УДК 004.02

О ФАКТОРАХ РИСКА В ПРОЦЕССЕ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Д. А. Макаров, М. Я. Розенберг, А. Б. Шильников

RISKS IN SOFTWARE DEVELOPMENT PROCESS

D. A. Makarov, M. Y. Rozenberg, A. B. Shilnikov

Дается представление о факторах риска, возникающих в процессе разработки программного обеспечения, описывается их возможное негативное влияние на проекты по разработке ПО. Приводится краткий обзор видов рисков, рассматриваются источники их возникновения, а также методы их выявления и минимизации. Указаны актуальные проблемы, связанные с факторами риска.

Ключевые слова: программная инженерия, программное обеспечение, процесс разработки, технические риски, управленческие риски, управление рисками, факторы риска

Software development risks and their influence on software development process have been considered. Sources of software development risks and methods of their revelation and minimization are described. Topical issues related to risk factors are outlined.

Keywords: software, software engineering, software development process, technical risks, managerial risks, risk management, risk factors

Введение

Несмотря на постоянный и значительный прогресс, наблюдаемый в индустрии программного обеспечения (ПО) на протяжении последних десятилетий, следует признать, что разработка ПО в заданные сроки с заданным качеством и с заданным бюджетом попрежнему представляет собой острую и актуальную проблему. Так, по данным [1] около двух третей от общего числа программных проектов в настоящее время либо полностью проваливаются, либо не завершаются в установленные сроки, либо не реализуют изначально запланированную функциональность. Исследование [1] со всей очевидностью показывает комплексный характер данной проблемы, решение которой требует такого же комплексного подхода. Научное направление, изучающее способы эффективного создания ПО с определенной функциональностью и в рамках приемлемых сроков и бюджета известно как программная инженерия (software engineering).

Одним из важнейших понятий программной инженерии является процесс разработки программного обеспечения, который можно охарактеризовать как совокупность некоторых процедур, выполняемых при разработке ПО. При этом часть таких процедур направлена непосредственно на конструирование проекта (разработка архитектуры, детальное проектирование, написание кода), а прочие – на управление им. В настоящее время известно несколько десятков различных процессов разработки ПО, каждый из которых продолжает эволюционировать [2].

Разработка программного обеспечения представляет собой, в общем случае, недетерминированный процесс, который сопровождается большим количеством неопределенностей (человеческий фактор, нестабильные требования и т.п.), способных оказывать резко негативное влияние на сроки выпуска и качество ПО. Такие неопределенности по своей сути являются для проекта факторами риска, которые необходимо по возможности минимизировать. В связи с этим важным элементом процесса разработки ПО является т.н. управление рисками (risk management) – деятельность, направленная на выявление и минимизацию рисков [3]. Управление рисками в рамках проекта по разработке ПО требует определенных дополнительных ресурсов, однако если ему уделяется недостаточно внимания, то проект будет разрабатываться в условиях неопределенности и непредсказуемости, возможный ущерб от которых способен вывести проект за рамки бюджета и расписания или же и вовсе привести к его сворачиванию.

В данной статье рассматриваются наиболее типичные для среднестатистического программного проекта риски, причины их возникновения, известные на настоящий момент подходы к их выявлению и минимизации, а также возникающие при использовании этих подходов проблемы.

1. Источники рисков

При разработке программного обеспечения разработчикам приходится ежедневно принимать тысячи решений различной степени значимости, причем число уровней абстракции, по которым распределяются эти решения, может доходить до девяти [4]. По причине высокой сложности и значительной нагрузки на специалистов, программная инженерия является одной из немногих технических областей, в которых человеческие ошибки происходят постоянно и являются неотъемлемой составляющей всего процесса трудовой деятельности.

Ошибки, допускаемые разработчиками, можно оценивать по степени причиняемого ими ущерба, выражаемого в расходе ресурсов (материальных, временных, трудовых и пр.) на поиск и исправление этих ошибок, а также на устранение их последствий. Кроме того, в зависимости от специфики разрабатываемого проекта и условий конкретной организации, определенные категории ошибок могут возникать чаще или реже других, то есть имеет смысл классифицировать ошибки по частоте их появления.

