УДК 004.78; 681.5

В.П. БОЖКО, М.В. ПОТАПОВА

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

МЕТОД КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ ПРОГРАММНЫХ ПРОЕКТОВ В УСЛОВИЯХ НЕОПРЕДЕЛЁННОСТИ

Проведён анализ основных особенностей программных проектов. Проведены основные отличия программных проектов от других типов технических проектов. Выполнен обзор требований, предъявляемых к системам планирования программных проектов. Рассмотрены теоретические подходы к календарному планированию программных проектов. Выполнен анализ существующего программного обеспечения для планирования проектов и построена его классификация. Доказана необходимость применения методов распространения ограничений и математических методов, ориентированных на обработку неточной и неполной информации

Ключевые слова: *метка, неопределенность, ограничения, календарное планирование, управление.*

Введение

Впервые проблемы управления программными проектами появились в 60-х – начале 70-х годов, после ряда неудачных попыток организации разработки сложных программных продуктов. Причины этих провалов коренились в тех подходах, которые использовались в управлении проектами. Применяемая в то время методика, заимствованная из практики управления техническими проектами, оказалась совершенно неэффективной при управлении программными проектами.

Процесс создания профессионального ПО существенно отличается от процессов реализации технических проектов, что приводит к возникновению ряда проблем при управлении программными проектами [2].

Эти проблемы вызваны наличием трех особенностей, которые определяют принципиальное отличие программных проектов от технических: нематериальность кода, как главной части программного продукта; отсутствие стандартных процессов разработки ПО.

1. Программный продукт нематериален. В отличие от технических проектов, где руководитель видит результаты выполнения проекта, программное обеспечение нельзя увидеть или потрогать. Руководитель программного проекта не видит процесс развития разрабатываемого ПО.

2. Не существует стандартных процессов разработки ПО. Для большинства технических систем процессы их создания хорошо изучены, в то время как процессы создания ПО исследуются только несколько последних лет. Поэтому нельзя точно предсказать, на каком этапе процесса разработки ПО могут возникнуть проблемы, угрожающие всему проекту.

3. Непригодность знания и опыта разработки предыдущих проектов. Большие программные проекты, как правило, значительно отличаются от ранее реализованных проектов. Поэтому руководители должны обладать очень большим практическим опытом. Однако постоянные технологические изменения в компьютерной технике обесценивают предыдущий опыт.

Эффективное управление программным проектом напрямую зависит от правильного планирования работ, необходимых для его выполнения. В процессе планирования проекта определяются процессы, этапы и полученные на каждом из них результаты, которые должны привести к выполнению проекта. Процесс планирования начинается с определения проектных ограничений (временные ограничения, возможности наличного персонала, бюджетные ограничения и т.д.). Эти ограничения должны определяться параллельно с оцениванием проектных параметров, таких как структура и размер проекта, а также распределение функций среди будущих исполнителей. Затем определяются этапы разработки и то, какие результаты (документация, прототипы, подсистемы или версии программного продукта) должны быть получены по окончанию этих этапов. Далее начинается циклическая часть планирования. Сначала разрабатывается график работ по выполнению проекта, затем при возникновении расхождений между реальным и плановым ходом работ возможен пересмотр первоначальных оценок параметров проекта. Это, в свою очередь, может привести к изменению графика работ. Если в результате этих изменений нарушаются сроки за-

вершения проекта, должны быть пересмотрены проектные ограничения.

План проекта должен четко показывать ресурсы, необходимые для реализации проекта, разделение работ на этапы и временной график выполнения этих этапов. В процессе составления графика весь массив работ, необходимых для реализации проекта, разбивается на отдельные этапы и оценивается время, требующееся для выполнения каждого этапа. Обычно многие этапы выполняются параллельно. График работ должен предусматривать это и распределять ресурсы между ними оптимальным образом. Нехватка ресурсов для выполнения какого-либо критического этапа – частая причина задержки выполнения всего проекта.

Кроме временных затрат, руководитель также должен рассчитать другие виды ресурсов, необходимых для успешного выполнения каждого этапа. Особый вид ресурсов – это команда разработчиков, привлеченная к выполнению проекта. Другими видами ресурсов могут быть необходимое свободное дисковое пространство на сервере, время использования какого-либо специального оборудования, либо средства на командировочные расходы персонала, работающего над проектом.

