Модель системы управления проектом информационного хранилища и методика ведения проекта на ее основе

© 2010 Ю.Ю. Красильников, В.С. Белов
Московский государственный университет экономики, статистики и информатики (МЭСИ)
E-mail: EGracheva@rector.mesi.ru

В статье содержатся результаты формализации предметной области управления проектом создания информационного хранилища и методика ведения проекта. Представленные материалы могут быть полезны для изучения руководителям проектов различных уровней и применены для любых проектов создания информационных технологий.

Ключевые слова: управление проектами, математическая формализация, ведение проектов, алгоритмизация, информационные технологии, информационное хранилище (ИХ).

Управление проектами информационных технологий (ИТ-проектами), к которым относятся проекты информационного хранилища (ИХ) - относительно новый для науки объект управления. В статье представлена формализация модели системы управления проектами информационного хранилища и описана методика ее исполнения. Предлагаемые в книгах по управлению проектами подходы не содержат метод математической формализации проекта. Данный же подход основан на теоретико-множественном подходе. Его преимущество состоит в однозначности понимания языка математики и относительно простой реализации компьютерных алгоритмов на его основе на практике.

Под проектом понимается уникальная задача, которая разрабатывается для достижения характерного результата и которая требует некоего множества ресурсов и ограничена во времени. Далее, под проектом подразумевается некий перечень проблем, которые формирует сам по себе проект:

1. Формирование новых и неизвестных задач.
2. Изменения в ежедневной работе сотрудников.
3. Проблемы с поиском нужных людей в нужное время и их доступность для проектной деятельности.
4. Следование строгим срокам выполнения.

Под управлением проектом будем понимать совокупность управляющих воздействий для достижения целей проекта.

На основании данных определений модель должна удовлетворять следующим требованиям:

1. Содержать информацию о целях проекта.
2. Описывать этапы проекта и их взаимосвязь и последовательность.
3. Описывать все потоки информации, которые возникают в ходе реализации проекта.
4. Содержать описание этапов в части информации.

4.1. О взаимосвязях этапов и целей проекта.

4.2. О важности этапов в рамках всего проекта.

4.3. Об исполнителях, их должностях и отношению к владельцу ИХ.

4.4. О плановых и фактических сроках и продолжительностях этапов.

4.5. О плановых и фактических уровнях выполнения этапов.

Ввиду особенностей реализации ИТ-проектов часть этапов проекта идет последовательно: следующий этап не может начинаться, пока не закончится предыдущий; часть этапов идет параллельно, дополняя друг друга или идя независимо.

С учетом вышеизложенных требований проект внедрения информационного хранилища P представляется следующим универсумом:

$$P = \{S_i : i=1,k\},$$

где S_i - i-й этап построения ХД;

k - общее количество этапов проекта P.

Каждый из этапов проекта ИХ описывается следующим кортежем атрибутов:

$$S_i=<NamS_i,\ EventsInS_i,\ EventOutS_i, HRS_i,\ InitT_i,\ T_i,\ InputS_{ij},\ OutputS_{ij},\ Tasks_i,\ EffS_i, W_i,\ TrgS_i>,$$

где $NamS_i$ - имя i-го этапа;

$EventsInS_i$ - событие, инициирующее выполнение i-го этапа;

$EventOutS_i$ - событие, являющееся результатом i-го этапа.

$EventsInS_i$ и $EventOutS_i$ принадлежат множеству значений {Events}, описывающих все события, которые могут быть сгенерированы в рамках проекта информационного хранилища.

Множество {*Events*} описывается кортежем:

$$Events = < G : G=1,k,\ NamEv_G,\ RTO >,$$

где G - порядковый номер события;

$NamEv_G$ - название события;

RTO - номер этапа, который генерирует данное событие. Предполагается, что событие под номером *G* может быть сгенерировано только на одном из этапов проекта.

Дополнительно введем в модель множество {*ExecEvents*}. Оно является подмножеством {*Events*} и содержит в себе порядковые номера событий, которые уже произошли. В каждый конкретный момент времени состав этого множества зависит от состояния каждого из этапов проекта.