При сопоставлении средней частоты появления ошибки заданного типа и возможного ущерба, который она может принести, формируется величина, называемая техническим риском (technical risk). Технический риск является наиболее качественной мерой относительной значимости ошибки, поскольку с его помощью можно оценить максимальный ущерб от систематического возникновения ошибки определенного типа. Помимо технических рисков, выделяют также управленческие риски (managerial risks). Этот тип рисков используется для описания проблем, связанных с непредвиденными расходами, текучестью кадров, возможностью незавершения проекта в срок, на который брала обязательства организация, и т.п. [5].

2. Методики выявления рисков

Выявление рисков – это ключевой вид деятельности в любом процессе разработки, поскольку игнорирование рисков ставит под угрозу исход всего программного проекта [6]. Качественное выявление рисков предполагает наличие точных данных о динамике развития проекта, которые могут быть получены за счет постоянного мониторинга состояния проекта, а также за счет организации эффективной обратной связи со всеми, кто имеет к нему отношение.

Мониторинг состояния проекта может производиться по множеству различных метрик. Исследователи выделяют пять основных величин, которые могут предоставить необходимую информацию для выявления абсолютного большинства технических рисков: размер реализованных фрагментов программы в строках кода (LOC), производительность (LOC/час), время, расходуемое на различные виды деятельности, объем проделанной работы (в человеко-часах) и надежность программы (число раскрытых ошибок за период времени) [7]. Сбор информации может также проводиться и по любым другим метрикам (например, по т.н. микровехам), но в этом случае существует вероятность, что накладные расходы на их сбор превзойдут полезный эффект от улучшения «видимости»рисков. Следует заметить, однако, что на рынке ПО существуют автоматизированные утилиты (например, SEER-PPMC), осуществляющие эффективный сбор информации как по пяти основным, так и по большому количеству вспомогательных метрик.

Некоторые управленческие риски выявляются не с помощью сбора и мониторинга данных по различным метрикам, а путем организации обратной связи со всеми людьми, имеющими отношение к проекту. Во многих случаях созданию обратной связи мешает традиционно негативное отношение менеджеров к «плохим новостям», которыми они считают информацию о высоких рисках. Тем не менее, многие ведущие организации осознают ценность подобного рода информации, и специально в целях повышения эффективности создают анонимный канал для уведомлений о рисках [3]. Наиболее эффективная обратная связь может быть создана, если организация концентрирует свои усилия на достижении и поддержании «атмосферы высокого доверия»между всеми участниками проекта, включая менеджеров [8].

Чтобы в любой момент времени сведения обо всех выявленных рисках проекта были актуальными и общедоступными, многие организации создают специально для учета рисков некоторый формализованный документ (например, «Список 10 основных рисков») и закладывают в план проекта регулярный его пересмотр [9].

3. Методики минимизации рисков

Деятельность, направленная на выявление и регистрацию рисков, занимает лишь небольшую часть процесса разработки программного обеспечения. Основные трудозатраты по управлению процессом разработки ПО приходятся на минимизацию рисков, представляющих для проекта наибольшую угрозу. Для минимизации различных видов рисков существует большое количество методик, рассматривать которые удобнее всего в контексте разделения рисков на категории.

3.1. Управленческие риски

Управленческие риски могут включать в себя негативное влияние условий труда, недостаточную надежность аппаратного обеспечения, низкую эффективность программного инструментария и др., но, даже если перечисленные риски существуют, они, как правило, редко выходят на первый план.

Одни из самых серьезных управленческих рисков связаны с комплектацией персонала. Поскольку новые работники будут работать медленно и допускать ошибки, пока не обучатся, полная стоимость замены опытного компьютерного специалиста оценивается в 20–100 тыс. долларов [10]. Результаты исследований в области программной инженерии показывают, что «лучшие»программисты опережают по производительности «худших»в 10 раз [11]. Исходя из этого, если в организации присутствует разработчик из числа «худших», то он будет являться источником массы проблем, и, фактически, его присутствие будет представлять собой весомый риск для всего проекта.

