Имитационное моделирование в управлении сложными проектами

Об авторах

Каталевский Дмитрий Юрьевич
Докторант, доцент, к. э. н.
Высшая школа бизнеса
МГУ имени М.В. Ломоносова
Москва, РФ
email: dkatalevsky@yahoo.com
ORCID: 0000-0002-3920-5041

Суслов Сергей Алексеевич
Директор по международным продажам и маркетингу
Компания AnyLogic
Санкт-Петербург, РФ
email: serge.suslov@gmail.com
ORCID: 0000-0002-0587-161X

Аннотация

Актуальность. В данной статье проводится анализ ведущих зарубежных работ в области имитационного моделирования проектного управления на основе качественного инструментария системной динамики (причинно-следственных диаграмм обратной связи). Имитационное моделирование имеет длительную историю изучения проектного управления. Сторонники применения имитационных моделей в управлении проектами отмечают важную роль понимания сложной структуры обратных связей в восприятии руководителями динамики проектного менеджмента. Зачастую нелинейность обратных связей, эффекты запаздывания, тесная взаимосвязанность и неочевидное взаимовлияние элементов проектной системы друг на друга в совокупности приводят к контринтуитивному поведению проектов, срывая их сроки и стоимость. Сложные проекты состоят из тесно взаимосвязанных подсистем, сбой хотя бы в одной из них вызывает каскадные эффекты, быстро распространяющиеся по всей системе. Это одна из ключевых причин выхода сложных проектов из-под контроля.

Цель исследования. Выявить механизм возникновения сбоев при управлении сложными проектами (выхода за пределы сметы, срыва сроков проектов).

Задачи исследования. Рассмотреть управление проектами через подходы, свойственные управлению сложными системами (системная динамика), проанализировать зарубежный опыт исследований в данной области; выявить ключевые обратные связи, возникающие в ходе управления проектами, рассмотреть механизм их воздействия на динамику проекта; по результатам анализа дать рекомендации проектным менеджерам.

Результаты исследования. Представлен обобщающий механизм возникновения управленческих сбоев в проектном управлении на примере разработки программного обеспечения.

Ключевые слова

проектное управление, системная динамика, имитационное моделирование, управление сложностью

Simulation modeling in complex project management

About the authors

Dmitry Yu. Katalevsky

Doctoral Student, Associate Professor, PhD, MPA
Graduate School of Business
Lomonosov Moscow State University
Moscow, RF
email: dkatalevsky@yahoo.com

Sergey A. Suslov

Director of International Sales and Marketing
«AnyLogic» Company
Saint Petersburg, RF
email: serge.suslov@gmail.com

Abstract

Relevance. This article analyzes the leading foreign works in the field of simulation modeling of project management based on qualitative tools of system dynamics (causal feedback diagrams). Simulation modeling has a long history of studying project management. Proponents of the use of simulation models in project management note the important role of understanding the complex structure of feedback in managers' perception of the dynamics of project management. Often, the non-linearity of feedbacks, lag effects, close interconnectedness and the non-obvious mutual influence of the elements of the project system on each other together lead to counterintuitive behavior of projects, disrupting their timing and cost. Complex projects consist of closely interconnected subsystems, a failure in at least one of them causes cascading effects that quickly spread throughout the system. This is one of the key reasons why complex projects get out of control.

The purpose of the study. To identify the mechanism of failures in the management of complex projects (going beyond estimates, project deadlines).

Research objectives. To consider project management through approaches peculiar to the management of complex systems (system dynamics), to analyze foreign research experience in this field; to identify key feedback arising during project management, to consider the mechanism of their impact on the dynamics of the project; to make recommendations to project managers based on the results of the analysis.

The results of the study. A generalizing mechanism of the occurrence of managerial failures in project management is presented on the example of software development.

Keywords

project management, system dynamics, simulation modeling, complexity management

Scientific specialty of the publication: 08.00.05 – Economics and management of the national economy

ВВЕДЕНИЕ

Изучение возможности использования имитационного моделирования для работы с комплексными проектами имеет многолетнюю историю. Так, первые исследования в области имитационного моделирования проводились в 1960-х гг., однако лишь в 1980-х они начали применяться в работе с конкретными проектами.

С 1990-х годов начали выходить многочисленные научные публикации, описывающие использование имитационного подхода при управлении проектами. Многие исследования по управлению проектами и имитационным моделям были посвящены поиску основных причин срыва проектов и значительных перерасходов средств.

В первых систематических исследованиях Робертс [15] применил имитационные модели (в частности, системно-динамические) для управления проектами, введя понятие потоков работ по проекту (единиц деятельности) и концепцию разрыва между восприятием и реальностью (разницы между предполагаемым и реальным прогрессом).

В своих исследованиях он указывал на то, что объем работ и усилий, необходимых для успешного завершения проекта, часто недооценивается и у ответственных лиц складывается искаженный взгляд на проект, что, в свою очередь, неминуемо ведет к нерациональному использованию ресурсов [16].