Цель настоящей работы – повышение эффективности процессов календарного планирования программных проектов за счет разработки и применения методов распространения ограничений.

Все это требует решения следующих задач:

– построение формальной модели плана программного проекта;

– формулировки задачи планирования программного проекта в виде задачи удовлетворения ограничений;

– разработки рабочего прототипа программного обеспечения для апробации построенных моделей и методов планирования программных проектов.

1. Обзор существующих методов планирования программных проектов

Задачей планирования проекта является расчет возможных сценариев его реализации и нахождение оптимального сочетания в треугольнике «время-цена-качество» в условиях неопределенности, причина которой – изменения, инициированные участниками проекта, и изменения, вызванные реализацией рисков. Определение оптимума программных проектов усложняется тем, что сам результат проекта не определен, или определен рамочно, поэтому вероятность изменений в них очень высока. Соответственно, экономический эффект от реализации проекта определить точно заранее не представляется возможным. В ситуации, когда неопределенными являются три параметра из трех, традиционное планирование проекта теряет всякий смысл, поэтому в программных проектах имеет смысл говорить о некой вероятности достижения неких результатов по срокам, цене и качеству.

Существующие системы управления проектами в основном применяются для учета достигнутых показателей и прогнозирования результатов и наиболее широко используются в традиционных отраслях проектной деятельности – строительстве, инженерии или в оборонной промышленности. Специализированные инструментальные средства для организации коллективной работы над программным проектом, например, Team Foundation Server компании Microsoft (TFS), могут интегрироваться с системами управления проектов общего назначения. Общими недостатками доступных инструментальных средств и известных методов планирования является отсутствие учета неполноты и приблизительности исходных данных, а также отсутствие гибкости анализа, то есть возможности уточнения параметров плана в процессе его выполнения.

Таким образом, необходимость создания специфического метода для планирования программных проектов объясняется существованием:

– неопределенности и многозначности самого понятия «эффективность проекта»,

– колоссального количества ограничений между отдельными элементами плана проекта при сравнительно небольшом количестве известных функциональных зависимостей между ними.

Задача удовлетворения ограничений впервые была сформулирована в 1970-ых годах Хаффманом (Huffman) и Маквортом (Mackworth). В начале 1980-ых годов А.С. Нариньяни был предложен метод недоопределенных вычислений. Основная идея недоопределенности состоит в том, что каждому объекту сопоставляется не одно точное значение, а некоторое подмножество из множества допустимых значений. Принцип недоопределенности трактует состояние частичной определенности как решение задачи, которое возможно на данном уровне знаний о задаче. При переходе задачи от более широкого пространства допустимых значений к более узкому возможна ситуация, когда становятся применимы другие (например, специализированные) методы решения, которые нельзя было применять в исходной постановке.

2. Метод диаграмм предшествования

Метод диаграмм предшествования применяется в методологии критического пути для построения сетевой диаграммы проекта, в которой операции

изображаются в виде квадратов или прямоугольников (называемых «узлами»), а логические взаимосвязи, существующие между ними, – стрелками. На рис. 1.4 показана простая сетевая диаграмма проекта, составленная с помощью метода диаграмм предшествования.

Данный метод также называется «операциями в узлах»; он используется в большинстве пакетов программ управления проектами.

Метод диаграмм предшествования включает четыре типа зависимостей, или логических взаимосвязей:

Финиш-старт. Инициация последующей операции зависит от завершения предшествующей операции.

Финиш-финиш. Завершение последующей операции зависит от завершения предшествующей операции.