Достаточно серьезными можно считать риски возможного превышения сроков и бюджета, хотя к таким рискам всегда необходимо относиться с особым вниманием. Если сроки и бюджет адекватны поставленной задаче, то риск их превышения вторичен и является следствием уделения слишком малого внимания другим рискам. Если же сроки и бюджет неадекватно малы, то это означает, что организация взяла на себя обязательства, основываясь на слишком неточной оценке. В таком случае для минимизации риска необходимо либо пересмотреть оценку сроков и бюджета, либо уменьшить объем проекта – пытаться теми или иными административными мерами заставить разработчиков уложиться одновременно в установленные сроки, бюджет и функциональный объем не только неэффективно, но и попросту вредно, поскольку это значительно увеличивает риск выпуска продукта неприемлемого качества. По статистике, программа, разработанная в условиях чрезмерного давления сроков, содержит в 4 раза больше ошибок, при этом давление сроков присутствует в 75–100% проектов [12].

Риск выпуска продукта неприемлемого качества является наиболее значительным из всех управленческих рисков, поскольку его минимизация предполагает кардинальные административные меры, вплоть до свертывания всего проекта. Это риск напрямую зависит от типа разрабатываемой программной системы и от специфических требований, предъявляемых к ней. Так, например, для типичной бизнес-системы приоритетной может быть скорость ее выпуска, возможно даже с некоторым ущербом качеству, в то время как для систем реального времени и встраиваемых систем требования к надежности, напротив, являются доминирующими. Игнорирование рисков, связанных с качеством, может привести к потере престижа организации, выходу из строя дорогостоящего оборудования и даже к гибели людей. К примеру, разрушение европейского ракетоносителя Ariane 5 вследствие ошибки в регулировочном коде встроенного ПО привело к потере более 370 млн долларов [13].

Таким образом, минимизация управленческих рисков не предполагает использования каких-либо формальных методик, и при их разрешении важна только квалификация и решительность руководителя. При неэффективном менеджменте неразрешенные управленческие риски приведут к появлению технических рисков, на борьбу с которыми уйдет немалое количество денег, времени и трудозатрат.

3.2. Технические риски, не связанные со сложностью разрабатываемого ПО

Некоторые категории ошибок возникают не из-за того, что разработчики не смогли справиться со сложностью задачи и приняли неверное проектное решение, а просто по невнимательности, вследствие «человеческого фактора».

Если такого рода ошибки возникают нечасто и приводят к незначительному ущербу, то и риски, связанные с такими ошибками, невелики и, как правило, не минимизируются. Примером такого класса ошибок могут являться опечатки в коде программы, которые легко обнаруживаются и устраняются при следующей сборке проекта.

С другой стороны, некоторые категории нечасто возникающих ошибок могут нанести проекту значительный ущерб: например, разработчик может случайно повредить или удалить файл с исходным кодом. Хотя такая ситуация происходит нечасто, она представляет для длительного проекта значительный риск, который нельзя игнорировать. Минимизация рисков такого типа производится путем введения в деятельность организации некоторой политики безопасности, включающей в себя разграничение доступа к информации, периодическое резервирование всех относящихся к проекту файлов и т.д. В настоящее время все эти задачи эффективно решаются системами контроля исходного кода (например, Microsoft

Visual SourceSafe), поэтому практически не сопряжены с дополнительными трудозатратами.

3.3. Технические риски, связанные с недостаточной квалификацией персонала

Поскольку каждая программная разработка по-своему уникальна, и эффективность борьбы с ее сложностью в значительной степени зависит от квалификации программиста, распространенным источником ошибок является периодически возникающая необходимость отправлять разработчиков завершать работу, начатую другими людьми. Разработчики, пытающиеся внести изменения или исправления в незнакомый код, будут допускать относительно большое количество ошибок до тех пор, пока полностью не разберутся в деталях поставленной перед ними задачи. Хотя ущерб от каждой из такого рода ошибок может быть незначительным, большое их количество в любом случае представляет для проекта большой риск.

