Win March. Adaptive Project Management: Principles and Examples

Arkhipenkov Sergey
email: sarkhipenkov@gmail.com

Abstract

Such important subject as management of a software developer team is considered in this report. It is known that working efficiency of software developers can vary tenfold. That's why the manager's task is to make reproducible the high efficiency of intellectual activity. Ways and means to solve this problem are in application of adaptive project management intended for study and amendment of characteristics and structure of the object of management, notably people and their cooperation. In this report the author submits seven management principles. By means of them a manager can ensure top team efficiency. Application of these principles is illustrated by examples. The author demonstrates that well-managed project can be successfully carried out by the ordinary developer team.

Keywords: *team management, leadership, project management, adaptive management.*

Марш победителей. Адаптивное управление проектом: принципы и примеры

Архипенков Сергей
email: sarkhipenkov@gmail.com

Аннотация

В докладе рассмотрены важные вопросы руководства командой разработчиков программного обеспечения. Известно, что производительность программистов может отличаться в десятки раз. Задача руководителя - сделать воспроизводимой высокую эффективность интеллектуальной деятельности. Путь к решению этой задачи – применение методов адаптивного управления, направленных на изучение и изменение свойств и структуры объекта управления: людей и их взаимодействия. Представлены семь принципов адаптивного управления проектом, используя которые, руководитель может обеспечивать наивысшую производительность команды. Применение принципов иллюстрируется примерами. Автор показывает, что хорошо управляемый проект может быть успешно выполнен обычной командой разработчиков.

Ключевые слова*: руководство командой; лидерство; управление проектом; адаптивное управление.*

Введение. Классические методы управления не работают

Уместно провести аналогию между классическими методами, применяемыми в системах автоматического управления летательными аппаратами, и подходами к управлению программными проектами.

«Как получится». Разомкнутая система управления. Например, провели kick-off meeting, поставили задачи разработчикам и пошли готовить банкет по поводу успешного завершения проекта. Аналогия: баллистический полет. Можно, но недалеко и неточно.

«Водопад». Жесткое управление с обратной связью. Расчет опорной траектории (план проекта), измерение отклонений, коррекция и возврат на опорную траекторию. Лучше, но не эффективно.

«Гибкое управление». Расчет опорной траектории, измерение отклонений, расчет новой попадающей траектории и коррекция для выхода на нее. «Планы - ничто, планирование - все» (Эйзенхауэр, Дуайт Дэвид)

«Метод частых поставок». Самонаведение. Расчет опорной траектории, измерение отклонений, уточнение цели, расчет новой попадающей траектории и коррекция для выхода на нее.

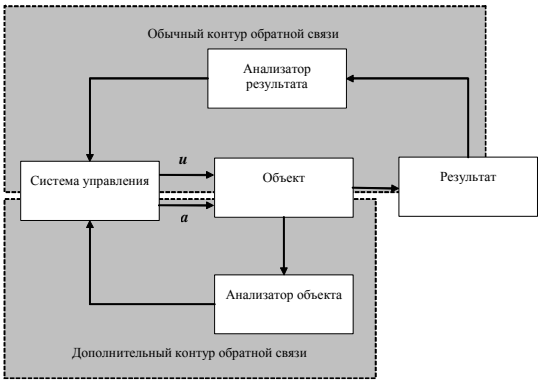
Классические методы управления перестают работать в случаях, когда структура и свойства управляемого объекта нам не известны и изменяются со временем. Эти подходы так же не помогут, если текущие свойства не позволяют объекту двигаться с требуемыми характеристиками. Например, летательный аппарат не может развить требуемое ускорение или разрушается при недопустимой перегрузке. Аналогично, если рабочая группа проекта не может обеспечить необходимую производительность и постоянно работает в режиме аврала, то это приводит к уходу профессионалов из проекта.

Адаптивное управление

Адаптивное управление, дополнительно к прямым управляющим воздействиям, направлено на изучение и изменение свойств управляемого объекта. Продолжая аналогию с управлением летательными аппаратами, - это расчет опорной траектории, измерение отклонений, уточнение цели, уточнение объекта управления, адаптация (необходимое

изменение) объекта управления, расчет новой попадающей траектории и коррекция для выхода на нее.