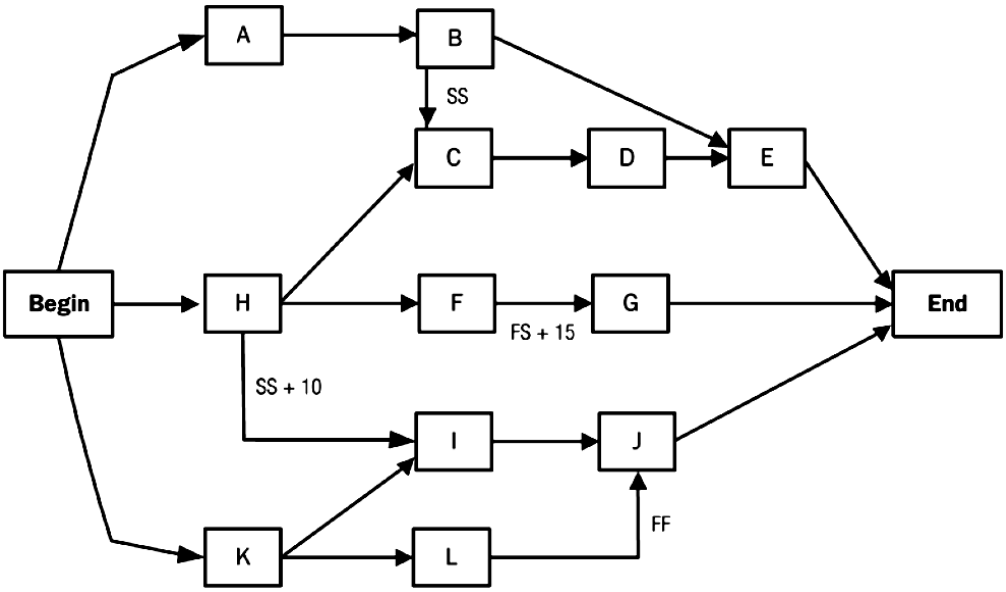

Рис. 1. Пример сетевой диаграммы (метод предшествования)

Старт-старт. Инициация последующей операции зависит от инициации предшествующей операции.

Старт-финиш. Завершение последующей операции зависит от инициации предшествующей операции.

В методе диаграмм предшествования чаще всего используется отношение предшествования типа «финиш-старт». Отношение «старт-финиш» используется редко, но рассматривается здесь для полноты списка типов отношений метода диаграмм предшествования.

3. Метод критического пути

Метод критического пути позволяет рассчитать теоретические даты раннего старта и финиша, а также даты позднего старта и финиша для всех операций без учета ресурсных ограничений путем проведения анализа прохода вперед и назад по сети проекта. Полученные даты раннего старта и финиша не обязательно являются расписанием проекта; они скорее указывают периоды времени, в рамках которых может быть запланированы операции с учетом длительностей операций, логических взаимосвязей, опережений, задержек и других известных ограничений.

На рассчитанные ранние и поздние даты старта и финиша может влиять общий временной резерв операции, который позволяет делать расписание гибким и может быть положительным, отрицательным или нулевым. Для любого пути в сети гибкость расписания, называемая «полным временным резервом», измеряется положительной разницей между ранними и поздними датами. У критических путей полный временной резерв либо нулевой, либо отрицательный, а запланированные операции на критическом пути называются «критическими операциями». Критический путь обычно характеризуется нулевым полным временным резервом. В сетях может существовать несколько путей, близких к критическому. Для создания путей в сети с нулевым или положительным полным временным резервом может потребоваться адаптация длительностей операции, логических взаимосвязей, опережений, задержек и других временных ограничений. После подсчета полного временного резерва пути в сети также может быть определен свободный временной резерв – период времени, на который операция может быть отложена, не вызывая задержки раннего

старта любой непосредственно последующей операции в данном сетевом пути.

4. Метод критической цепи

Критическая цепь представляет собой метод анализа сети, который изменяет расписание проекта с учетом ограниченности ресурсов. Изначально сетевая диаграмма проекта строится на основе оценок длительности, заданных зависимостей и ограничений. Затем рассчитывается критический путь. После определения критического пути учитывается наличие ресурсов и в результате определяется расписание с учетом ресурсных ограничений. Полученное расписание часто имеет измененный критический путь.

Критический путь с ресурсными ограничениями известен как «критическая цепь». Метод критической цепи добавляет буферы длительности в виде операций, не предусматривающих выполнения работ, для управления неопределенностью. Один из буферов, расположенный в конце критической цепи, известен как проектный буфер и защищает директивную дату завершения от задержек на критической цепи. Дополнительные буферы, известные как «питающие буферы», располагаются в каждой точке, в которой в критическую цепь входят цепи взаимосвязанных операций извне критической цепи. Питающие буферы, таким образом, защищают критическую цепь от отставания по входящим цепям. Размер каждого буфера должен учитывать неопределенность длительности цепи зависимых операций, ведущих к данному буферу. Как только буферные операции расписания определены, операции расписания планируются на максимально поздние плановые даты старта и финиша. Таким образом, вместо управления полным временным резервом сетевых путей метод критической цепи концентрируется на управлении оставшимися длительностями буферов, сопоставляя их с оставшейся длительностью цепей операций.

