## УПРАВЛІННЯ ПРОЕКТАМИ

УДК 004.052.42

**И.Б. ТУРКИН, М.С. СОКОЛОВ, Д.С. ДЕРЕЖЕНЕЦ**
**Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ"**

# Планирование Программных проектов на основе недоопределенных моделей и программирования в ограничениях

*Проведен анализ проблем планирования проектов, специфичных для программной инженерии. Дана формальная постановка задачи планирования программного проекта в виде задачи удовлетворения ограничений, обосновано использование недоопределенных моделей и программирования в ограничениях для решения задач планирования. Показана практическая апробация построенных моделей и методов на разработанном прототипе инструментального средства для планирования программных проектов.*

*Виконано аналіз проблем планування проектів, специфічних для програмної інженерії. Дано формальну постановку задачі планування програмного проекту у вигляді задачі задоволення обмежень, обґрунтовано використання недовизначених моделей та програмування в обмеженнях для розв'язку задач планування. Показана практична апробація побудованих моделей та методів на розробленому прототипі інструментального засобу для планування програмних проектів.*

*The analysis of the software projects planning is conducted. The formal raising of task of planning of software project is given as a task of satisfaction of limitations, the use of underdefined models and constrained programming is grounded for the decision of planning tasks. Practical approbation of the built models and methods is reviewed on the developed tool prototype for software projects planning.*

**Ключевые слова:** управление проектами, программное обеспечение, программный проект, планирование проекта, программирование в ограничениях, недоопределенная модель, удовлетворение ограничений

**Введение**

Впервые проблемы управления программными проектами появились в 60-х – начале 70-х годов после провала многих проектов по разработке программных продуктов. Причины этих провалов коренились в подходах, которые использовались в управлении проектами.

Эти методы были основаны на опыте управления техническими проектами и оказались совершенно неэффективными при управлении программными проектами.

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.

**Анализ исследований и публикаций**

Процесс создания профессионального программного обеспечения (ПО) существенно отличается от процессов реализации технических проектов, что приводит к определенным трудностям в управлении программными проектами.

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.

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.

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 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 завершения проекта, то 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 [1].

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, необходимые 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.

В наиболее общей постановке, задача оптимизации проектного плана включает в себя формирование множеств исполнителей, а затем поиск такого их назначения на проектные роли в конкретных задачах, который обеспечивает максимизацию интегрального показателя эффективности проекта.

Существует несколько промышленных методологий разработки программных продуктов, например, широко известна технология Rational Unified Process (RUP) [2], имеющая инструментальную поддержку в виде различных программных систем (Rational Rose) и предлагающая весьма сильно формализованный подход к процессу разработки.

Microsoft Solutions Framework (MSF) версии 4.0 была представлена в 2005 году и представляет собой эволюционное развитие предыдущей версии методологии. Важное нововведение состоит в том, что в MSF 4.0 произошло разделение методологии на два направления: MSF for Agile Software Development и MSF for CMMI Process Improvement.

MSF for CMMI Process Improvement – это строгий, документированный процесс, рассчитанный на большие команды и длительный процесс разработки, что предполагает больше верификации, больше планирования, процедуры утверждения, отслеживание потраченных ресурсов и т.д.

Модель Microsoft Solutions Framework for Agile Software Development [3] объединяет ряд наиболее эффективных принципов других известных моделей процессов, сформировав при этом единую базу для работы над проекта-

ми любых типов: ориентированных на фазы (phase-based), основанных на вехах/контрольных точках (milestone-driven) и итеративных (iterative). MSF for Agile Software Development поддерживает быструю итеративную разработку.

MSF for Agile Software Development в определенной степени отражает тенденции последнего времени, связанные с появлением методологий, предлагающих максимально облегченный и гибкий подход к процессу разработки. Одним из примеров подобных методологий является Extreme Programming (XP).

Agile направление в MSF ориентируется на небольшие команды (5-6 человек), предполагает, что информация о разрабатываемом продукте не просто выясняется в процессе разработки, а может и будет изменяться по ходу. Таким образом, первая рабочая версия системы должна быть создана как можно раньше, а сам продукт фактически проявляется из прототипов путем повторения итераций в цикле разработки.

Существующие системы управления проектами в основном применяются для учета достигнутых показателей и прогнозирования результатов. Наиболее широко используются в традиционных отраслях проектной деятельности – строительстве, инженерии или в оборонной промышленности, а 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.

**Постановка задачи**

Необходимость создания специфического инструментария 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.

Цель статьи - сформулировать задачу планирования программного проекта в виде задачи удовлетворения ограничений; показать апробацию построенных моделей и методов планирования программных проектов на разработаном прототипе ПО. Таким образом, за счет разработки и применения инструментальной среды, в которой реализованы математические методы, ориентированные на обработку неточной и неполной информации повысится эффективность процессов планирования программных проектов.

**Результаты исследования**

**Основные понятия недоопределенных вычислений**

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 [4-7].

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. Спецификация задачи в условиях недоопределенности называется недоопределенной моделью (н-моделью).