Позднее Ричардсон и Пью разработали концепцию выявленных и невыявленных ошибок, предполагаемого прогресса и реальной продуктивности (в данный момент эти труды считаются классикой в области системно-динамических моделей для управления проектами).

В модели, представленной Ричардсоном и Пью, особое внимание уделялось изучению нескольких ключевых понятий, свойственных любому комплексному проекту: цикл мониторинга и контроля, исправление ошибок и наем персонала [17].

Опираясь на достигнутые успехи в области имитационного моделирования проектов, Пью и Ричардсон создали «Систему моделирования для управления проектами» (набор сложных системно-динамических моделей, предназначенных для использования в качестве инструмента поддержки принятия управленческих решений). Данный инструмент успешно применялся в ряде проектов по управленческому консалтингу в сфере крупного строительства и даже урегулирования споров (случаи возникновения задержек и сбоев в осуществлении проекта).

Широкую известность получили также исследования проектного управления с помощью имитационного моделирования в таких областях, как управление разработкой программного обеспечения [3], кумулятивный эффект от влияния задержек и внесения проектных изменений на стоимость проекта [20], стратегическое управление комплексными проектами [11], стратегии кризисного управления проектами [19] и многие другие.

Проведенные исследования позволили сформировать обобщенную *теорию механизма возникновения сбоев при осуществлении проектов*.

Важно понимать разницу между традиционными методами управления

Рис. 1 ***Визуальное отображение накопителей и потока в модели***

Источник *Составлено авторами*

проектами и имитационным моделированием. Традиционные подходы основаны на методе критического пути и представляют проект в виде связанной последовательности отдельных технических задач и событий. При этом подходе проект является суммой отдельных составляющих работы.

Такие инструменты и системы управления могут ввести в большое заблуждение, потому что не показывают, что в реальности *осуществление проекта – это не линейное, последовательное выполнение ряда задач, а циклический процесс*, где каждая задача проходит несколько итераций. Более того, данные инструменты способствуют восприятию проектов в виде «неуправляемых реактивных снарядов, стремящихся к результату, на который мало влияет вмешательство человека» [6].

Среди практиков имитационного моделирования, в частности метода системной динамики, широко распространено мнение, что ключевой причиной провалов проектов является слишком сложная система обратной связи, затрудняющая понимание проектов ответственными лицами и принятие решений. Принятые решения и меры обычно приводят в действие запутанную систему множества причинно-следственных связей. Это, как правило, влечет за собой задержки, мешающие осуществлению проекта.

Ниже описан типичный механизм срыва проектов на примере компании, занимающейся разработкой программного обеспечения.

УПРАВЛЕНИЕ СЛОЖНЫМИ ПРОЕКТАМИ: ТИПОВОЙ МЕХАНИЗМ СБОЕВ

Представим, что некая компания ABC хочет осуществить комплексный проект по разработке программного обеспечения. Допустим, для того чтобы успешно выпустить ПО, сотрудникам компании ABC необходимо написать примерно 100 тыс. строк исходного кода[1].

Используя устоявшийся язык системно-динамического моделирования – диаграмму потоков и накопителей, мы можем представить структуру этого проекта следующим образом (рис. 1):

[1] Продуктивность выполнения проекта разработки программного обеспечения может измеряться разными способами: строками исходного кода, написанными за определенный промежуток времени; количеством ошибок на 1000 строк кода; рабочими днями (потраченным программистом временем на написание кода без учета планирования и прочей сопутствующей основной работе деятельности); объемом кода, который программист может написать за год. Иногда в качестве меры продуктивности используют длительность цикла (от принятия в производство до выполнения) и срок выполнения (от получения заказа от клиента до передачи ему продукта). Здесь для удобства продуктивность измеряется в строках кода, написанных за определенное время (хотя многие специалисты в этой области считают такой способ измерения продуктивности разработки ПО неэффективным).

Накопитель[2] ***Запланированная работа*** отражает изначальный объем работы, который, по мнению руководства компании ABC, необходимо выполнить, чтобы выпустить программный продукт (например, 100 тыс. строк кода). Накопитель ***Выполненная работа*** отражает ту часть работы, которая закончена и не нуждается в дальнейшей доработке.

В обоих накопителях в качестве единицы измерения используется число строк программного кода. По мере разработки программного обеспечения ***Запланированной работы*** становится меньше, а ***Выполненной работы*** – больше. Поток[3] ***Выполняемая работа*** отражает ход выполнения работ в конкретный момент времени. Значение потока может исчисляться в строках кода, добавленных за определенный период времени, например за день.