5. Анализ «что-если»

Это анализ вопроса: «Что произойдет, если ситуация будет развиваться по сценарию ‘X’?» В этом случае выполняется анализ сети, при котором с помощью модели расписания просчитываются различные сценарии (например, задержка поставки основных элементов, увеличение длительности отдельных инженерных операций) или моделируется влияние непредвиденных внешних факторов (например, забастовка или изменение процедуры лицензирования). Результаты анализа «что если» могут использоваться для оценки выполнимости расписания проекта при неблагоприятных условиях и для составления резервных планов и планов реагирования для преодоления или смягчения последствий неожиданных ситуаций. Моделирование включает в себя расчет различных длительностей проекта при использовании различных допущений о длительностях операций. Наиболее известен метод Монте-Карло, в котором распределение вероятных значений длительности операции определяется для каждой операции и используется для вычисления распределения вероятных выходов всего проекта.

6. Сжатие расписания

Сжатие расписания сокращает длительность проекта без изменения содержания проекта, временных ограничений, директивных дат или иных целевых параметров расписания. Методы сжатия расписания включают в себя:

Сжатие. Метод сжатия расписания, в котором анализируются компромиссы между стоимостью и расписанием, чтобы определить, каким образом возможно максимально сжать сроки при минимальных затратах. Примеры сжатия могут включать одобрение сверхурочной работы, использование дополнительных ресурсов или плату за ускорение поставки для операций на критическом пути. Сжатие эффективно только для тех операций, где дополнительные ресурсы способны сократить длительность. Сжатие не всегда создает жизнеспособную альтернативу и может привести к увеличению рисков и/или стоимости.

Быстрый проход. При этом методе сжатия расписания фазы или операции, обычно выполняемые последовательно, выполняются параллельно. Примером является строительство фундамента здания до подготовки всех архитектурных чертежей. Быстрый проход может привести к доработкам и увеличению риска. Быстрый проход применим только в том случае, когда операции могут накладываться одна на другую для сокращения длительности.

7. Метод планирования программного проекта на основе программирования в ограничениях

Способ представления недоопределенного значения влияет как на качество полученных результатов, так и на вид ограничений, связывающих это значение. В зависимости от характера представляемой информации недоопределенные значения могут быть представлены в виде целочисленных и вещественных интервалов, множеств, перечислений и других, более специальных, конструкций [1].

Областью определения переменной называется множество всех возможных значений, которые могут быть присвоены этой переменной. В дальнейшем мы будем говорить просто "область переменной" и обозначать эту область переменной x через Dx.

Меткой называется пара переменная-значение, которая определеяет присваивание некоторого значения переменной. Метка, определяющая присваивание значения v переменной x, обозначается $< x, v >$. Метка $< x, v >$ осмысленна только в том случае, когда $v \in Dx$.

Составное меткой называется одновременное присваивание значений некоторому множеству переменных. Составная метка, присваивающая значения $v_1,...,v_n$ соответственно переменным $x_1,...,x_n$ обозначается $(< x_1, v_1 >,...,< x_n, v_n >)$. Здесь также полагается, что $v_i \in Dx_i$, для $i = 1,...,n$.

Ограничением на некоторое множество переменных S называется набор составных меток для всех переменных множества S. Будем обозначать ограничение Cs.

Определение 1. Задача удовлетворения ограничений – это тройка $P = (X, D, C)$, обозначаемая $CSP(P)$, где

X - конечное множество переменных $\{x_1...x_k\}$,

D – функция, отображающая каждую переменную из X на множество объектов произвольного типа: $D : X$ –> {конечное множество объектов некоторого типа}. Будем рассматривать Dx_i как множество объектов, отображенных из x_i функцией D. Эти объекты называются значениями переменной x_i, а множество Dx_i – областью x_i,

C – конечное (возможно пустое) множество ограничений на произвольном подмножестве переменных из X, то есть C – это множество наборов составных меток.