Если объективной необходимости для перераспределения разработчиков на незнакомые задачи не было, то данный технический риск вторичен и порожден управленческими изъянами, поэтому достичь его эффективной минимизации возможно путем корректировки административных мер. Однако, вне зависимости от этого, технический риск такого типа может и должен быть минимизирован напрямую, и для этого придумана масса разнообразных методик. Суть всех этих методик заключается в повышении «прозрачности» проекта: чем более четко структурирован проект и чем более ясно и недвусмысленно изложены его исходный код и документация, тем меньше времени разработчик потратит на его изучение и тем меньше ошибок он допустит в конечном итоге. Данная деятельность весьма важна: по статистике, сопровождение кода проекта производит до 10 поколений программистов [14].

Чтобы проект получился четко структурированным, необходимо на всех «конструкторских» стадиях (разработка архитектуры, детальное проектирование, кодирование) планомерно бороться с его нарастающей сложностью. Эта деятельность включает в себя построение наилучшей декомпозиции проекта на подсистемы, модули, классы, функции, назначение понятных имен идентификаторам, написание комментариев, отказ от сложных языковых конструкций и многое другое [4]. Недостаточная борьба со сложностью увеличивает риски проекта не только в случае, когда на сопровождение кода назначается новый человек, но и при непосредственном «штатном» написании, поскольку с модификацией большого объема сложного кода может не справиться даже сам автор этого кода.

Чтобы достичь большей ясности исходного кода в условиях, когда разработкой занимается много программистов, большинство организаций создают собственные стандарты кодирования. Главная цель введения стандарта кодирования заключается в достигаемом с его помощью единообразии исходного кода [3], за счет которого минимизируется вероятность возникновения ошибок при перераспределении программистов.

Наконец, ускорить обучение программистов незнакомым задачам позволяет документация. Требования к документации просты: она должна быть недвусмысленной и информативной. Информативность предполагает возможность легко обнаружить в документации необходимые сведения о концепции проекта, архитектуре, декомпозиции на классы, о структуре базы данных и т.д., а недвусмысленность достигается путем использования определенных стандартов представления информации, понятных всем разработчикам. Для разных типов программного обеспечения стандарты представления информации могут быть разные: например, для объектно-ориентированных систем стандартом является язык моделирования UML (Unified Modeling Language), а для систем со структурной декомпозицией – функциональные диаграммы SADT (Structured Analysis and Design Technique), диаграммы потоков данных DFD (Data Flow Diagrams) и диаграммы «сущности-связи» ERD (Entity-

Relationship Diagrams). Структура, объем и степень детализации прочей проектной документации также во многих случаях регламентируются стандартами, принимаемыми в каждой конкретной организации. Например, чтобы максимально оградить проекты от человеческого фактора, в Министерстве Обороны США для 100 тыс. строк программного кода создается около 40 тыс. страниц документации [15].

3.4. Технические риски, связанные с поздним исправлением ошибок

В отличие от ошибок, допускаемых в исходном коде, возникающие на ранних стадиях проекта (при сборе требований, разработке архитектуры, детальном проектировании) ошибки могут представлять собой весомый риск для проекта не только за счет многочисленности, но и за счет значительного причиняемого ими ущерба. Риск, связанный с такими ошибками, возрастает со временем: чем позже обнаруживается ошибка, допущенная на ранней стадии проекта, тем большим будет ущерб, связанный с устранением ошибки вместе со всеми ее последствиями. Статистика утверждает, что стоимость позднего исправления ошибки, внесенной в проект на ранней стадии, в 50–200 раз выше стоимости исправления этой же ошибки сразу после ее внесения [16].

Минимизация данного риска лучше всего достигается за счет раннего обнаружения ошибок. Чтобы обнаружить большинство ошибок как можно раньше, организации подвергают все промежуточные продукты (список требований, проектную документацию, исходный код) тщательному коллективному инспектированию и рецензированию. Для крупных проектов техническое рецензирование является одной из ключевых методик, поскольку его эффективность может быть до 20 раз эффективнее тестирования [17].