Для того чтобы понять структуру и свойства объекта и воздействовать на него с целью их приведения к желаемому состоянию, в проекте должен быть дополнительный контур обратной связи (Рисунок 1).



u – управляющее воздействие; ***a*** – адаптационное воздействие.

Рисунок 1. Дополнительный контур обратной связи в системе адаптивного управления

Результатом управления проектом являются: соответствие спецификациям, срок и бюджет.

Множество управляющих воздействий (***u***) ограничено: составить план работ, расставить приоритеты, назначить на работы исполнителей. Как правило, таким и только таким управлением (администрированием) занимаются менеджеры - поклонники диаграмм Ганта и фанаты MS Project. Если у команды проекта низкая производительность, то единственный путь ее повысить при административном подходе - это постоянное давление, авралы, сверхурочные и субботники. Работать больше, это совсем не значит - работать продуктивнее. Скорее наоборот. Излишнее давление и суета приводят к непродуманным решениям, большому проблемному коду и многочисленным последующим переработкам. Для подобных руководителей любой программный проект будет безнадежным.

Вместе с тем, известно, что производительность разных программистов может отличаться в десятки раз. Утверждаю, что производительность одного и того же программиста может так же отличаться в десятки раз. Заставьте лучшего в мире бегуна бегать в мешке, и он покажет в 10 раз худший результат. Заставьте лучшего программиста заниматься «сизифовым трудом»: плодить документацию (которую, как правило, никто не читает) в угоду «Методологии» (именно с большой буквы ‘М’), - и его производительность снизится в 10 раз.

Поэтому, основные усилия руководителя, если он стремится получить наивысшую производительность рабочей группы, должны быть направлены на изучение и изменение объекта управления: людей и их взаимодействия. Следовательно, задачу адаптивного управления мы можем разделить на две подзадачи:

1. Обеспечить эффективность каждого участника рабочей группы.
2. Обеспечить эффективные процессы взаимодействия.

Все люди разные и ситуаций, в которых они могут находиться в ходе проекта, бесчисленное множество. Бойтесь стереотипов. Если вы не учитываете индивидуальные особенности конкретной личности, то эффективность ваших взаимодействий сильно снижается.

Модель объекта управления нам неизвестна, следовательно, не может существовать исчерпывающий набор правил, тип «если…, то…», по которым смог бы действовать руководитель. Поэтому, сколько людей и ситуаций, столько и вариантов решений должен иметь эффективный руководитель в своем запасе. «Если у руководителя в руках только молоток, то все вокруг будут похожи на гвозди».

Принцип 1. *«Принцип достаточного разнообразия». Для «хорошего» управления количество возможных состояний управляющего устройства (разнообразие) должно быть не меньше, чем количество состояний объекта управления [1].*

Руководитель при поиске решения опирается на свой багаж знаний и умений. Он пытается понять каждого участника, классифицировать состояние, найти в своем опыте похожую ситуацию и адаптировать ранее использованное успешное решение применительно к данному конкретному случаю. Таким образом, руководитель стремится помочь человеку (объекту управления) перейти в новое более эффективное с точки зрения целей проекта состояние.

Затем руководитель должен наблюдать за результатами своего воздействия – это и есть дополнительный контур обратной связи. Необходимо помнить, что понять человека можно, только слушая и *слыша*, что он говорит. Руководитель, который в течение недели не пообщался индивидуально с каждым из своих

прямых подчиненных, зря получает зарплату. И совсем не обязательно разговор должен идти о статусе проектных работ. Порой, достаточно поговорить о погоде, кино или футболе.

После этого руководитель анализирует полученные результаты и аккумулирует новый опыт (положительный или отрицательный) в своей «базе знаний». Как это может происходить на практике, будет далее проиллюстрировано на примерах.

Чем опытней руководитель, тем точнее он может распознать и классифицировать сложившуюся ситуацию, тем больше в его «базе знаний» прецедентов, используя которые, он может синтезировать решение для данного конкретного случая. Именно поэтому в управлении программными проектами в первую очередь ценится опыт руководителя и только потом, возможно, его звания и знания.