Далее в кортеже представлены аспекты проекта, отражаемые его атрибутами.

Кадровый аспект. HRS_i - множество людей, участвующих в реализации этапа *i*. Данное множество заполняется на основании содержания задач в рамках проекта и ответственных за задачи (множество Tasks) и входит в множество *HR*.

Общее множество людей, которые участвуют в проекте, - *HR*. Кортеж атрибутов:

$$HR = < NamHR,\ Position,\ PrjRole >,$$

где *NamHR* - ФИО сотрудника;
Position - его должность в компании;
PrjRole - роль сотрудника на проекте.

Временной аспект. T_i - кортеж длительностей выполнения *i*-го этапа. Размерность входящих в него множеств - время. В ходе реализации проекта каждый руководитель сам определяет, какой масштаб времени использовать оптимально.

$$T_i = \{PlanT_i; ActT_i\},$$

где $PlanT_i$ - плановое время реализации *i*-го этапа;
ActTi - фактическое время реализации *i*-го этапа,
$InitIT_i$ - кортеж времени начала реализации *i*-го этапа;
$InitIT_i = \{PlanIT_i; ActIT_i\}$,
где $PlanIT_i$ - плановое время начала реализации;
$ActIT_i$ - фактическое время начала реализации.

Для обозначения последовательности этапов должны выполняться следующие неравенства:

$$PlanIT_{i-1} \le PlanIT_i, i=1,k,$$
$$ActIT_{i-1} \le ActIT_i, i=1,k.$$

Время начала реализации этапа *i* не может наступить раньше, чем произойдут все события, его инициирующие:

$$EventsInS_i \in \{ExecEvents\}.$$

Плановое время начала реализации этапа описывается выражением:

$$PlanIT_i = \max_{f=1,i-1} (PlanIT_f + PlanT_f)$$
$$\text{при } f = RTO_i.$$

Информационный аспект. В рамках этапов проекта существуют определенные потоки информации, которые позволяют выполнять работы на следующих этапах наиболее корректным образом. Эта информация привязана к этапам проекта следующими атрибутами: $InputS_{ij}$ - *j*-я единица входящей информации *i*-го этапа; $OutputS_{il}$ - l-я единица исходящей информации *i*-го этапа. $InputS_{ij}$ и $OutputS_{il}$ входят в множество {Information}, описывающее все единицы информации, которые входят в этапы проекта:

$$Information = <Y:\ Y=1,y,$$
$$NamInf_y,\ IType,\ RTO>,$$

где *Y* - единица информации. Предполагается, что все единицы информации уникальны, номер по порядку является ключом;
$NamInf_y$ - описание единицы информации;
IType - тип информации, к которому относится данная единица информации.

Определяется кортежем:

$$IType = <N, TypeNam>,$$

где *N* - порядковый номер типа информации;
TypeNam - название типа информации (структурированный файл, регламентный документ, письмо, и т.д.);
RTO - номер этапа, который является родителем данной информации.

Предполагается, что каждая единица информации ($Information_Y$) может быть порождена только на соответствующем этапе.

Задачи этапа

$Tasks_i$ - кортеж множеств задач *i*-го этапа, содержит два множества всех задач, которые могут быть реализованы на *i*-м этапе.

$$Tasks_i = <PlanTs_i,\ ActTs_i>,$$

где $PlanTs_i$, $ActTs_i$ - плановый перечень задач и фактический перечень задач, соответственно.

$$PlanTs_i = <TaskNam,\ HR, TaskType>,$$
$$ActTs_i = <TaskNam,\ HR, TaskType>,$$

где *TaskNam* - название задачи;
HR - множество людей, которые будут выполнять эту задачу;
TaskType - тип задачи, например, интервью, формализация отчета.

Для каждого этапа перечень задач свой.

Уровень выполнения этапа проекта. $EffS_i$ - выполнение *i*-го этапа. Определяется кортежем:

$$EffS_i = <EffMinS_i, PlanEffS_i,\ ActEffS_i>,$$

где $EffMinS_i$ - минимальный уровень выполнения, ниже которого нельзя считать данный этап выполненным и приступать к выполнению последующих этапов, зависящих от этого;
$PlanEffS_i$ - плановый уровень выполнения *i*-го этапа, закладывается руководством для достижения определенного уровня эффективности;
$ActEffS_i$ - фактический уровень выполнения *i*-го этапа.