Каждое ограничение из C имеет вид одной из следующих формул:

$$y = x \quad (1)$$

$$y = c \quad (2)$$

$$V = f\{x_i,...,x_n\} \quad (3)$$

где x, x_i, y – символы переменных из V,

c – константный символ,

f – функциональный символ арности n.

Более сложные ограничения распадаются на множество более простых (вышеприведенных видов) после введения дополнительных переменных [2].

Определение 2. Решением задачи удовлетворения ограничений (V, C) называется такое приписывание каждой переменной из V некоторого определенного значения из ее универсума, при котором выполняются все ограничения из C.

В общем случае все точные решения задачи, если таковые существуют, должны лежать в декартовом произведении таких н-значений.

Определение 3. Недоопределенным расширением (н-расширением) универсума X называется любой конечный набор его подмножеств $*X$, содержащий $\varnothing$, и являющийся замкнутым относительно пересечения.

Эти свойства гарантируют однозначное представление $*[\xi]$ любого подмножества $\xi \subseteq X$ в н-расширении $*X$, а именно: $*[\xi] = \bigcap_{\xi \subseteq \varsigma \in *X} \varsigma$.

Таким образом, представлением множества ξ в системе $*X$ является минимальное подмножество из н-расширения $*X$, содержащее ς.

Определение 4. Для вычисления Н-модели обобщенная вычислительная модель (н-модель) M состоит из четырех множеств:

$$M = (V, C, W, CORR) \quad (4)$$

где V – множество н-объектов v из заданной предметной области,

C – множество ограничений на н-объектах из V,

W – множество функций присваивания,

$CORR$ – множество функций проверки корректности.

С каждым объектом из V связаны недоопределенный тип данных (недоопределенное расширение некоторого универсума), начальное значение, функция присваивания и функция проверки корректности.

Функция присваивания – это двухместная функция, работающая при каждой попытке присваивания очередного значения н-объекту и определяющая его новое значение как пересечение текущего и присваиваемого значений.

Функция проверки корректности – это унарный предикат, который проверяет непустоту значения н-объекта.

Алгоритм вычислений, реализованный в н-моделях, является высокопараллельным процессом, который управляется потоком данных. Изменение значений переменных, располагающихся в общей памяти, автоматически влечет интерпретацию тех ограничений, для которых эти переменные являются аргументами. Процесс останавливается, когда сеть стабилизируется или хотя бы один из н-объектов

становится некорректным. В последнем случае устанавливается противоречивость исходной н-модели.

При решении задачи с использованием недоопределенных моделей необходимо определить множество н-объектов из заданной предметной области и множество ограничений на н-объектах на этих объектах.

Каждая задача ($t_{i,j,k}$) является структурным расширением интервального н-объекта, включая: трудоемкость выполнения, время начала, время окончания. Отношения непосредственного предшествования между задачами, которые характеризуют частичную упорядоченность задач, задаваемое в виде неравенств входит в общую систему C – множество ограничений на н-объектах из V. В начале вычислений для всех задач времена начала и окончания совпадают с началом и окончанием всего проекта, в результате вычислений происходит уточнение этих сроков.

Модель компетенций определена в декартовом пространстве

$$E \times T \to F_{experience}$$

и представляется целыми константами для каждого сочетания сотрудник-работа, характеризуя имеющийся опыт участников проекта.

Модель ролевых запретов определена на пространстве

$$T \times T \to F_{ban}$$

в множестве констант логического типа.

Искомое решение задачи о назначениях ищется в декартовом произведении пространств сотрудники-задачи-время

$$E \times T \times \tau \to Works$$

как множество вещественных переменных, характеризующих относительную занятость сотрудника на исполнение определенной роли в определенный момент времени, при выполнении неравенств, входящих в общее множество ограничений:

– роль должна соответствовать модели компетенции и возможности исполнителя с определенным уровнем подчиненности ее исполнять;

– для каждого момента времени сумма занятостей сотрудника во всех исполняемых ролях не должна превышать 1.

Таким образом, исходная постановка оптимизационной задачи при использовании недоопределенных моделей заменена на поиск и определение такого множества вариантов работ, при которых все частные показатели эффективности проекта удовлетворяют соответствующим ограничениям.