Обычно на скорость ***Выполняемой работы*** влияют непосредственно ***Сотрудники*** и ***Продуктивность*** (рис. 2). Можно было бы предположить, что чем больше технического персонала (разработчиков программного обеспечения) будет привлечено к работе над ПО и чем продуктивнее они добавляют строки к коду в заданный период времени, тем быстрее будет расти значение накопителя

Выполненная работа. Однако разумно предположить, что не вся ***Выполняемая работа*** полезна и отвечает нашим требованиям к качеству. В систему необходимо включить новую переменную – ***Качество***, которая оценивает единицы ***Выполняемой работы*** (присваивает значение 1 или 0), прежде чем они будут добавлены в накопитель ***Выполненной работы*** и, соответственно, не будут нуждаться в дальнейшей доработке.

Для простоты примем, что ***Качество*** связано с количеством ошибок, допущенных в программном коде. На самом же деле на ***Качество*** и ***Продуктивность*** может прямо либо опосредованно влиять множество факторов – как внутренних, так и внешних.

Как отмечалось прежде, большим скачком в понимании динамики управления проектами было добавление ***Цикла исправления ошибок*** (рис. 3). Ошибки выявляются в процессе тестирования. По мере выявления ошибок (в зависимости от скорости их обнаружения) соответствующая часть кода отправляется на доработку и увеличивает объем работы, которую необходимо выполнить.

Обычно при создании системно-динамических моделей используется накопитель ***Выявленные ошибки***, обозначающий объем работы, которую нужно переделать. Также существует понятие ***Невыявленные ошибки***, обозначающее работу, содержащую существующие, но еще не выявленные ошибки.

Значение накопителя ***Выявленные ошибки*** увеличивается в ходе входящего потока ***Выявление ошибок*** (в данном случае единица измерения – количество строк, классифицированных как нуждающиеся в исправлении, за единицу

---

[2] ***Накопитель*** – термин в системной динамике, обозначающий элемент системы, который со временем наполняется или истощается. Накопители отображают состояние системы и могут быть изменены лишь потоками.

[3] ***Поток*** – термин в системной динамике, обозначающий течение чего-либо из накопителя или в накопитель. Потоки отражают процессы – изменения в накопителях в любой заданный момент времени.

времени) и уменьшается за счет исходящего потока ***Исправление ошибок*** (количество исправленных строк кода за единицу времени).

Выявление ошибок может вызывать определенные сложности, ведь они далеко не всегда быстро обнаруживаются. Однако даже накопитель ***Выявленные ошибки*** может содержать ошибки, которые потребуют повторного прохождения цикла исправления ошибок.

Так, исследование *NASA*, посвященное проблемам разработки средств программного обеспечения полета [13, с. 47], показало, что в процессе разработки программного обеспечения много ошибок допускается на стадиях создания архитектуры программы и написания кода.

Автор одного из лучших практических руководств по разработке программного обеспечения С. Макконнелл отмечает, что «при работе над крупными проектами исправление ошибки в требованиях, обнаруженной на этапе проектирования архитектуры, обычно обходится втрое

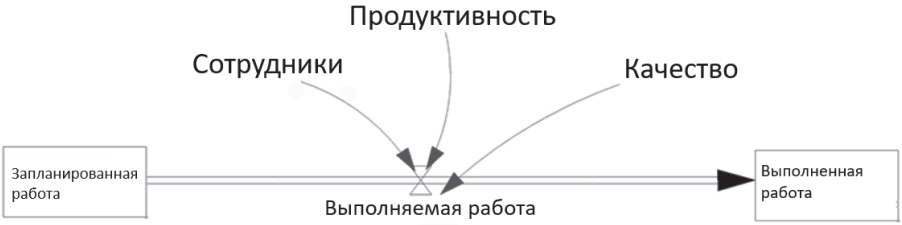

Рис. 2 ***Ключевые переменные, влияющие на выполнение проекта***

Источник *Составлено авторами*

Рис. 3 ***Типичный цикл исправления ошибок***

Источник *Составлено авторами*

дороже, чем исправление аналогичной ошибки, найденной на этапе выработки требований… Такая же ошибка, обнаруженная при кодировании, обходится дороже уже в 5–10 раз, при тестировании системы – в 10, а после выпуска программы – в 10–100 раз. В менее крупных проектах с меньшими административными расходами последний показатель ближе к 5–10, чем к 100…» [2, с. 37].

Таким образом, важно отслеживать не только качество работы, но и *показатель выявления ошибок*. Так, Линейс и Форд называли цикл исправления ошибок характеристикой, оказывающей *наибольшее влияние* на успешность проекта: «По нашему мнению, цикл исправления ошибок является наиболее важной характеристикой системно-динамических моделей проектов. Повторяющийся характер цикла исправления ошибок, при котором исправления влекут за собой новые ошибки, снова требующие исправления и так далее, создает критические режимы работы, которые часто затягиваются почти на весь срок выполнения проекта и становятся причиной многих проблем» [12, с. 160].

Подсчитано, что цикл исправления ошибок легко может занять *до половины времени* осуществления проекта [6].

