РАЗДЕЛ I. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Думанов П.А.

Моделирование системы производства программного обеспечения

Кубанский Государственный Технологический Университет

(Россия, Краснодар)

doi:10.18411/spc-12-05-2019-01

idsp: sciencepublic-12-05-2019-01

Производство ПО (программного обеспечения) – сложный многоэтапный процесс, требующий большого количества специалистов различного профиля. И естественной является практика выделения из общего штаба сотрудников более мелких структурных единиц – подразделений, специализирующихся на том или ином этапе процесса производства ПО.

Разработка – подразделение, в котором сосредоточены специалисты, занимающиеся непосредственно написанием кода.

Специалисты по обеспечению качества дают заключение о завершенности и готовности к эксплуатации разрабатываемого продукта (либо его части), а также обнаруживают дефекты и недочеты, контролируют их исправление силами Разработки.

В подразделении Интеграции сосредоточены специалисты, занимающиеся интеграцией программных модулей, настройкой интерфейсов взаимодействия и наладкой тестовой среды.

Также существует потребность в анализе документации, составлении требований для проведения тестирования, чем занимается Тестовая модель.

Все работы каждого из подразделений ведутся в рамках соответствующих задач, имеющих дату начала и завершения, а также выделенное на проведение работ количество часов.

Процесс тестирования разрабатываемого ПО, как правило, заключается в прохождении специалистами обеспечения качества тестовых случаев – формально описанных алгоритмов, созданных для проверки получения определенного результата в работе программы на определенных тестовых данных. Написание тестовых случаев находится в зоне ответственности Тестовой модели.

Рисунок 1. Схема процессов производства ПО.

Тестовый случай может находиться в одном из следующих состояний, выделенных на рисунке зелёным цветом (рис.1):

- Активен – тестовый случай готов для прохождения. При экстренном изменении кода разрабатываемого функционала в процессе тестирования, тестовый случай может быть сброшен в активное состояние из другого статуса;

- В процессе – специалист по обеспечению качества приступил к прохождению;
- Ошибка – во время прохождения тестового случая был обнаружен дефект в работе программы;
- Заблокирован – дефект, обнаруженный при прохождении другого тестового случая, мешает прохождению этого;
- Успешно – прохождения тестового случая завершено без ошибок.

Ошибка, обнаруженная в результате прохождения тестового случая, должна быть задокументирована в виде сущности «Дефект», содержащей детальное описание ошибки, тестовые данные, на которых она была получена, шаги для воспроизведения и, в случае необходимости, дополнительные материалы – снимки экрана и файлы регистрации систем (log-файлы).

Созданные дефекты назначаются специалистом обеспечения качества на подразделения из поддержки тестирования – Разработки, Тестовой модели или Интеграции с целью их исправления.

Дефект может находиться в одном из следующих состояний, выделенных на рисунке красным цветом (рис.1):

- Обнаружен - специалист обеспечения качества оформил сущность дефект, назначив его на соответствующее подразделение;
- Принят – специалист из соответствующего подразделения поддержки тестирования взял дефект в работу;
- Переназначен - специалистом из поддержки тестирования принято решение о том, что исправление дефекта возможно другим специалистом (в том числе – из другого подразделения);
- Отвергнут – специалисту из поддержки тестирования не удаётся воспроизвести дефект в связи с ошибкой автора дефекта либо устаревшими тестовыми данными;
- Исправлен - дефект устранен и ожидает ретеста;
- Ретест - проверка специалистом обеспечения качества исправления дефекта на тех же тестовых данных, что и при обнаружении;
- Закрыт – отсутствие ошибки подтверждено;
- Возвращен – отрицательный ретест, требуются дополнительные работы со стороны поддержки тестирования.

Разработка и тестирование – поэтапный процесс, разбитый на итерации (спринты, релизы), подразумевающие реализацию определенной порции функционала разрабатываемой системы. Однако в рамках одной итерации регрессионное тестирование, функциональное тестирование, починка дефектов и разработка нового функционала ведутся параллельно (рис.2).