1-я задача адаптивного управления. Обеспечить эффективность каждого участника

Для того чтобы ваш сотрудник мог эффективно решить поставленную вами задачу, необходимо и достаточно выполнение четырех условий.

Принцип 2. *«4 условия эффективной работы».*

1. *Понимание целей работы.*
2. *Умение ее делать.*
3. *Возможность ее сделать.*
4. *Желание ее сделать.*

Для того чтобы обеспечить выполнение этих условий, руководитель должен уметь эффективно выполнять четыре функции.

Принцип 3. *«4 функции руководителя».*

1. *Направлять.* Если сотрудник не понимает что делать, задача руководителя - обеспечить общее видение целей и стратегии их достижения.
2. *Обучать.* Если сотрудник не умеет, задача руководителя – «обучать», быть наставником и образцом для подражания.
3. *Помогать.* Если у сотрудника не может выполнить работу, задача руководителя – «помогать», обеспечить исполнителя всем необходимым, убрать препятствия с его пути.
4. *Вдохновлять.* Если у сотрудника не достаточно желание выполнить работу, задача руководителя – «вдохновить», обеспечить адекватную мотивацию участника на протяжении всего проекта.

Пример. «Хочет и может, но не делает»

Ситуация. Программист стремится найти наиболее общее решение задачи, учесть все возможные последующие изменения и расширения. Старается разработать самый быстрый алгоритм, требующий минимальных ресурсов. Использует в решении все лучшие практики, паттерны проектирования, самые новые инструменты.

Классификация. Неоправданное усложнение задачи. Программист неадекватно понимает цели проекта и приоритеты. В результате стремления к совершенству - низкая производительность. Для выполнения работы в срок постоянно не хватает времени.

Решение. Направлять. Более четко формулировать цели и расставлять приоритеты. Определить конкретные критерии оценки качества результата. Нацеливать программиста на правильное решение задачи максимально простым способом. Опыт свидетельствует, что более чем в 90% случаев ее не придется переделывать.

Пример. «Хочет, но не может»

Ситуация. Программист своевременно приходит на работу. Не отвлекается. Настойчиво работает над решением поставленных задач. Часто задерживается, чтобы уложиться в срок. В результате сроки, которые он сам оценивает, постоянно срываются. Большой проблемный код.

Классификация. У программиста недостаточно опыта. Он не умеет оценивать и планировать свою работу.

Решение. Учить самому или закрепить за программистом более опытного наставника.

Известно, что ни одна задача не будет решена за любое, отведенное на это время, если человек не захочет ее сделать. Он всегда найдет для оправдания этого 100 «объективных» причин, вместо того, чтобы найти хотя бы одну возможность для решения задачи.

У каждого участника рабочей группы должна быть личная цель (внутренняя мотивация), которую он сможет достичь, продвигая проект к успеху. Начните с себя! Вам нужно четко понимать, в чем состоит ваш выигрыш в случае успешного завершения проекта. Добиться от

участников приверженности проекту больше, чем имеете вы сами, вам не удастся.

Если у участника нет такой личной значимой цели, избавьтесь от него. Иначе вам придется потратить все свое время на «промывание его мозгов» и попытки мотивировать его на эффективную работу.

Необходимо помнить, что по ходу проекта мотивы людей, как правило, изменяются.

Пример. «Может, но не хочет»

Ситуация. Программист имеет глубокие знания и развитый интеллект, быстро осваивает все новое, нацелен на решение трудных задач. Пользуется заслуженным авторитетом среди коллег. В начале проекта активно выдвигал новые идеи, убедительно их обосновывал, добивался их признания всеми. Находил неизвестные возможности, существенно сократившие трудоемкость работ по проекту. В середине проекта потерял интерес. Стал «витать в облаках» и отвлекаться на изучение каких-то новых технологий. Постоянно заваливает сроки, делает глупые ошибки, непростительные для его опыта. Расхолаживающе воздействует на команду.

Классификация. Программист комфортно чувствует себя в роли «генератора идей» и не мотивирован на методичную реализацию.