Остаточные ошибки, допущенные при разработке программного обеспечения, и небольшие отдельные огрехи часто незначительны сами по себе, но все вместе могут оказаться губительными для комплексного проекта.

Эксперт *NASA* по разработке комплексного программного обеспечения Дж. Хольцманн пришел к выводу, что добавление резервных копий и защиты от неисправностей оборачивается увеличением размера и сложности системы с «незапланированными связями между прежде не связанными компонентами системы… В результате количество возможных ошибочных комбинаций настолько велико, что просто невозможно проверить их все в процессе систематического тестирования ПО. К примеру, всего сотня остаточных дефектов может встретиться примерно в 10 тысячах различных комбинаций» [9]. Именно сочетание незначительных дефектов привело к неудачному окончанию миссии *NASA* Mars Global Surveyor [8].

Обычно проектные менеджеры стараются контролировать четыре ключевых фактора при реализации проектов:

- необходимый объем ресурсов;
- продуктивность;
- качество выполнения работы;
- время выявления ошибок.

Зачастую лица, управляющие проектом, уделяют наибольшее внимание именно ресурсам и продуктивности. В рассматриваемом примере с компанией ABC это сотрудники-программисты («ресурс») и их навыки программирования (ошибочно принимаемые за «продуктивность»), которые считаются залогом успеха или срыва проекта. Однако этого недостаточно – искомый набор факторов представляется *неполным* для понимания истинных причин срывов проекта.

Для дальнейшего анализа работы компании ABC над проектом по разработке программного обеспечения необходимо добавить причинно-следственные цепочки и петли обратной связи, отражающие динамику жизненного цикла проекта.

Руководство компании ABC контролирует реализацию проекта с помощью периодической оценки прогресса

в работе. Прежде всего оно попытается оценить необходимость привлечения дополнительного персонала, чтобы закончить оставшуюся работу (включая исправление выявленных ошибок) в срок. Так, ***Предполагаемый прогресс*** выражается в расчете ***Ожидаемого срока завершения работ***. Если ***Ожидаемый срок завершения*** значительно отличается от ***Назначенного срока завершения***, то руководству потребуется принять меры.

Наиболее распространенный, общепринятый подход – *привлечь больше сотрудников*, чтобы увеличить объем ***Выполняемой работы*** (этот цикл отражен на рис. 4: *Невыявленные ошибки → Предполагаемый прогресс → Ожидаемый срок завершения → Назначенный срок завершения → Оставшееся время → Необходимость в дополнительном персонале → Привлечение персонала → Наем → Работа привлеченного персонала над проектом*).

Если этот вариант по каким-либо причинам невозможен (например, ограничены ресурсы или недоступны кандидаты с подходящей квалификацией и навыками), то руководство введет ***Продленный рабочий день*** для группы разработки программного обеспечения (переработки). Это самый распространенный способ избежать дополнительных затрат и обязательств, связанных с наймом новых сотрудников.

Переработки быстро становятся для компании «обычной» новой практикой. Но разрыв между ***Выполненной работой*** и ***Запланированной работой*** не исчезает. Постоянные переработки

Рис. 4 ***Цикл «Нам нужно больше людей!»***

Источник *Составлено авторами*

через некоторое время снижают продуктивность и повышают количество совершаемых ошибок (эффект выгорания).

Увеличение числа ошибок еще сильнее растягивает цикл исправления ошибок, что, в свою очередь, увеличивает объем работы и откладывает срок ее завершения. Весь цикл повторяется снова. Петля обратной связи от ***Переработок*** становится *самовоспроизводящейся*[4] (рис. 5a). Обычно руководство серьезно недооценивает цикл ***Переработок*** и его негативное влияние на работоспособность и моральное состояние персонала [5, 14].

Через несколько недель несколько разработчиков программного обеспечения (фактически ключевые сотрудники) решают покинуть компанию ABC из-за постоянной ***Работы в напряженном режиме*** и подрыва ***Морального духа*** (рис. 5b).

Часто наиболее ценные специалисты уходят первыми, поскольку с их навыками и квалификацией проще найти работу с лучшими условиями. Помимо того что проект сталкивается с утечкой мозгов, ***Запланированная работа*** уволившихся сотрудников перераспределяется между оставшимися сотрудниками, что дополнительно ухудшает их производственные показатели и усиливает самовоспроизводящуюся петлю обратной связи ***Переработок***.

Теперь руководство намерено нанять новый персонал или привлечь к проекту своих сотрудников, ранее в нем не участвовавших. Однако на поиск и прием на работу подходящих специалистов требуется время (это отложенный процесс). Поэтому перегруженный оставшийся персонал продолжает падать духом, а ***Качество*** работы – снижаться.

Однако привлечение нового персонала имеет еще один парадоксальный эффект[5], который обычно недооценивают или игнорируют. Когда к работе в группе по разработке программного обеспечения привлекают новых сотрудников, проект пополняется менее опытными специалистами, чем те, кто уже был задействован в нем. Это в наибольшей степени касается случаев, когда нужны узкопрофильные специалисты или те, на которых спрос на рынке труда очень высок.