Обнаружить при помощи рецензирования удается лишь около 60% всех ошибок, а все остальные должны быть найдены на поздней стадии проекта при помощи системного тестирования [4]. Нахождение ошибок на поздней стадии также можно считать мерой по уменьшению рисков, поскольку если ошибка не будет найдена вовсе и попадет в окончательный продукт, то затраты на ее исправление и на поддержку пользователей окажутся несоизмеримо большими. Правильными практиками системного тестирования являются заблаговременное построение эффективного тестового набора, покрывающего 100% кода программы, ежедневное тестирование герметичности («дымовой» тест), как можно более частое интеграционное тестирование и др. Если же к качеству системы предъявляются повышенные требования, то оправдывает себя выделение значительных ресурсов на обеспечение тестирования. К примеру, при разработке ПО для космического шаттла количество тестировщиков превосходило количество программистов в 10 раз [18].

Как можно догадаться, наиболее серьезной угрозой для проекта являются ошибки, допущенные на самой первой стадии проекта – при сборе требований. Эти ошибки могут, в случае их несвоевременного обнаружения, оказывать на проект наиболее длительный и масштабный негативный эффект, и, в то же время, в отличие от ошибок, допускаемых на остальных стадиях, не могут быть выявлены ни рецензированием, ни тестированием. По этой причине, риски, связанные с изменением набора требований к ПО, очень велики, и их минимизация редко бывает неактуальной.

Тем не менее, далеко не все меры, которые необходимо предпринять для исчерпывающего сбора требований, осуществимы на практике. Большинство пользователей не имеют технического образования и не обладают достаточной квалификацией, чтобы четко и полно сформулировать требования в терминах, понятных программистам. Программисты в таких случаях пытаются общаться в терминах, понятных пользователям: представляют им прототип пользовательского интерфейса, диаграмму вариантов использования и пр., однако и эти меры в большинстве случаев не позволяют уменьшить риск до приемлемого значения.

Единственный способ получить гарантированную минимизацию рисков, возникающих от изменения требований – это организация итеративного процесса разработки ПО, при котором на каждой итерации для некоторой отдельной части программы производится весь «водопадный» цикл от сбора требований до выпуска. В контексте взаимодействия с пользователями, итеративный процесс можно рассматривать как многоэтапную поставку: по завершению каждой итерации пользователи получают промежуточный рабочий вариант программы и возможность, анализируя его работу, скорректировать и дополнить свои требования. Чем выше риск изменения требований, тем больше должно быть спланировано итераций. Если риск слишком высок, наиболее эффективным становится т.н. экстремальное программирование: модель процесса, характеризующаяся предельно короткими итерациями (2 – 3 недели) и постоянной обратной связью с конечными пользователями [19]. Поскольку разбиение на множество итераций достаточно ресурсоемко, при средней степени риска изменения требований применяют гибридные модели процесса [1], сочетающие в себе как последовательную, так и итеративную части.

4. Актуальные проблемы управления рисками

Поскольку проекты различны по своему характеру и разрабатываются в различных условиях, методики выявления и минимизации рисков, рассмотренные в данной статье, могут быть эффективными в большей или меньшей степени в зависимости от каждого конкретного случая. В то же время, область исследования эффективности применения методик в реальных ситуациях остается малоизученной: вопрос о том, какой выигрыш (в материальных средствах, времени, трудозатратах) принесет использование какой-либо методики в заданной конкретной ситуации в большинстве случаев остается без ответа. Фактически, именно это обстоятельство является основной причиной провалов программных проектов, либо выхода их за рамки бюджета и расписания.

В настоящее время такая оценка сильно затруднена отсутствием полноценного формального описания процесса разработки ПО. Созданию в достаточной мере адекватной математической модели процесса разработки ПО препятствует значительное влияние человеческого фактора: степень необходимости применения тех или иных методик управления рисками сильно зависит от квалификации программистов, их мотивации, формы взаимодействия друг с другом и т.д. Существующие методы учета человеческого фактора при разработке ПО, включающие в себя, как правило, один или несколько субъективно выставляемых числовых коэффициентов, явно недостаточны для получения оценок эффективности методик с высокой степенью точности. Таким образом, повышение точности формального описания процесса разработки ПО способно привести к значительному повышению эффективности программных разработок в целом. Исследования в этом направлении осуществляются в настоящее время ведущими мировыми институтами программной инженерии.

