1. Амблер С. Гибкие технологи: Экстремальное программирование и унифицированный процесс разработки / С. Амблер. – Спб.: Питер. 2005. – 412 с.

2. Бек К. Экстремальное программирование / К. Бек. – Спб.: Питер, 2002.

3. Брукс Ф. Мифический человеко-месяц, или как создаются программные системы / Ф. Брукс. – Пер. с англ. – СПб.: Символ-Плюс, 2006. – 304 с.

4. Кириленко Е.Г. Обоснование содержания обучения в рамках методологии преподавания профессионально-ориентирован-ной дисциплины «Групповая динамика и коммуникация» / Е.Г. Кириленко, О.В. Лучшева // Інженерія програмного забезпечення. – 2010. – № 1. – С.71-78.

5. Константин Л. Человеческий фактор в программировании / Л. Константин. – Пер. с англ. – СПб.: Символ-Плюс, 2004. – 384 с.

6. Липаев В.В. Методы обеспечения качества ПС / В.В. Липаев. – М.: Теис, 2003. 485 с.

7. Салливан Э. Время – деньги. Создание команды разработчиков программного обеспечения. Пер. с англ. / Э. Салливан - М.: Издательско-торговый дом «Русская Редакция», 2002. – 368 с.

8. Соммервилл И. Инженерия программного обеспечения. Пер. с англ. / И. Соммервилл. – М.: Вильямс, 2002. – 624 с.

9. Шнейдерман Б. Психология программирования: Человеческие факторы в вычислительных и информационных системах. Пер. с англ. / Б. Шнейдерман. – М.: Радио и связь, 1984. – 304 с.

10. Якобсон А. Унифицированный процесс разработки программного обеспечения / А. Якобсон, Г. Буч, Дж. Рамбо– СПб.: Питер, 2002. – 496 с.

11. Benjamin J. J. Voigt. Dynamic System Development Method. Zürich, Switzerland. Department of Information Technology University of Zurich. 2004.

12. Metzger, Phillip W.,Managing a Programming Project, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

13. Schwaber, Ken; Beedle, Mike (2002). Agile software development with Scrum. Prentice Hall.

14. Stapleton, J. (1997) DSDM: Dynamic Systems Development Method. Harlow, England: Addison-Wesley.

15. Manifesto for Agile Software Development [Электронный ресурс] / Д. Слиньков – Режим доступа: http://agilemanifesto.org

16. Scrum (development) [Электронный ресурс] – Режим доступа: http://en.wikipedia.org/wiki/Scrum_(development)

17. Feature Driven Development [Электронный ресурс] – Режим доступа: http://en.wikipedia.org/wiki/Feature_Driven_Development.

18. Eclipse Process Framework Project (EPF) [Электронный ресурс] – Режим доступа к ресурсу: www.eclipse.org/epf.

19. Welcome to INSONO Consulting [Электронный ресурс] – Режим доступа: www.insono.com.

20. Microsoft Solutions Framework [Электронный ресурс] – Режим доступа к ресурсу: http://ru.wikipedia.org/wiki/Microsoft_Solutions_Framework.

21. OpenUP [Электронный ресурс] – Режим доступа к ресурсу: http://en.wikipedia.org/wiki/OpenUP.

Поступила в редколлегию 23.01.2012

Рецензент: д-р техн. наук, проф. И.В. Шостак, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.

МОДЕЛІ ОРГАНИЗАЦІЇ КОЛЕКТИВІВ РОЗРОБНИКІВ В ГАЛУЗІ ПРОГРАММНОЇ ІНЖЕНЕРІЇ

О.Г. Кіріленко, Г.О. Фролова

У статті досліджено особливості організації колективів розробників програмного забезпечення. Визначено, що важливим завданням в проекті є розподіл ролей і функцій в групі, а також організація взаємодії як усередині групи, так і між групами. Виявлено характеристики традиційних і гнучких моделей організації груп розробників програмного забезпечення (ПЗ). Доведено, що структура групи та взаємодія в ній визначаються, в основному, технологією процесу розробки ПЗ. Виявлені особливості організації колективної розробки ПЗ можуть стати основою для розробки змісту відповідних модулів навчальної дисципліни «Групова динаміка і комунікація».

Ключові слова: *програмна інженерія, групова динаміка та комунікація, моделі організації колективів розробників програмного забезпечення, технології процесів розробки програмного забезпечення.*

ORGANIZATION MODELS OF SOFTWARE DEVELOPMENT TEAMS

O.G. Kirilenko, G.A. Frolova

Review of existing models of the collective software development organization has revealed a variety of models of group organization, structure and interactions that define, in general, technology of the software development process. These features of collective organization of software development can be the basis for developing the content of the modules of discipline, "Group dynamics and communication," because knowledge of the functional roles of groups and individual members of the group will develop among future engineers a systemic understanding of the developed product, to understand the division of functions between the staff, to understand the job hierarchy, subordination, to understand the nature of the group.

Keywords: *software Engineering, software development group, the traditional model teams organization, agile development.*