Чем сильнее ограничен рынок труда, тем ниже уровень компетенций привлекаемых к проекту сотрудников. Поэтому новичкам требуется время, чтобы ознакомиться с проектом и пройти обучение, которое уже прошел остальной персонал.

Привлечение нового персонала вызывает возникновение ряда взаимосвязанных петель обратной связи. Новички могут преднамеренно или непреднамеренно способствовать дестабилизации коллектива за счет того, что: иногда они менее преданны компании; могут не прижиться на новой работе; менее квалифицированны, чем предполагалось

[4] ***Самовоспроизводящаяся петля обратной связи (reinforcing loop)*** – петля обратной связи, в которой суммарный эффект причинных связей со временем усиливает (подкрепляет) движение переменных величин в едином направлении за счет позитивной обратной связи (изменения в одном направлении ведут к еще большим изменениям в том же направлении).

[5] ***Контринтуитивное (парадоксальное) поведение*** сложных систем – термин, введенный американским ученым Дж. Форрестером, родоначальником системной динамики. Такое поведение представляет собой удивительный результат стратегий, направленных на решение проблемы. Часто предполагаемое «решение» контрпродуктивно и приносит результат, обратный желаемому, поскольку проблем становится больше, прикладываются еще большие усилия, что фактически усугубляет ситуацию.

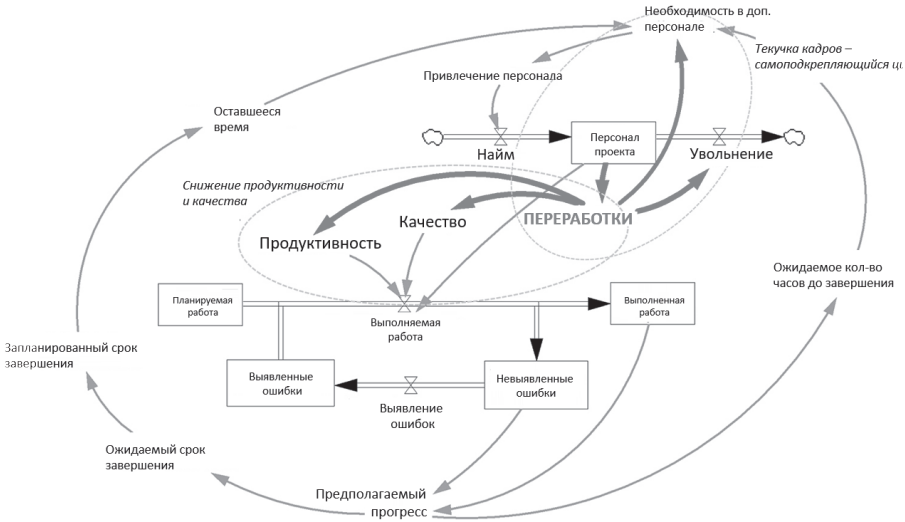

Рис. 5a ***Цикл переработок***

Источник *Составлено авторами*

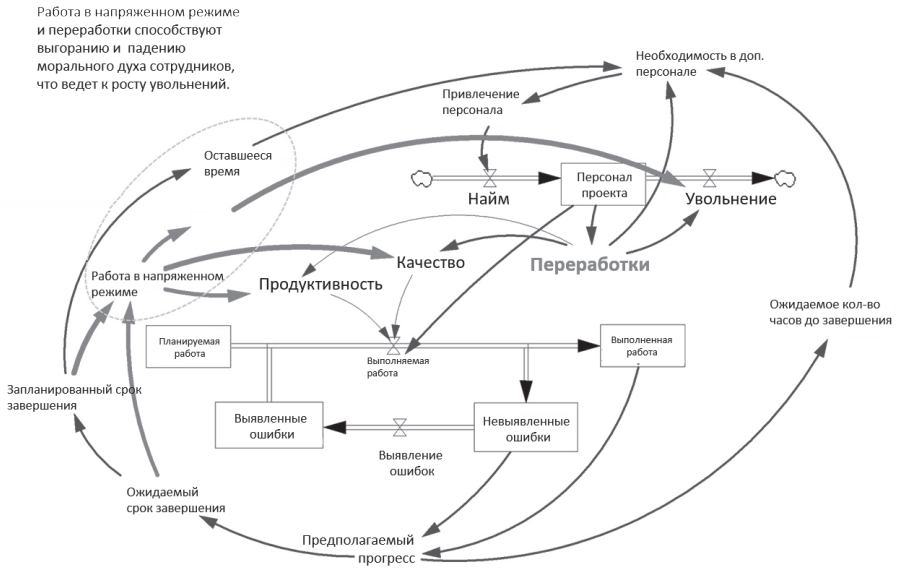

Рис. 5b ***Работа в напряженном режиме и подрыв морального духа способствуют росту текучести кадров***

Источник *Составлено авторами*

(переоценены); обладают нереалистичными представлениями о проекте. Это еще один порочный (самовоспроизводящийся) цикл: *Наем (новичков) → Увеличение нагрузки на текущий персонал → Выгорание / демотивация текущего персонала → Увеличение числа увольнений сотрудников → Новый наем (новичков)*.