Решение. Мотивировать. Например, пообещать роль архитектора, если данный проект завершится успешно. Или поручить роль наставника, предоставить возможность передавать свои знания и умения менее опытным коллегам.

Пример. «Может, но не хочет 2»

Ситуация. Программист активен, самостоятелен, напорист. По любому вопросу имеет свое собственное мнение. Всегда стремится быть победителем в конфликтах. Часто оценивает других и указывает им на недостатки. Использует любой повод, чтобы продемонстрировать свое превосходство. Сильно переоценивает свой личный вклад в общее дело и поэтому считает, что он должен работать меньше, чем его «менее способные» коллеги.

Классификация. Человек – эгоист, достиг личной независимости, но неспособен к конструктивному взаимодействию. Вредно воздействует на команду. Самооценка, скорее всего, неадекватно завышенная.

Решение. Избавиться. Если нет такой возможности, найти для него четко специфицированную, изолированную задачу, которая не находится на критическом пути проекта. Иметь под рукой другого специалиста, который сможет решить эту задачу, когда потребуется.

2-я задача адаптивного управления. Обеспечить эффективные процессы взаимодействия

В отрасли программостроения многие уже признали, что наиболее эффективные производственные процессы складываются в самоуправляемых и самоорганизующихся рабочих командах, для которых характерны ясность общих ценностей и целей, самоконтроль, взаимопомощь, взаимозаменяемость, коллективная ответственность за результаты труда, всемерное развитие и использование индивидуального и группового потенциалов.

Эффективные команды не образуются сами по себе, они кристаллизуются вокруг признанного лидера. Как не бывает лидеров без последователей, так и не бывает команд без лидеров. Поэтому первый шаг руководителя при создании эффективной команды - это стать лидером, вокруг которого сможет сплотиться рабочий коллектив.

Принцип 4. *«Принцип лидерства». Руководителю программного проекта недостаточно быть хорошим управленцем, он должен стать признанным лидером.*

Лидера нельзя назначить. Лидер должен быть признан коллективом. Чтобы руководитель получил признание в качестве лидера, необходимо выполнение следующих двух условий.

1. Признание коллективом профессиональной компетентности и превосходства руководителя.
2. Полное доверие коллектива к действиям и решениям руководителя, признание его исключительных человеческих качеств, убежденность в его честности, порядочности, вера в его искренность и добросовестность.

Для того чтобы получить признание, если, конечно, руководитель его объективно заслуживает, он должен использовать принцип номер 5.

Принцип 5. *«4 стратегии лидера». Не существует одной лучшей стратегии руководства. В зависимости от готовности участников рабочей группы выполнять задания*

руководителя, он должен использовать одну из 4-х стратегий [2]:

1. *S1. «Директивное управление».* Руководитель говорит, указывает, направляет, устанавливает. Жесткое назначение работ, строгий контроль сроков и результатов.
2. *S2. «Объяснения».* Лидер "продает", объясняет, проясняет, убеждает. Сочетание директивного и коллективного управления. Объяснение своих решений.
3. *S3. «Участие».* Лидер участвует, поощряет, сотрудничает, проявляет преданность. Приоритетное коллективное принятие решений, обмен идеями, поддержка инициативы подчиненных.
4. *S4. «Делегирование».* Лидер делегирует, наблюдает, обслуживает. «Не мешать» - пассивное управление сформировавшегося лидера.

Пример. «Ситуационное лидерство»

1. *Ситуация*. Вас назначили руководителем в новый коллектив. Вы еще не получили признания, а дело делать надо. *Решение.* Стратегия S1. «Директивное управление».
2. *Ситуация*. Вы были участником команды. Вас назначили руководителем этой команды. Доверие есть, а уверенности в правильности ваших действий нет. *Решение*. Стратегия S2. «Объяснения».
3. *Ситуация*. Вас назначили руководителем в новый коллектив. Все знают о ваших прежних сложных и успешных проектах. Все признают ваше превосходство, но доверия к вам нет. Никто не знает, какой ценой были достигнуты ваши победы. *Решение*. Стратегия S3. «Участие».
4. *Ситуация*. Между вами и участниками установлено взаимное доверие. Все достаточно мотивированы на успех проекта. Каждый сам себе может быть руководителем. *Решение.* Стратегия S4. «Делегирование».