Единица измерения данных трех показателей - процент. Для этапа *i* за 100% берется весь перечень задач, который был определен при составлении множества $Tasks_i$. Важно заметить, что минимальный, плановый и фактический уровни

выполнения этапов могут отличаться. Например, следующий этап проекта может начинаться, если текущий выполнен уже на 80%. Вместе с этим для конкретного этапа может быть выполнено больше задач, чем планировалось, т.е. уровень выполнения будет больше 100%.

W_i - значимость этапа проекта для всего проекта в целом. Предполагается, что у каждого этапа есть свой вес, абстрактный показатель меньше единицы, который определяется руководством проекта на основании экспертных оценок.

Аспект целей проекта. $TrgS_i$ - множество целей проекта, которые связаны с данным этапом проекта. $TrgS_i \subset Targets$, где *Targets* - все множество целей проекта.

$$Targets = <TargNam, Description>,$$

где *TargNam* - название цели,

Description - описание цели.

Методика ведения проекта

В соответствии с данной моделью системы управления ИТ-проектом представим следующую методику по формализованному ведению проекта, состоящую из этапов:

1. Создание плана проекта.

2. Ведение проекта по факту исполнения

Процесс планирования проекта представлен на диаграмме (рис. 1).

Кратко опишем этапы данного процесса:

1. Первичное определение этапов проекта S_i. На этом этапе мы заполняем значения $NamS_i$ в кортежах множества S_i. Определение этапов S_i - процесс итерационный, но чем точнее будут определены этапы сначала, тем меньше будет кардинальных изменений в структуре проекта. Кроме того, сразу надо отметить, как именно эти этапы проекта влияют на выполнение общих целей проекта.

2. Каждый из этапов после завершения порождает событие. Эти события надо формализовать и определить в множестве *EventOut*.

3. Так как этапы взаимосвязаны между собою, определяем последовательность этапов проекта информационного хранилища, задавая *EventsIn*. На данном этапе взаимосвязи проставляются первично, не задаваясь высокой точностью, но позволяя увидеть общую структуру проекта.

4. Для каждого этапа определяем начальное плановое множество задач *Tasks*, которые необходимо решить.

5. На основании задач, которые должны быть решены в рамках каждого из этапов, уточняем связи с другими этапами проекта, доопределяя *EventsIn*.

6. В блоке 4 определено все множество задач, которые можно решить, но в реальности продолжение реализации проекта наступает раньше выполнения всех задач из множества $Tasks_i$, а по достижении определенных результатов, определенных в кортеже $EffMin_i$.

7. На основании задач из множества *Tasks* определяем примерное время реализации данного этапа проекта.

8. Определяем множество входящей информации, необходимой для реализации этапа, и множество исходящей информации, получающейся в результате реализации задач *Tasksi*.

9. Анализируя множество необходимой информации, решаем, достаточно ли формируется единиц информации на других этапах.

10. Если единиц информации недостаточно, то определяем, может ли она быть порождена на других этапах или необходимо выделить дополнительные этапы реализации информационного хранилища.

11. Анализируя получившиеся этапы проекта информационного хранилища, отвечаем на вопрос: реализует ли данный проект все цели, заложенные руководством предприятия или холдинга.

12. Определяем дополнительные этапы проекта, в случае отрицательного ответа переходим на блоки 9 и 10.

На основании данного итеративного алгоритма получаем укрупненный план проекта с перечнем задач и событий каждого этапа. Полученный по результатам моделирования план требуется поддерживать в актуальном состоянии для дальнейшего управления проектом.

Следующим мероприятием является ведение проекта по факту. Основываясь на практике, можно сделать вывод, что это более сложный комплекс работ, состоящий из нескольких частей:

1) изменение проекта в соответствии с изменяющимися целями организации;

2) отражение фактически прошедших этапов.

Далее представлено описание каждой из частей в виде комплекса работ по ведению проекта.