Рисунок 2. Схема работ внутри итерации (спринта).

Данная схема диктует необходимость в жестком соблюдении сроков внутри этапов, поскольку несвоевременные правки функционала предыдущих спринтов могут оказать влияние на регресс и потребуют отката и повторного прохождения тестовых случаев, что, в свою очередь, негативно скажется на сроках и качестве тестирования текущего спринта.

Основным риском является несоблюдение, и, как следствие – смещение сроков, что требует дополнительных трудовых и, следовательно, финансовых затрат. Однако локализация источника риска – крайне сложная задача в условиях столь комплексной динамической системы.

В процессе задействовано большое количество подразделений со своей иерархией и руководством. И у каждого подразделения собственная специфика управления и ведения отчетности, что в результате ведет к разрозненности информационного пространства подразделений, невозможности оценить общую картину состояния системы производства ПО, несвоевременным или неверным управленческими решениями и, как следствие – несоблюдению сроков, финансовым потерям.

Одним из основных источников рисков является неграмотное планирование процесса разработки. Процесс планирования подразумевает расчет необходимого количества итераций и распределение внутри них количества реализуемых пользовательских историй (User Story).

Пользовательские истории (U.S.) — способ описания требований к разрабатываемой системе, сформулированных как одно или более предложений на повседневном или деловом языке пользователя. Одна пользовательская история описывает изолированную часть разрабатываемого функционала (Например, «Я как Пользователь хочу осуществлять авторизацию в системе», «Я как Администратор хочу осуществлять настройку языка интерфейса»). Пользовательские истории могут серьезно различаться по трудоемкости реализации.

На этапе планирования необходимо обозначить, какие пользовательские истории будут реализованы в рамках того или иного спринта. Однако неверная оценка трудоемкости реализации пользовательских историй или необходимого количества спринтов могут привести к излишней нагрузке на команду разработки и несоблюдению сроков.

Также, в случае, когда разработка не успевает реализовать одну или несколько U.S. в рамках текущего спринта, их откладывают на следующий спринт, либо планируют «горячие правки» - реализацию нескольких, часто не связанных друг с другом U.S., вне спринтов. Такая мера крайне негативно сказывается на общем процессе производства ПО, поскольку требует привлечения сотрудников обеспечения качества, занятых плановыми работами над текущим спринтом, для дополнительного функционального тестирования и повторного прохождения регрессионных тестов в рамках нескольких прошедших спринтов.

Процесс планирования осуществляется со стороны Разработки, а разрозненность подразделений не позволяет взять в расчет время, необходимое на проведение работ прочими подразделениями. Также проблема коммуникации подразделений распространяется и на процессы, следующие за планированием. Часты случаи, когда информацию об изменении требований к системе уже в процессе реализации получает лишь Разработка, что ведет к лишним работам на стороне Обеспечения Качества по тестированию функционала, который будет изменен или упразднен вовсе. Либо же напротив – Разработка не обладает информацией о процессе тестирования, и своевременное решение о выделении дополнительных сил на исправление неожиданно большого количества дефектов не может быть принято.

Исходя из вышесказанного, можно выделить следующие проблемы:

- Низкий уровень наблюдаемости рабочего процесса в рамках задач конкретного подразделения в разрезе всей системы производства ПО;
- Низкий уровень наблюдаемости жизненного цикла тестовых случаев в разрезе всей системы производства ПО;
- Низкий уровень наблюдаемости жизненного цикла дефектов в разрезе всей системы производства ПО.