Новым сотрудникам нужно четко ставить задачи, их работа требует большего курирования и контроля. Поэтому квалифицированный персонал теперь уделяет все больше внимания не разработке программного обеспечения, а работе с новичками: обучению, помощи и контролю их работы.

Руководство компании ABC понимает, что в процессе разработки программного обеспечения теперь возникает еще больше ошибок, а цикл ***Исправления ошибок*** растягивается. Неожиданным результатом значительного расширения штата становится *снижение уровня профессионализма* команды, что негативно влияет на продуктивность и качество. Снижаются объем ***Выполняемой работы***, ***Качество*** и ***Продуктивность***, а число ошибок и объем работы по ***Выявлению ошибок*** и ***Исправлению ошибок*** возрастают. Разрыв между ***Выполненной работой*** и ***Предполагаемым прогрессом*** увеличивается, но на практике это обычно приводит к решению *нанять еще больше сотрудников*.

Команда менеджеров не может справиться с увеличением стоимости проекта, а его разрастающаяся структурная организация – нормально функционировать. Это влечет за собой проведение многочисленных собраний и обсуждений, но причины сложившегося положения остаются *неявными*.

Еще один парадокс заключается в том, что, несмотря на увеличение состава проектной группы, доля полезного участия каждого из ее сотрудников может уменьшиться, поскольку теперь коллектив разделен на подгруппы и гораздо больше времени уходит на переговоры между ними и согласование вносимых изменений. В результате члены коллектива все больше морально и физически устают, что приводит к эмоциональному выгоранию, снижению морального духа и другим, вытекающим из этого обстоятельства негативным последствиям.

Наконец, как отмечает С. Макконнелл, помимо неэффективного управления, непредсказуемо сложным может стать также и сам программный продукт, в создании которого участвуют множество сменяющих друг друга членов команды: «Программные проекты редко терпят крах по техническим причинам. Чаще всего провал объясняется неадекватной выработкой требований, неудачным планированием или неэффективным управлением. Если же провал обусловлен все-таки преимущественно технической причиной, очень часто ей оказывается неконтролируемая сложность. Иначе говоря, приложение стало таким сложным, что разработчики перестали понимать, что же оно делает. <…> Управление сложностью – самый важный технический аспект разработки ПО» [2, с. 75].

Но ситуация может продолжать ухудшаться. Проблемы, возникшие на ранних стадиях проекта, быстро увеличиваются в количестве по мере его развития: требования к работе плохо формулируются (непонятные, неполные, слишком обобщенные; заказчик

пытается их менять уже в ходе реализации проекта), а тестирование и оценка ПО проводятся недостаточно строго и полно. Так, ряд исследований показывает, что при реализации типичного проекта по разработке программного обеспечения требования во время разработки меняются примерно на 25% [4], что увеличивает на 75–80% объем дополнительной работы [10].

Все это может вызвать необходимость последующего исправления значительного числа ошибок. Классические примеры такого развития событий наблюдаются в практике управления крупными проектами по строительству, производству, разработке новых продуктов и прочими проектами, в которых внесение изменений на этапах проектирования и конструирования влияет на стадии строительства и производства.

Иногда ошибки могут обнаружиться через месяцы и годы, после того как они возникли. Это распространенная ситуация для крупных комплексных проектов (строительных, инфраструктурных, связанных с разработкой программного обеспечения). Некоторые выявленные ошибки могут быть губительными для реализации проекта и повлечь необходимость внесения изменений, которые отбрасывают его далеко назад.

ЗАКЛЮЧЕНИЕ

Анализ комплексных проектов показал, что основными источниками рисков для реализации проектов в соответствии с установленными сроками и бюджетом являются:

- задержки в обмене информацией или внесение изменений в первичную конструкцию;
- ресурсные ограничения (например, заторможенный запуск проекта, неполный профессиональный состав команды, вынужденная экономия средств в связи с финансовыми ограничениями);
- новые процессы, ресурсы, сотрудники;
- организационные и управленческие изменения;
- изначально грубые допущения относительно проекта (сжатые сроки, недостаточный бюджет, неправильное восприятие общей сложности проекта).

И другие факторы.

Поскольку проекты (особенно комплексные) обычно представляют собой системы с тесно взаимосвязанными элементами, для них крайне важно избегать резонансного эффекта. Из-за множества взаимосвязей нелинейных петель обратной связи с нежелательными и непредсказуемыми последствиями, а также их комбинированного характера комплексные проекты часто выходят из-под контроля.