Изменение проекта в соответствии с изменяющимися целями организации

В условиях неопределенности и нестабильности рыночной ситуации цели проекта не будут оставаться неизменными в течение всего хода проекта. По этой причине множество *Targets* может претерпевать изменения:

$$Targets \rightarrow Targets'.$$

Обновленный состав целей (ОСЦ) может быть определен советом директоров, финансовым директором или управляющим комитетом проекта. После представления ОСЦ проекта необходимо проанализировать структуру плана проекта и внести соответствующие изменения. Предлагается следующий алгоритм анализа (рис. 2):

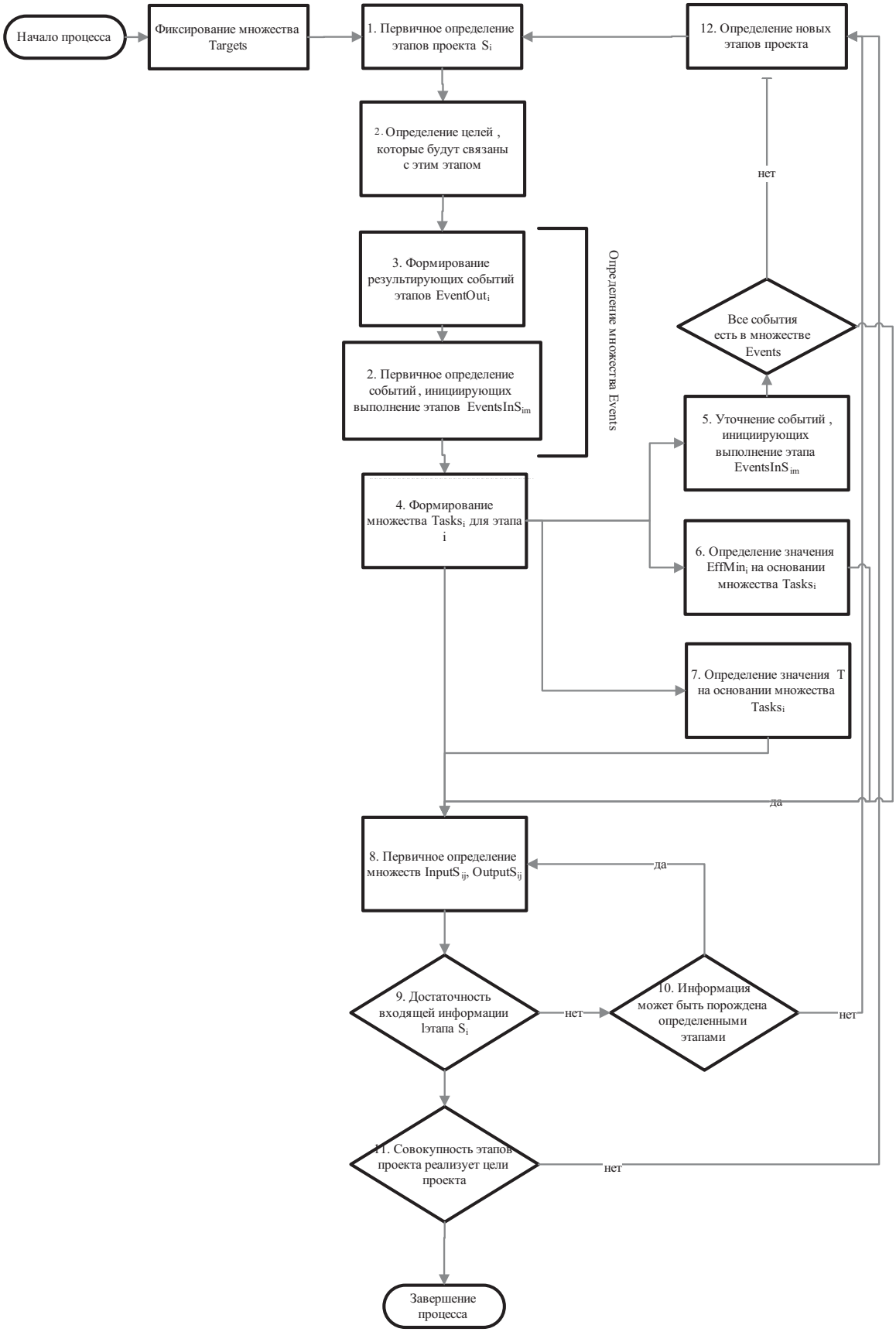

Рис. 1. **Этапы реализации методики ведения проекта по созданию информационного хранилища**