Мало стать лидером, надо еще суметь сплотить коллектив. Эксперты в области командного менеджмента выделяют 4 обязательные последовательные стадии, через которые должна пройти рабочая группа прежде, чем она станет эффективной командой, это:

1. Forming. Формирование. Характеризуется избытком энтузиазма, связанного с новизной. Люди должны преодолеть внутренние противоречия, переболеть конфликтами прежде, чем сформируется действительно спаянный коллектив.
2. Storming. Разногласия и конфликты. Самый сложный и опасный период. Мотивация новизны уже исчезла, а сильные и глубокие стимулы у команды еще не появились. Неизбежные сложности или неудачи порождают конфликты и «поиск виновных». Участники команды методом проб и ошибок вырабатывают наиболее эффективные процессы взаимодействия.
3. Norming. Становление. В команде растет доверие, люди начинают замечать в коллегах не только проблемные, но и сильные стороны. Закрепляются и оттачиваются наиболее эффективные процессы взаимодействия. На смену битве амбиций приходит продуктивное сотрудничество. Четче становится разделение труда, исчезает дублирование функций.
4. Performing. Отдача. Команда работает эффективно, высок командный дух, люди хорошо знают друг друга и умеют использовать сильные стороны коллег. Все стремятся придерживаться выработанных общих процессов. Высок уровень доверия. Это лучший период для раскрытия индивидуальных талантов.

Часто случается, что рабочая группа вязнет на одной из стадий и никогда не достигает плато наивысшей производительности.

Пример. «Шумиха»

Ситуация. Частые смены приоритетов задач. Споры о том, что надо делать, а что не надо. Сомнения в реальности сроков. Недовольство отсутствием прогресса. Много вопросов по каждой задаче.

Классификация. Команда находится на первом этапе образования – «Объединение». Цели и стратегия их достижения не ясны и не приняты всему участниками команды.

Решение. Стратегия S1. «Директивное управление». Функции: «направлять» и «помогать». Четко ставить цели и формулировать стратегию их достижения.

Устанавливать проектные процедуры распределять роли и разграничивать зоны ответственности.

Пример. «Споры и дискуссии»

Ситуация. На совещаниях бесконечные неконструктивные споры и дискуссии. Постоянно доминируют одни и те же лица. Другие предпочитают отмалчиваться. Мнения высказываются как объективные факты. «На самом деле эта задача решается так…!» Постоянно даются оценки. «Это все неправильно!» «Это все не важно!» Присутствует агрессия «Ты, просто, ничего не понимаешь!». «А ты разве не знаешь, что…!»

Классификация. Команда находится на стадии «Разногласия и конфликты». Отсутствие культуры эффективных коммуникаций. Неумение слушать и слышать.

Решение. Стратегия S2. «Объяснения». Функции: «обучать» и «помогать». Стать для команды образцом эффективного взаимодействия. Конфликты необходимо разруливать спокойно, терпеливо и тщательно.

Пример. «Группомыслие»

Ситуация. Девиз участников проекта: «Давайте работать, а не конфликтовать!» Все стараются избегать конфликтов и поддерживать согласие. Как правило, никто не спорит, все соглашаются с мнением руководителя и следуют его указаниям. При возникновении трудных ситуаций, все ждут решения от руководителя. Редкие противоречия разрешаются общим голосованием.

Классификация. В психологии подобная ситуация называется «Парадокс Абилина». Люди принимают решения, основанные не на том, что они сами хотят, но на том, что они думают, что другие хотят. В результате получается, что каждый делает что-то, что никому на самом деле не нужно. Подобное избегание производственных конфликтов снижает здоровую интеллектуальную конкуренцию, ведет к шаблонности и застою.

Решение. Стратегия S3. «Участие». Функции «помогать» и «вдохновлять». Целенаправленно мотивировать инакомыслие и интеллектуальную конкуренцию. Например, назначать рецензентов – критиков, которые должны найти слабые стороны в решении, предлагаемом их коллегой для общего обсуждения. Или поручить двум разработчикам решить одну и ту же критичную для проекта задачу, используя разные подходы или технологии, а затем поручить сравнить и оценить полученные результаты третьему коллеге.