Один из ключевых выводов состоит в том, что комплексные проекты уязвимы перед ошибками – как техническими (которые имеют свойство накапливаться), так и управленческими. Людям, как правило, очень сложно работать с системами, компоненты которых тесно взаимосвязаны, в то время как цена ошибки подчас слишком высока (в том числе может измеряться человеческими жизнями).

Способность прогнозировать разные сценарии дальнейшего развития проекта и оценивать возможные результаты управленческих решений становится крайне важным навыком. Поэтому

имитационное моделирование начинает играть все более важную роль в управлении проектом.

Причинно-следственные диаграммы представляют собой удобный инструмент для проведения подобных системных расчетов и сравнения различных сценариев. Следующий шаг – создание имитационной модели. Это полезный инструмент управления проектами, позволяющий выявить и смоделировать ключевые риски в управлении проектом, тем самым минимизируя негативное воздействие на него.

При этом в исследовательской среде накоплен уникальный опыт имитационного моделирования динамики поведения сложных проектов, позволяющий учесть множество субъективных факторов, которыми часто пренебрегают в традиционных подходах (например, моральный дух коллектива, продуктивность, мотивация, усталость, ухудшение качества).

С годами системно-динамическое моделирование обрело свой язык и методологию для выражения различных материальных и нематериальных переменных (факторов), которые могут быть измерены (можно провести их численные подсчеты) и добавлены в модель.

Инструментарий имитационного моделирования (причинно-следственные диаграммы обратной связи) позволяет рассмотреть проект системно, концептуализировав его динамику. Общая сложность даже не очень больших проектов, как правило, быстро превосходит человеческие возможности по рациональной оценке системы в целом.

Имитационные модели позволяют выявить, казалось бы, парадоксальные факты, отражающие неочевидное (контринтуитивное, как отмечал Дж. Форрестер) поведение системы, и сделать из них правильные выводы.

Список источников

1. Каталевский Д.Ю. Основы имитационного моделирования и системного анализа в управлении. М. : Дело. 2015.
2. Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ. М. : Русская редакция. 2010.
3. Abdel-Hamid T.K. A Multiproject Perspective of Single-Project Dynamics // Journal of Systems and Software. Vol. 22. 1993, no. 3. Pp. 151–165.
4. Boehm B.W. Software Engineering Economics. Englewood Cliffs. NJ : Prentice-Hall. 1981.
5. Chan M. Fatigue: The Most Critical Accident Risk in Oil and Gas Construction // Construction Management and Economics. Vol. 29. 2011, no. 4. Pp. 341–353. https://doi.org/10.1080/01446193.2010.545993
6. Cooper K.G. The $2,000 Hour: How Managers Influence Project Performance through the Rework Cycle // Project Management Journal. Vol. 25. 1994, no. 1. Pp. 11–24.
7. Graham A.K. Beyond PM 101: Lessons for Managing Large Development Programs // Project Management Journal. Vol. 31. 2000, no. 4. Pp. 7–18.

8. Holzmann G. Appendix D – Software Complexity // NASA Study of Flight Software Complexity: Final Report / NASA Office of Chief Engineer. 2009. https://www.nasa.gov/pdf/418878main_FSWC_Final_Report.pdf
9. Holzmann G.J. Conquering Complexity // Computer. Vol. 40. 2007, no. 12. Pp. 111–113. https://doi.org/10.1109/MC.2007.419
10. Leffingwell D. Calculating Your Return on Investment from More Effective Requirements Management // American Programmer. Vol. 10. 1997, no. 4. Pp. 13–16.
11. Lyneis J.M., Cooper K.G., Els S.A. Strategic Management of Complex Projects: A Case Study Using System Dynamics // System Dynamics Review. Vol. 17. 2001, no. 3. Pp. 237–260. https://doi.org/10.1002/sdr.213
12. Lyneis J.M., Ford D.N. System Dynamics Applied to Project Management: A Survey, Assessment, and Directions for Future Research // System Dynamics Review. Vol. 23. 2007, no. 2-3. Pp. 157–189. https://doi.org/10.1002/sdr.377
13. NASA Study of Flight Software Complexity: Final Report / NASA Office of Chief Engineer. 2009. https://www.nasa.gov/pdf/418878main_FSWC_Final_Report.pdf
14. Neves F.G., Borgman H., Heier H. Success Lies in the Eye of the Beholder: The Mismatch between Perceived and Real IT Project Management Performance // Proceedings of the 49th Annual Hawaii International Conference on System Sciences (5–8 January 2016, Kauai, Hawaii). Los Alamitos, CA: IEEE Computer Society. 2016. Pp. 5878–5887.
15. Roberts E.B. The Dynamics of Research and Development. NY : Harper & Row. 1964.
16. Roberts E.B. A Simple Model of R&D Project Dynamics // R&D Management. Vol. 5. 1974, no. 1. Pp. 1–15. https://doi.org/10.1111/j.1467-9310.1974.tb01217.x
17. Richardson G.P., A. Pugh A.L., III. Introduction to System Dynamics Modeling with Dynamo. Cambridge. MA : MIT Press. 1981.
18. Suslov S., Katalevsky D. Modeling and Simulation Toolset // Evolving Toolbox for Complex Project Management / Ed. by A. Gorod, L. Hallo, V. Ireland, I. Gunawan. Boca Raton. FL : CRC Press. 2020. Pp. 417–450.
19. Taylor T., Ford D.N. Tipping Point Failure and Robustness in Single Development Projects // System Dynamics Review. Vol. 22. 2006, no. 1. Pp. 51–71. https://doi.org/10.1002/sdr.330
20. Williams T., Eden C., Ackermann F., Tait A. The Effects of Design Changes and Delays on Project Costs // Journal of the Operational Research Society. Vol. 46. 1995, no. 7. Pp. 809–818.