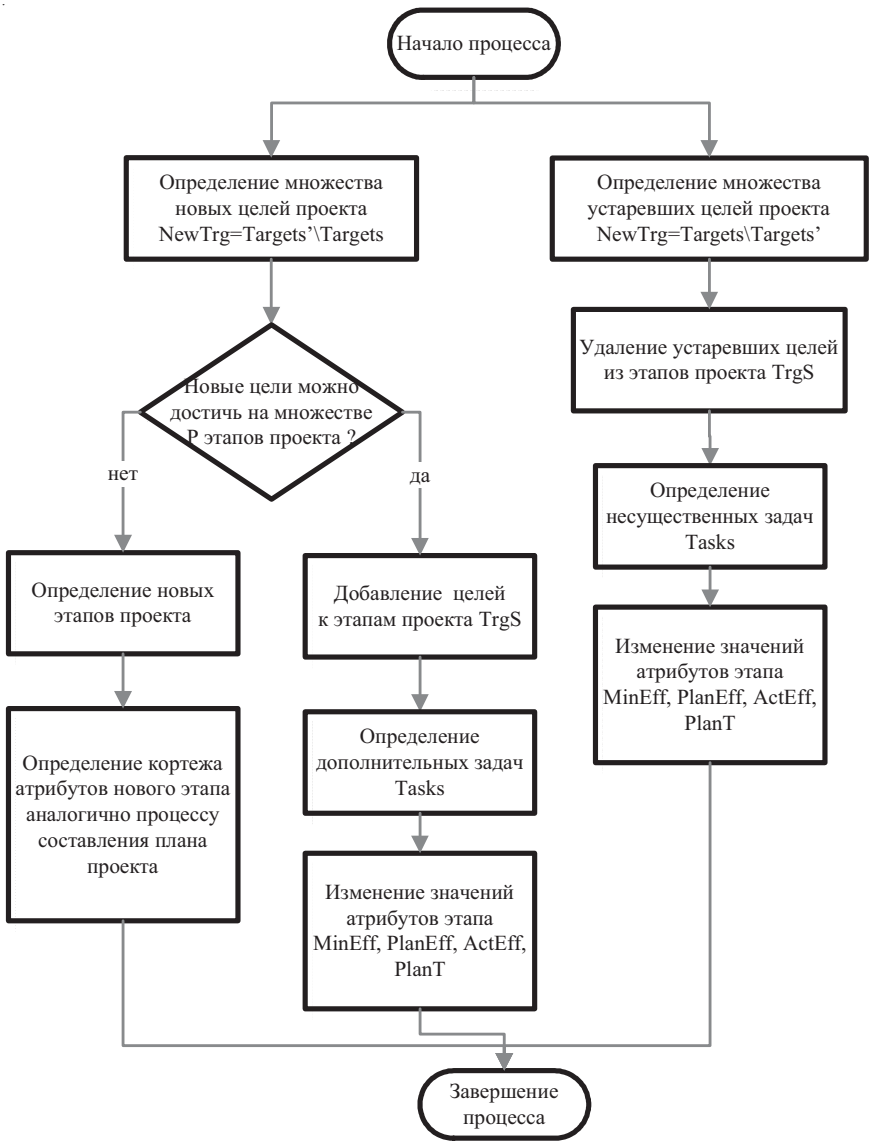

Рис. 2. **Изменение целей проекта**

1. Из ОСЦ проекта определить множество целей, которые еще не были учтены в структуре проекта.

1.1. Если возможно учесть эти цели проекта в процессе выполнения уже существующих этапов проекта, то добавить их к кортежам множества $TrgS_i$ соответствующих этапов проекта. Дополнить множества задач этапа проекта и заново оценить время и уровни реализации проекта.

