

Система принятия решения для формирования команды разработчиков программного обеспечения с использованием компьютерного моделирования

Ярослав Витязев

В статье будет рассмотрен один из возможных подходов к решению задачи формирования проектной команды разработчиков программного обеспечения (ПО) с использованием компьютерного моделирования.

Как и другие традиционные инженерные дисциплины создание программного обеспечения имеет дело с проблемами стоимости и надёжности. На протяжении нескольких десятилетий стоит задача поиска повторяемого, предсказуемого процесса или методологии, которая бы улучшила продуктивность и качество создаваемого ПО [1, 2, 3, 4]. Без четкого управления разработка ПО выходит из-под контроля, съедая лишнее время и средства. Управление и планирование должно позволять сокращать затраты на разработку и поддержку ПО, позволять завершать ее в срок и с учетом выдвигаемых к ней требований качества.

Существует ряд подходов к процессу создания программного обеспечения, описываемых некоторыми моделями [5]. В рамках статьи мы рассмотрим лишь две из них: модель водопада и итеративный процесс (включающий экстремальное программирование и гибкие методологии). На основе этих подходов в дальнейшем будут построены базовые модели процесса разработки ПО — модели структуры задач проекта.

В каскадной модели каждая из процессных областей представляет собой отдельную фазу проекта [5]. Фазы выполняются строго последовательно, т. е. анализ и дизайн начинаются после написания требований, началу реализации предшествует завершение дизайна и т. д. Этот метод хорошо работает в проектах, где требования могут быть четко определены и зафиксированы. В таких проектах каскадная модель позволяет обеспечить заданный уровень качества (который может быть весьма высоким) и соблюдать бюджетные и временные ограничения. Благодаря этому она часто используется в больших организациях при строгих требованиях к надежности создаваемого ПО.

Процесс итеративной (или инкрементальной) разработки стал эволюционным развитием модели водопада [5, 8]. Процесс состоит из серии повторяющихся итераций (их число зависит от конкретного проекта), каждая из которых фактически является полноценным мини-проектом с фазами определения требований, анализа, дизайна и т. д. В результате очередной итерации продукт приобретает новую функциональность или улучшения в существующей функциональности. Полный набор требований, зафиксированный границами проекта, оказывается реализованным после завершения финальной итерации. Итеративная модель используется во многих процессах разработки, включая RUP и гибкие методологии [3, 5].

Одной из важных задач при разработке программного обеспечения является задача подбора проектной команды. Определение проектной команды лишь одна из задач проектного управления, однако, от подобранной команды будет зависеть успешность всего проекта. Суть задачи состоит в эффективном выборе перечня специалистов при существующих проектных ограничениях и заданной цели проекта. Задача подбора проектной команды относится к стадии планирования и обычно следует после определения структуры работ проекта.

Но как определить, какая из альтернативных проектных команд будет наилучшей в экономическом плане? Для решения этого вопроса разумно применить компьютерное моделирование, построив специальную моделирующую систему для решения данной задачи.

Суть моделирующей системы состоит в проигрывании модели на различных исходных данных: они могут быть определены заранее как массив параметров конкретных специалистов или при помощи определенного метода, например, при помощи параметров, случайно заданных в рамках определенных разумных границ. В этом случае можно будет получить характеристики специалистов, при которых цель проекта в соответствии с определенным показателем достигается наиболее эффективно. По мере многократного проигрывания итераций модели собирается устойчивая статистика модельных показателей. По мере окончания процесса эти показатели сравниваются между собой. На основе результатов сравнения осуществляется выбор того состава проектной команды, который будет наиболее эффективен.

Для реализации такой моделирующей системы необходимы следующие модели: общая модель проекта, субмодели задачи, структуры (или последовательности) задач, ресурсов, а также субмодель специалиста как частный случай субмодели ресурса. Для каждой из этих субмоделей должен быть определен ряд показателей — количественно измеримых переменных, которые могут быть применены для последующего анализа эффективности проектной команды.

Модель задачи (МРТ) в рамках моделирующей системы должна отражать параметры, необходимые для вычисления показателей эффективности. При моделировании разумно использовать схему «план-модель-факт», при помощи которой впоследствии можно будет сравнить плановые, модельные и фактические показатели стоимости и сроков выполнения задачи. Каждая задача уникальна, поэтому могут потребоваться дополнительные параметры и показатели эффективности, кроме общих для всех задач, таких как: вероятность завершения задачи в срок, вероятность завершения задачи в рамках бюджета и т. п.