Помимо перечисленных, необходимо учитывать дополнительные общепроектные ограничения, к которым относятся: ограничение на фонд заработной платы, на общую трудоемкость работ, на средний уровень занятости исполнителей в проекте, на степень участия каждого исполнителя в конкретных работах проекта и т.д.

Метод вычислений, реализующий вычисления на н-моделях, является высокопараллельным процессом, который управляется потоком данных. Изменение значений переменных, располагающихся в общей памяти, автоматически влечет интерпретацию тех ограничений, для которых эти переменные являются аргументами. Процесс останавливается, когда сеть стабилизируется или хотя бы один из н-объектов становится некорректным. В последнем случае устанавливается противоречивость исходной н-модели.

8. Алгоритмизация прототипа программного обеспечения для планирования программных проектов

Для рассмотрения алгоритмов распространения ограничений необходимо ввести ряд определений, которые используются в данных алгоритмах.

Определение 3.1. Областью определения переменной называется множество всех допустимых значений, которые могут быть присвоены данной переменной. Если x – это переменная, то через Dx обозначается область ее определения.

Определение 3.2. Меткой называется пара переменная-значение, которая представляет присваивание значения переменной. Для обозначения метки используется запись $< x, v >$ (что означает присваивания значения v переменной x). Метка $< x, v >$ имеет смысл только тогда, когда v принадлежит области определения переменной x (т.е. $x \in D_x$).

Определение 3.3.Составная метка – это одновременное присваивание значений множеству (возможно пустому) переменных. Запись $(< x_1, v_1 >, < x_2, v_2 >, \ldots, < x_n, v_n >))$ обозначает составную метку, которая присваивает значения $v_1, v_2, ..., v_n$ переменным $x_1, x_2, ..., x_n$.

Определение 3.4. k-составной меткой называется составная метка, которая k переменным присваивает k значений.

Определение 3.5. Ограничением на некотором множестве S переменных называется множество составных меток для переменных из множества S. Для удобства используется запись CS, обозначающая ограничение на переменные из множества S.

Определение 3.6. Задачей удовлетворения ограничений (ЗУО) называется тройка

$$(Z, D, C), \qquad (5)$$

где Z – конечное множество переменных $\{x_1, x_2, ..., x_n\}$;

D – функция, которая отображает каждую переменную из множества Z на множество объектов произвольного типа:

$D: Z \rightarrow$ конечное множество объектов (заданного типа);

C – конечное (возможно пустое) множество ограничений на произвольном подмножестве переменных из Z.

Определение 3.7. Гиперграфом ограничений ЗУО (Z, D, C) называется гиперграф, в котором каждая вершина соответствует переменной из Z, а каждое (гипер-)ребро соответствует ограничению из C.

Определение 3.8. ЗУО является совместной в вершинах тогда и только тогда, когда для каждой переменной все значения в области определения этой переменной удовлетворяют всем ограничениям над данной переменной:

$$\forall csp((Z,D,C)):$$

$$\text{node-consistent}((Z,D,C,)) \equiv$$

$$(\forall z \in Z: (\forall v \in D_x : satisfies(<x,v>, C_x))).$$

Определение 3.9. Путь $(x_0, x_1, ..., x_m)$ в графе ограничений для ЗУО называется совместным по пути тогда и только тогда, когда для каждой 2-составной метки $(< x_0, v_o >, < x_m, v_m >)$, которая удовлетворяет всем ограничениям на x_0 и x_m, существует метка для каждой переменной из $x_1, ..., x_{m-1}$ такая, что каждое бинарное ограничение на смежных переменных в этом пути удовлетворяется.

$$\forall csp((Z,D,C)):$$

$$\forall x_0, x_1, \ldots, x_m \in Z: PC((x_0, x_1, \ldots, x_m), (Z,D,C)) \equiv$$