Пример. «Менеджер должен занимать очередь…»

Ситуация. Ни одно предлагаемое участником команды решение не принимается на веру. Все требуют факты для его обоснования. Активно анализируются возможные негативные последствия или упущенные возможности при принятии решения. Конфликты носят исключительно производственный характер. При решении конфликтов активно ищутся взаимовыгодные возможности. Говоря словами Тома Демарко, «менеджер проекта должен занимать очередь, чтобы покритиковать сотрудника, не выполняющего свои обещания».

Классификация. Команда находится на стадии «Отдача». Наконец-то, она достигла плато наивысшей эффективности.

Решение. Стратегия S4. «Делегирование». Функции: «направлять» и «вдохновлять». Поддерживать требуемый уровень мотивации. Быть штурманом, искать новые пути и открывать новые возможности. Постоянно наблюдать и оценивать эффективность всех процессов, применяемых в проекте. Искать ответ на вопросы: «Что лишнее мы делаем?» «Что можно делать проще?» «Что угрожает проекту?». Работать на сокращение ненужных усилий вместо того, чтобы «стремиться к новым героическим подвигам».

Если руководитель не будет прилагать дополнительные усилия команда, рано или поздно, начнет «сползать» с плато наивысшей эффективности в состояние застоя и стагнации (Рисунок 2). Помните, что окружение и команда изменяются по ходу проекта. Прежняя мотивация ослабевает или перестает действовать.

Принцип 6. *«Принцип цикличности». 4 стадии развития команды должны циклически повторяться, чтобы обеспечить непрерывный рост эффективности.*

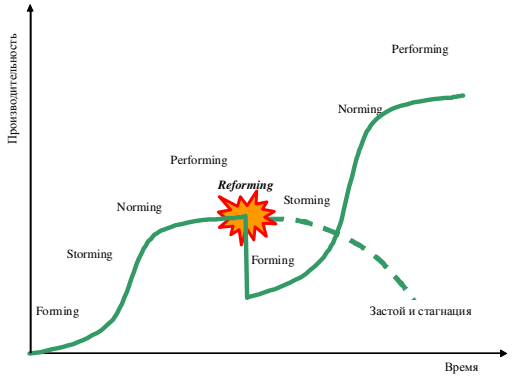


Рисунок 2. Reforming. «Встряхивание» и перевод команды проекта на новый уровень производительности

Изменяйте правила и процессы. Отказывайтесь от того, что перестало действовать или стало работать неэффективно. «Встряхивайте» (Reforming) и возвращайте команду в стадию Forming. Это позволит ей снова, пройдя через все этапы становления, выйти на новый более высокий уровень производительности.

Разумеется, делать это следует, после сдачи очередного релиза программного продукта, ну и, возможно, в случае глубокого кризиса проекта.

Заключение

Ранее мы говорили, что результатом управления проектом являются: соответствие спецификациям, срок и бюджет. Утверждаю, что у успешного программного проекта должен быть еще один, четвертый результат: *каждый участник команды уходил с работы в 18:00 с чувством победы.* Чувство победы – это чувство удовольствия человека, который приближается к своей цели. А все, что человек делает с удовольствием, он делает максимально эффективно.

Принцип 7. *«Принцип победителей». Программист состоит из четырех компонентов: тело, сердце, разум и душа.*

1. *Телу необходимы деньги и безопасность.*
2. *Сердцу - любовь и признание.*
3. *Разуму – развитие и самосовершенствование.*
4. *Душе – самореализация.*

Предоставьте все это вашим сотрудникам, и эффективность их труда возрастет многократно.

В противном случае люди, которые хотят побеждать, найдут все это в другой команде, а в вашей останутся только неудачники.

Ссылки

[1] У.Р.Эшби “Введение в кибернетику” М, ИЛ, 1959

[2] Hersey P., Blanchard K.H. “Management of Organizational Behavior”, 6th ed., Englewood Cliffs: Prentice-Hall, 1993.