1.2. Если в текущее множество этапов нельзя добавить новые цели, то необходимо сформировать новые этапы проекта, пользуясь вышеупомянутой структурой формирования новых этапов проекта.

2. В составе обновленного множества целей проекта выделить множество устаревших целей. Исключить их из множества целей каждого из этапов проекта, одновременно уточнив множество $Tasks_i$ и скорректировав время и уровни реализации этапов.

Отражение фактически прошедших этапов

Управление проектом по факту помогает ответить на ряд вопросов:

1. Отслеживание выполнения всех задач каждого этапа проекта.

2. Оценка корректности планирования задач этапа.

3. Снижение задержек между этапами проекта. Модель проекта однозначно определяет, что по завершении этапа i можно начинать этап i+1.

4. Отслеживание затрат времени на выполнение этапов проекта и сравнение с плановыми значениями.

5. Оценка эффективности реализации этапов проекта на основании плановых и фактических значений уровней качества, в том числе затрат времени.

6. Корректировка плановых сроков выполнения следующих этапов проекта в соответствии с фактически прошедшими этапами.

Для выполнения названных мероприятий предлагается следующий алгоритм:

1. Определение фактических значений временных показателей. Заполняется в соответствии с диаграммой (рис. 3).

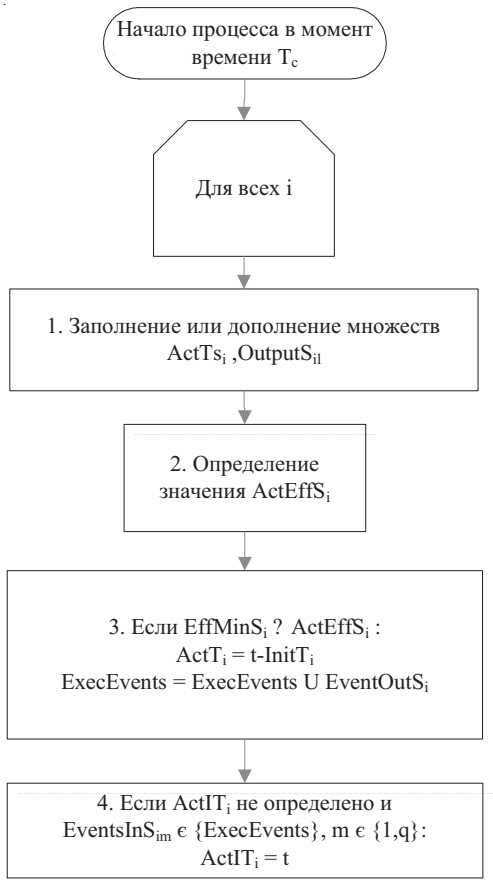

Рис.3. **Процесс фиксации значений фактических параметров модели**

1.1. В соответствии с текущим уровнем выполнения проекта определяются фактически выполненные задачи и фактически полученная информация на каждом из этапов, которые реализуются в данное время.

1.2. В соответствии с выполненными задачами определяется уровень эффективности выполнения этапа.

1.3. Если уровень $ActEff_i$ на этапе уже превысил минимальный, этап считается выполненным с точки зрения начала реализации следующих этапов. Во множество *ExecEvents* добавляется событие завершенности *i*-го этапа.

1.4. Для каждого этапа, который по факту еще не начал выполняться, проверяется, все ли события, которые его инициируют, выполнены. Если это так, отмечается фактическое значение начала этапа *i*. Задержки в начале выполнения *i*-го этапа не происходит.

2. Оцениваются множества плановых и фактически выполненных задач при реализации *i*-го этапа.

2.1. $PlanTs_i = ActTs_i$, значит, все плановые задачи были выполнены, во время реализации этапа не было сделано ничего не запланированного. На практике такой случай встречается достаточно редко.

2.2. $PlanTs_i <> ActTs_{i.}$ В этом случае возможны варианты.