Как правило, проект по разработке ПО состоит из нескольких задач, которые далеко не всегда выполняются по принципу обыкновенной очереди. В этом случае в моделирующей системе должна быть предусмотрена **модель для построения структуры задач (МРТС)**. Такая модель может иметь множество воплощений, например, реализовывать стандартные структуры процессного подхода к управлению задачами в виде WBS-связей; она может использовать сети Петри или же основываться на цикле Деминга PDCA [5]. Для последовательности задач также формируются показатели эффективности, при помощи которых можно при анализе определить вероятности осуществления перехода от одной задачи к другой, задержки между задачами (которые могут возникнуть, например, при определенных параметрах исполнителей), стоимость и продолжительность выполнения всех задач проекта.

Как правило, с проектом и каждой его задачей связаны различные ресурсы. Это следует отразить в **модели ресурса (МРР)**. В некоторых моделях исполнителей (специалистов) также рассматривают как особый вид ресурса. Ресурсы могут тратиться (например, деньги, сроки), быть занятыми (рабочее время специалиста). Такие свойства ресурсов как исчерпаемость и занятость должны быть отражены и обязательно учитываться в модели. Структура параметров варьируется для каждого типа ресурса и определяется требуемыми показателями эффективности, которые будут использованы при последующем анализе. Среди таких показателей можно выделить загруженность ресурса, состояние занятости ресурса, объем ресурса и т. д.

При создании **субмодели специалиста (МПРС)** по разработке ПО необходимо выделить ряд обобщенных параметров, характерных для каждого специалиста и важных при альтернативном выборе. Такими параметрами могут быть как простые: оценка стоимости часа работы, возможность поддержания оперативной связи, оценка сроков выполнения работы, возможность выполнения работы, занятость, так и факторы со сложной структурой:

- профессионализм (позволяет определить профессиональный уровень специалиста);
- ответственность (позволяет определить уровень ответственности);
- адекватность (поведение специалиста в нетривиальных проектных ситуациях).

В различных проектах и при различных способах анализа будут полезны и такие параметры специалистов, как возраст, географическое положение (часовой пояс), семейное положение и прочие. Каждый проект уникален и для некоторых проектов могут потребоваться дополнительные факторы, существенно влияющие на модель специалиста, которые должны быть учтены в модели. Возможны дополнительные — кроме общих для всех — показатели эффективности и параметры для отдельных типов специалистов (вероятно для программистов и копирайтеров следует использовать различную структуру показателей).

На рисунке 1 представлена общая схема работы системы. На этапе формирования проектной команды уже должна существовать структура работ, которая может быть определена как модель MPTS в виде определенной последовательности задач (MPT). Аналогично задаются и ресурсы проекта — в виде модели MPR. Формируются исходные данные в виде массива параметров специалистов, которые могут быть использованы как члены проектной команды. На каждой итерации происходит расчет показателей эффективности для каждой из используемых моделей, при этом на каждой итерации подставляются новые элементы из массива параметров специалистов. Показатели эффективности (E) накапливаются и записываются в специальное хранилище. Таким образом, на выходе мы получаем набор показателей эффективности по каждой из описанных выше моделей для всех возможных составов проектной команды. Поскольку для каждого состава проектной команды рассчитанные показатели совпадают по структуре, мы можем их сравнить различными способами между собой и сделать альтернативный выбор команды разработчиков ПО. Эту задачу должна решать подсистема анализа и сравнения, описание которой выходит за рамки данной статьи.