References

1. Katalevsky D.Yu. Fundamentals of Management Simulation Modeling and System Analysis. M. : Delo. 2015.
2. McConnell S. Code Complete. Master Class / Trans. from English. M. : Russkaya redaktsiya. 2010.
3. Abdel-Hamid T.K. A Multiproject Perspective of Single-Project Dynamics // Journal of Systems and Software. Vol. 22. 1993, no. 3. Pp. 151–165.

4. Boehm B.W. Software Engineering Economics. Englewood Cliffs. NJ : Prentice-Hall. 1981.
5. Chan M. Fatigue: The Most Critical Accident Risk in Oil and Gas Construction // Construction Management and Economics. Vol. 29. 2011, no. 4. Pp. 341–353. https://doi.org/10.1080/01446193.2010.545993
6. Cooper K.G. The $2,000 Hour: How Managers Influence Project Performance through the Rework Cycle // Project Management Journal. Vol. 25. 1994, no. 1. Pp. 11–24.
7. Graham A.K. Beyond PM 101: Lessons for Managing Large Development Programs // Project Management Journal. Vol. 31. 2000, no. 4. Pp. 7–18.
8. Holzmann G. Appendix D – Software Complexity // NASA Study of Flight Software Complexity: Final Report / NASA Office of Chief Engineer. 2009. https://www.nasa.gov/pdf/418878main_FSWC_Final_Report.pdf
9. Holzmann G.J. Conquering Complexity // Computer. Vol. 40. 2007, no. 12. Pp. 111–113. https://doi.org/10.1109/MC.2007.419
10. Leffingwell D. Calculating Your Return on Investment from More Effective Requirements Management // American Programmer. Vol. 10. 1997, no. 4. Pp. 13–16.
11. Lyneis J.M., Cooper K.G., Els S.A. Strategic Management of Complex Projects: A Case Study Using System Dynamics // System Dynamics Review. Vol. 17. 2001, no. 3. Pp. 237–260. https://doi.org/10.1002/sdr.213
12. Lyneis J.M., Ford D.N. System Dynamics Applied to Project Management: A Survey, Assessment, and Directions for Future Research // System Dynamics Review. Vol. 23. 2007, no. 2-3. Pp. 157–189. https://doi.org/10.1002/sdr.377
13. NASA Study of Flight Software Complexity: Final Report / NASA Office of Chief Engineer. 2009. https://www.nasa.gov/pdf/418878main_FSWC_Final_Report.pdf
14. Neves F.G., Borgman H., Heier H. Success Lies in the Eye of the Beholder: The Mismatch between Perceived and Real IT Project Management Performance // Proceedings of the 49th Annual Hawaii International Conference on System Sciences (5–8 January 2016, Kauai, Hawaii). Los Alamitos. CA : IEEE Computer Society. 2016. Pp. 5878–5887.
15. Roberts E.B. The Dynamics of Research and Development. NY : Harper & Row. 1964.
16. Roberts E.B. A Simple Model of R&D Project Dynamics // R&D Management. Vol. 5. 1974, no. 1. Pp. 1–15. https://doi.org/10.1111/j.1467-9310.1974.tb01217.x
17. Richardson G.P., A. Pugh A.L., III. Introduction to System Dynamics Modeling with Dynamo. Cambridge. MA : MIT Press. 1981.
18. Suslov S., Katalevsky D. Modeling and Simulation Toolset // Evolving Toolbox for Complex Project Management / Ed. by A. Gorod, L. Hallo, V. Ireland, I. Gunawan. Boca Raton. FL : CRC Press. 2020. Pp. 417–450.
19. Taylor T., Ford D.N. Tipping Point Failure and Robustness in Single Development Projects // System Dynamics Review. Vol. 22. 2006, no. 1. Pp. 51–71. https://doi.org/10.1002/sdr.330
20. Williams T., Eden C., Ackermann F., Tait A. The Effects of Design Changes and Delays on Project Costs // Journal of the Operational Research Society. Vol. 46. 1995, no. 7. Pp. 809–818.