Существует необходимость в мониторинге и анализе всей системы производства ПО в целом, с целью управления рисками. Для решения данной задачи может быть использована модель системы производства ПО, состоящая из следующих элементов:

- Задача. Данная сущность необходима для повышения уровня наблюдаемости рабочего процесса в рамках задач конкретного подразделения и обладает такими атрибутами, как дата начала работ, дата завершения работ, количество часов, выделенное на проведение работ, количество часов, потраченное на проведение работ и оставшееся количество часов.
- Граф жизненного цикла дефектов. Вершины графа соответствуют статусам, в которых может находиться дефект, а ребра, связывающие вершины, соответствуют возможности перехода между статусами.
- Дефект. Маркер, соответствующий конкретному дефекту и находящийся в вершине графа жизненного цикла дефектов в зависимости от текущего статуса. Данная сущность необходима для повышения уровня наблюдаемости жизненного цикла дефектов в разрезе всей системы производства ПО. Имеет идентификатор, название, приоритет, место возникновения, способ обнаружения, дату обнаружения, дату закрытия и связи с тестовыми случаями (тестовый случай, в результате которого был создан дефект, и заблокированные дефектом тестовые случаи).
- Граф жизненного цикла тестовых случаев. Вершины графа соответствуют статусам, в которых может находиться тестовый случай, а ребра, связывающие вершины, соответствуют возможности перехода между статусами.
- Тестовый случай. Маркер, соответствующий конкретному тестовому случаю и находящийся в вершине графа жизненного цикла тестовых случаев в зависимости от текущего статуса. Данная сущность необходима для повышения уровня наблюдаемости жизненного цикла тестовых случаев в разрезе всей системы производства ПО. Имеет такие атрибуты, как идентификатор, название, время начала прохождения, время перевода в статус «Успешно».

Цель анализа данной модели – минимизация времени разработки:

$$T = f(p_1, p_2, \dots, p_n), T \rightarrow \min, \quad (1)$$

где T – время, необходимое на проведение работ всеми подразделениями;

$f(p_1, p_2, \dots, p_n)$ – n -арная функция оценки времени T .

$p_1 - p_n$ – параметры, оказывающие влияние на время разработки. К таким параметрам относятся:

- k – количество человек, задействованных в работах в том или ином подразделении.
- t_c – среднее (медианное) время прохождения тестового случая (возможность классификации трудоемкости прохождения тестовых случаев по признаку трудоемкости).
- b_f – среднее (медианное) количество функциональных дефектов на спринт.
- b_r – среднее (медианное) количество регрессионных дефектов на спринт.

- a – отношение количества исправленных за день дефектов к количеству сотрудников, задействованных в их исправлении.
- t_w - среднее (медианное) время взятия дефекта (функционального и регрессионного) в работу тем или иным подразделением.
- t_b - среднее (медианное) время решения дефекта (функционального и регрессионного).
- c - средняя (медианная) цикличность дефекта – количество переводов дефекта в статус «Ретест».
- t_d - среднее (медианное) время, необходимое на реализацию порции функционала в разрезе «Условный Балл Сложности Пользовательской Истории – Человеко-час».
- l - прогресс тестирования – отношение процента пройденных тестовых случаев (в статусе «Успешно»/Всего в наборе) к проценту завершенности спринта (прошедшее время/отведенное время).

Таким образом, двухграфная модель системы производства программного обеспечения позволит сократить общее время, необходимое на проведение работ, за счет повышения уровня наблюдаемости жизненных циклов дефектов и тестовых случаев, что повлечет за собой своевременность эскалации проблем и принятия верных управленческих решений вне зависимости от подразделения. Накопление статистической информации о значениях параметров, оказывающих влияние на время разработки, обеспечит возможность расчета прогнозного времени, необходимого на проведение работ, что позволит внести соответствующие корректировки в план работ на начальных этапах.

1. Рубин К. Основы Scrum: Практическое руководство по гибкой разработке ПО. — М.: «Вильямс», 2016. — С. 544.
2. Котов В. Е. Сети Петри. — М: Наука, 1984. — 160 с.
3. Лельчук А. Л. Актуарный риск-менеджмент. — Москва: Анкил, 2014. — 424 с.