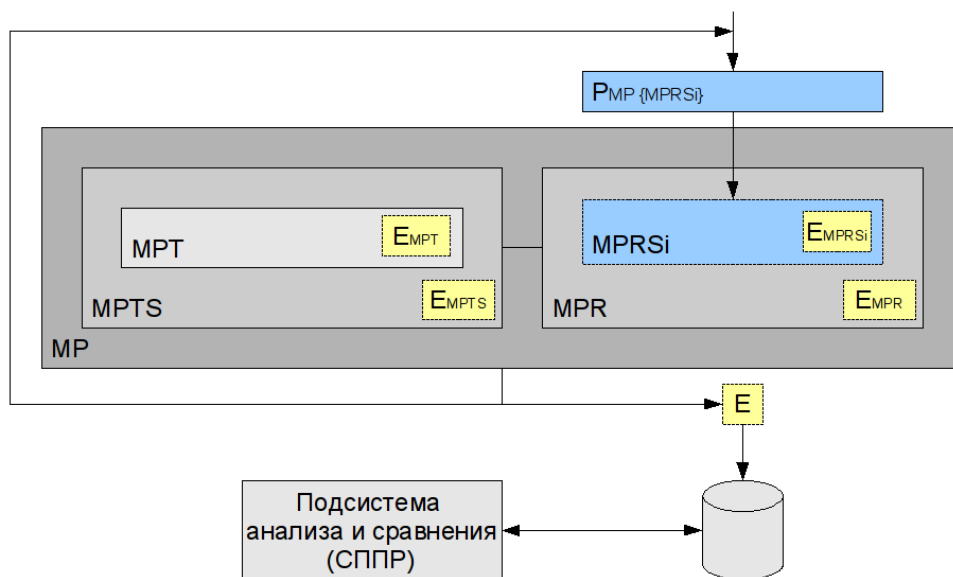


Рисунок 1. Общая схема работы системы

Ввиду того, что каждый проект по разработке ПО уникален, построить одну единственную универсальную модель для всех проектов не представляется возможным.

Существует несколько способов реализации моделирующей системы. Реализация каждой моделирующей системы как уникальной на определенном языке программирования — длительный и затратный процесс, однако, при этом подходе можно использовать готовые разработки в виде программных библиотек. В некоторых случаях разумно применить уже существующие специализированные программные системы для моделирования, однако, не в каждой такой системе есть все требуемые модели или возможность их реализации.

В таком ключе имеет смысл разработка и использование так называемого фреймворка (от англ. framework): набора стандартных базовых для каждой модели классов с возможностью расширения их функциональных возможностей [6].

Термин фреймворк используется в программировании, обозначая «простую концептуальную структуру, используемую для решения сложной, проблемной задачи». Значение этого термина зависит от контекста его использования.

Framework отличается от библиотеки (library) тем, что выполняет код, написанный для него, а не исполняется сам. Также в отличие от библиотеки, которая объединяет в себе набор близкой функциональности, фреймворк содержит в себе большое число разных по сфере применения библиотек, которые могут использоваться для более простого решения каждой конкретной задачи [6, 7].

Использование фреймворка для решения задачи моделирования является удобным методом и позволяет использовать единую функциональную основу, облегчающую построение каждой модели. В сам фреймворк необходимо заложить основные модели (MPT, MPTS, MPR, MPRS), которые на практике можно применить для решения задачи подбора проектной команды.

Список источников

1. И. Соммервилл. Инженерия программного обеспечения = Software Engineering. — 6-е изд. — М.: «Вильямс», 2002. — С. 642. — ISBN 5-8459-0330-0
2. Управление проектами: Пер. с англ. / Паула Мартин, Карен Тейт. - К.: [КПЯ «СИСТЕМИ»], 2005. - 192 с.: ил.
3. Разработка программного обеспечения [Электронный ресурс] : Википедия, свободная энциклопедия. Режим доступа http://ru.wikipedia.org/wiki/Разработка_программного_обеспечения , свободный.
4. Brooks, Fred (1995). The Mythical Man-Month, 20th Anniversary Edition, Adison Wesley. ISBN 0-201-83595-9.
5. Современные процессы разработки программного обеспечения [Электронный ресурс] : Библиотека RSDN. Режим доступа <http://rsdn.ru/article/Methodologies/SoftwareDevelopmentProcesses.xml>, свободный.
6. Framework [Электронный ресурс] : Википедия, свободная энциклопедия. Режим доступа <http://ru.wikipedia.org/wiki/Framework> , свободный.
7. Джек Гринфилд, Кит Шорт, Стив Кук, Стюарт Кент, Джон Крупи. Фабрики разработки программ (Software Factories): потоковая сборка типовых приложений, моделирование, структуры и инструменты = Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. — М.: «Диалектика», 2006. — С. 592. — ISBN 978-5-8459-1181-0
8. Managing the Development of Large Software Systems [Электронный ресурс] : Royce, Winston (1970). Режим доступа <http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf> , свободный.