В этих условиях представляется весьма перспективным создание специализированных технологических средств, способных выполнять оценку последствий применения различных методик управления рисками в конкретных ситуациях. Полученная с помощью таких средств информация могла бы стать для менеджеров проекта важнейшим инструментом, поскольку позволила бы определять набор наиболее эффективных методик, выявлять оптимальную последовательность их внедрения в процесс разработки и т.д. Эта же информация могла бы использоваться для наилучшей адаптации хорошо зарекомендовавших себя методик под конкретные проектные условия, а также для создания новых эффективных методик выявления и минимизации рисков.

Литература

1. The Standish Group. Charting the Seas of Information Technology / The Standish Group. – Dennis, MA: The Standish Group, 2007.
2. Ramsin, R. Process-Centered Review of Object-Oriented Software Development Metodologies / R. Ramsin, R. Paige // ACM Computer Surveys. – February 2008. – V.40, №1.
3. Макконнелл, С. Остаться в живых. Руководство для менеджера программных проектов. Библиотека программиста / С. Макконнелл. – СПб.: Питер, 2006. – 240 с.: ил.
4. Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл. – М.: Рус. Редакция; СПб.: Питер, 2007. – 896 стр.: ил.
5. Manager's Handbook for Software Development, Revision 1. Document number SEL-84-101 / NASA Software Engineering Laboratory, Goddart Space Flight Center, Greenbelt, MD, 1990.
6. Gilb, T. Principles of Software Engineering Management / T. Gilb. – Wokingham, England: Addison-Wesley, 1988.
7. Putnam, L.H. Five Core Metrics / L. H. Putnam, M. Ware – New York, NY: Dorset House, 2003.
8. Мораско, Дж. IT-проекты: фронтовые очерки / Дж. Мораско – СПб: Символ-Плюс, 2007. – 384 с.: ил.
9. Boehm, B. W. Software Risk Management / B. W. Boehm. – Washington, DC: IEEE Computer Society Press, 1989.
10. Thomsett, R. Effective Project Teams: A Dilemma, A Model, A Solution / R. Thomsett // American Programmer. – July-August 1990. – P. 25 – 35.
11. Curtis, B. Substantiating Programmer Variability / B. Curtis // Proceedings of the IEEE. – V.69, №7. – P. 846.
12. Capers, J. Assessment and Control of Software Risks / J. Capers. – Englewood Cliffs, NJ: Yourdon Press, 1994.
13. Nuseibeh, B. Ariane 5: Who Dunnit? / B. Nuseibeh // IEEE Sofware. – May–June 1997. – P. 15 – 16.
14. Thomas, R. A. Using Comments to Aid Program Maintenance / R. A. Thomas // Byte. – May 1984. – P. 415 – 422.
15. Capers, J. Applied Software Measurement: Assuring Productivity and Quality, 2d Ed. / J. Capers. – New York, NY: McGraw-Hill, 1997.
16. Boehm, B. W. Understanding and Controlling Software Costs / B. W. Boehm, P. N. Papaccio // IEEE Transactions on Software Engineering. – October 1998. – V.14, №10. – P. 1462 – 1477.
17. Russell, G. W. Experience with Inspection in Ultralarge-Scale Developments / G. W. Russell // IEEE Software. – January 1991. – V.8, №1. – P. 25 – 31.
18. Basili, V. The Experience Factory: How To Build And Run One / V. Basili, F. McGarry // 17th International Conference on Software Engineering. – Seattle, WA. – April 1995.
19. Кент, Б. Экстремальное программирование / Б. Кент. – СПб: Питер, 2002. – 224 с.

Кафедра «ИВТиПМ»,
Читинский государственный университет
makarov@chitgu.ru

Поступила в редакцию 9 февраля 2009 г.