$$(\forall v_0 \in Dx_0, v_m \in Dx_m:$$

$$(satisfies((<x_0, v_0>, <x_m, v_m>), Cx_0, x_m) \Rightarrow$$

$$(\exists v_1 \in Dx_1, v_2 \in Dx_2, \ldots, v_{(m-1)} \in Dx_{(m-1)}:$$

$$satisfies((<x_1, v_1>), Cx_1) \wedge \ldots \wedge$$

$$satisfies((<x_{m-1}, v_{m-1}>), Cx_{m-1}) \wedge$$

$$satisfies((<x_o, v_0>, <x_1, v_1>), Cx_{1,} v_1) \wedge \ldots \wedge$$

$$satisfies((<x_1, v_1>, <x_2, v_2>), Cx_{1,} v_1) \wedge \ldots \wedge$$

$$satisfies((<x_{m-1}, v_{m-1}>, <x_m, v_m>), Cx_{m-1,} x_m).$$

Определение 3.10. ЗУО называется совместной по путям тогда и только тогда, когда каждый путь в графе этой ЗУО является совместным по пути.

Это предполагает, что если ЗУО является совместной по путям, то для всех переменных x и y, всякий раз, когда составная метка $(< x, a >, < y, b >)$ удовлетворяет ограничениям на x и y, существует метка $(< z, c >)$ для каждой переменной z такая, что $(< x, a >, < y, b >, < z, x >)$ удовлетворяет всем ограничениям на x, y и z.

Определение 3.11. Графом ограничений задачи ЗУО (Z,D,C) называется неориентированный граф, в котором каждая вершина соответствует переменной из Z, а ребра между вершинами заданы для тех пар переменных их Z, которые являются частью k-ограничения из C.

$$\forall graph((V,E)):$$

$$(V,E)=G((Z,D,C)) \equiv ((V=Z) \wedge E=$$

$$\{(x,y) | x,y \in Z \wedge (\exists C_s \in C: x,y \in S))).$$

Выводы

В статье представлена формализованная модель плана программного проекта, которая содержит 2 основных блока:

1. Модель объектов управления – процессов создания продукта и управления проектом;

2. Модель субъектов управления – проектной команды, которая предназначена для учета ограничений, характеризующих: опыт решения задач различного типа, запрет на одновременное выполнение двух определенных ролей одним исполнителем, а также фактическую возможность выполнения конкретной задачи определенным исполнителем описание команды проекта.

Литература

1. Телерман, В.В. Удовлетворение ограничений в задачах математического программирования [Текст] / В.В. Телерман, Д.М. Ушаков // Вычислительные технологии. – 1998. – Т. 3, N 2. – С. 45 – 54.

2. Телерман, В.В .Недоопределенные модели: формализация подхода и перспективы развити. [Текст] / В. В. Телерман, Д. М. Ушаков // Пробл. преставления и обработки не полностью определенных знаний. — М.-Н-ск, 1996, — С. 7 – 32.

Поступила в редакцию 11.05.2012

Рецензент: д-р техн. наук, проф., зав. каф. И.В. Чумаченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

МЕТОД КАЛЕНДАРНОГО ПЛАНУВАННЯ ПРОГРАМНИХ ПРОЕКТІВ В УМОВАХ НЕВИЗНАЧЕНОСТІ

В.П. Божко, М.В.Потапова

Проведено аналіз основних особливостей програмних проектів. Проведено основні відмінності програмних проектів від інших типів технічних проектів. Виконано огляд вимог, пропонованих до систем планування програмних проектів. Розглянуто теоретичні підходи до календарного планування програмних проектів. Виконано аналіз існуючого програмного забезпечення для планування проектів і побудована його класифікація. Доведено необхідність застосування методів поширення обмежень і математичних методів, орієнтованих на обробку неточної й неповної інформації.

Ключові слова: мітка, невизначеність, обмеження, календарне планування, керування.

METHOD OF THE CALENDAR PLANNING SOFTWARE PROJECTS UNDER UNCERTAINTYO

V.P. Bozhko, M.V. Potapova

The analysis of the main features of the software projects. Carried out the main differences between software projects from other types of technical projects. A review of the requirements which are made by the systems of planning of software projects. The theoretical approaches to the calendar Nome planning software projects. The analysis of existing software for project planning, and built his classification. The necessity of application of methods of distribution restrictions, and mathematical methods, focused on the processing of inaccurate and incomplete information.

Key words: label, uncertainties, limitations, scheduling, management.

Божко Валерий Павлович – д-р техн. наук, профессор, заведующий кафедрой финансов Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

Потапова Марина Викторовна – зав. лаб. кафедры финансов Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: marina-k604@yandex.ru.