2.2.1. $ActTs_i \subset PlanTs_i$. Был запланирован определенный перечень задач, но не все задачи были выполнены. Назовем это множество $UnDnTs_i$, оно будет определяться формулой

$$UnDnTs_i = PlanTs_i \setminus ActTs_i.$$

Анализируя это множество, руководитель проекта может оценить, какие еще задачи нужно выполнить на текущем этапе проекта или какие задачи оказались неактуальными, тем самым корректируя последующие этапы проекта.

2.2.2. $PlanTs_i \subset ActTs_{i.}$. Был запланирован определенный перечень задач, но по факту пришлось выполнить большее количество задач для реализации этапа. Назовем это множество задач $OverTs_i$, оно будет определяться формулой:

$$OverTs_i = ActTs_i \setminus PlanTs_i.$$

Анализируя это множество, руководитель проекта может дополнять множества задач последующих этапов, которые имеют схожие действия для того, чтобы корректно планировать время и ресурсы на последующие этапы (например, для будущего этапа z):

$$PlanTs'_z = PlanTs_z + AddTs,$$

где $AddTs \subset OverTs_i$.

2.2.3. $PlanTs_i \setminus ActTs_{i.} \neq ActTs_i \setminus PlanTs_i \neq ActTs_i \cap PlanTs_i \neq \varnothing$. Выражение определяет наличие задач, которые еще не были выполнены по факту, и задачи, которые были выполнены по факту, но не были запланированы. В этом случае требуется пользоваться комбинацией методов 2.2.1 и 2.2.2. для решения задач проекта.

3. Оцениваются временные затраты на этапе *i*.

3.1. При достижении на этапе значения $MinEff_i$ сравниваются $PlanT_i$ и $ActT_i$. Ищутся причины задержки выполнения этапа или выполнения этапа раньше запланированного срока. Формируется два множества задач этапа проекта, являющихся подмножеством $ActTs_i$:

$IncTs_i$ - множество задач этапа, которые были выполнены медленнее, чем ожидалось, повлияли на увеличение продолжительности этапа.

$DcrTs_i$ - множество задач этапа, которые были выполнены быстрее, чем планировалось, повлияли на уменьшение продолжительности этапа.

Чтобы принять во внимание сроки выполнения этих задач к последующим этапам проекта, необходимо включить в множество проверки *CheckS* последующие этапы с помощью соотношения:

$$S_{i+z} \subset CheckS: task \Rightarrow \exists task \quad IncTs_i$$
или $task \subset DcrTs_i$ и $task \subset TasksS_{i+z}$.

3.2. Сравнивается эффективность выполнения этапа относительно плановых значений (единица измерения: уровень процента выполнения в единицу времени, в соответствии с размерностями эффективности выполнения и продолжительности):

$$Effect_i = \frac{ActEff_i}{ActT_i} - \frac{PlanEff_i}{PlanT_i}.$$

1. $Effect < 0$ - план был составлен некорректно. Итоговый результат был достигнут с меньшей производительностью. Руководителю необходимо проанализировать причины снижения производительности и учесть их на последующих этапах при выполнении аналогичных задач.

2. $Effect > 0$ - производительность выше, чем была заложена в плане. В среднем все задачи этапа были реализованы лучше, чем рассчитывалось. Руководителю проекта желательно проанализировать задачи этапа и учесть аналогичные задачи в последующих этапах для оптимизации времени выполнения.

Для приведения сроков реализации последующих этапов к более реалистичным вводим коэффициент пересчета будущих этапов, сходных с данным:

$$Coeff = \frac{PlanEff \cdot ActT}{ActEff \cdot PlanT}.$$

Используем его. Для этого продолжительность последующих этапов, сходных с рассчитанным этапом, умножаем на данный коэффициент.

Представленная модель и методика ведения проекта являются одними из вариантов ведения проекта, которые возможно реализовать в ходе ведения ИТ-проекта. Они учитывают особенности обязательной последовательности этапов, возможные изменяющиеся цели организации, распределение ресурсов и информации в ходе проекта, а также содержат механизм адаптации управления проектом на основании прошедших этапов. Данная модель будет полезна для изучения руководителям проектов различных уровней и может быть применена для любых ИТ-проектов, отвечающих обозначенным вначале требованиям.

Поступила в редакцию 05.12.2009 г.