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 [8]. К достоинствам н-моделей относится то, что:

– вычисления на всех н-моделях выполняются единым потоковым алгоритмом, который не зависит от класса исходной задачи. Это позволяет использовать в одной и той же н-модели такие объекты и ограничения, которые традиционно относятся к задачам разных типов, например целочисленные, вещественные и логические объекты, множества и массивы, линейные, нелинейные и теоретико-множественные отношения и т.д.;

– на основе аппарата н-моделей созданы высокотехнологичные программные продукты, позволяющие относительно просто настраиваться на самые разнообразные предметные области [9-11].

***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,

y = c,

$V = f\{x_i, ...,x_n\}$,

где x, xi, y - символы переменных из V,

c - константный символ,

f - функциональный символ арности n.

Более сложные ограничения после введения дополнительных переменных [12] распадаются на множество более простых (вышеприведенных видов).

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

M=(V, C, W, CORR), 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.

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. Классификация алгоритмов решения задачи удовлетворения ограничений приведена на рис. 1.


Рис. 1. Классификация алгоритмов решения задачи удовлетворения ограничений

Модель плана программного проекта

Модель плана программного проекта содержит два основных блока:

– модель объектов управления – процессов создания продукта и управления проектом;

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

На рис. 2 представлена структура системной модели. В этой модели с каждой задачей связаны ее время начала, время окончания, трудоемкость и тип работы. Следовательно, задачу можно представить в виде четверки:

$$Task = \langle time\_begin, time\_end, effort, type \rangle .$$

Здесь элементы *time_begin* и *time_end* представляют время начала задачи и время ее окончания соответственно.

Элемент *type* – это элемент множества $WorkType = \{work\_type_i, i = 1..n\}$, которое содержит все возможные типы работ на проекте (например, тестирование, написание документации, проектирование, кодирование и т.д.).

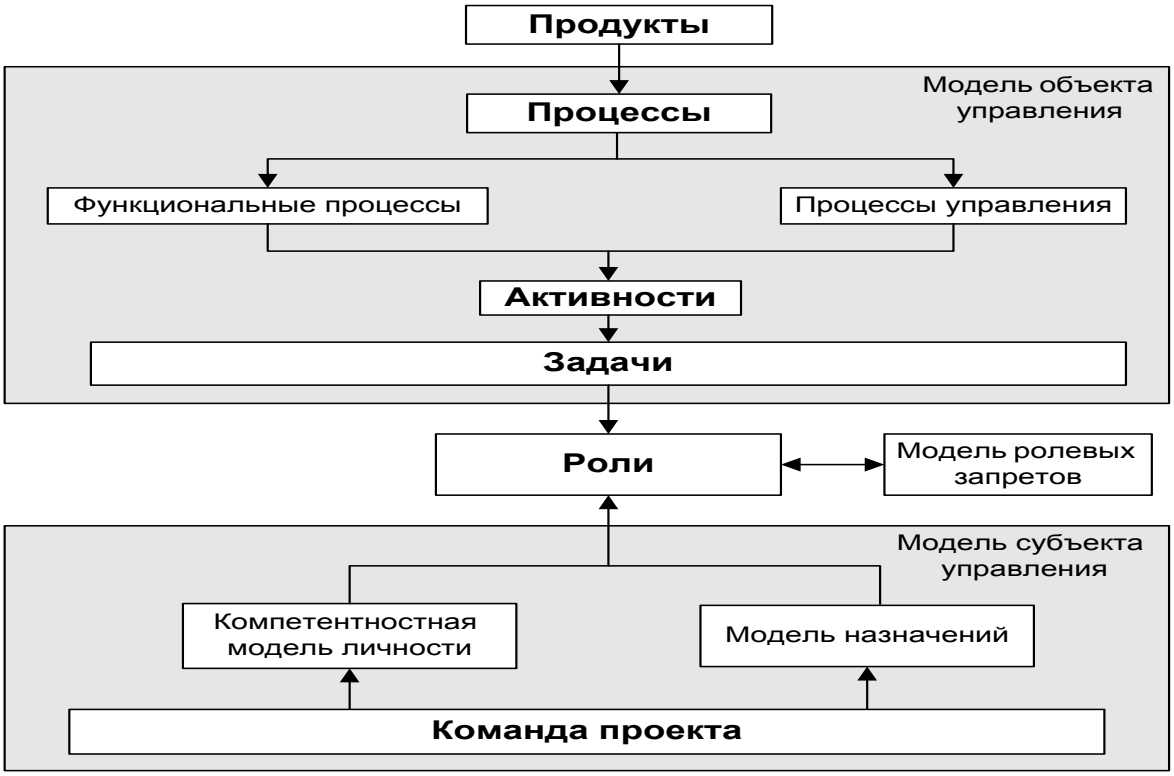


Рис. 2. Структура системной модели

Будем полагать, что трудоемкость *effort* и допустимое число исполнителей n для задачи t определяется экспертным путем и представляется в виде интервальной оценки. Тогда для определения продолжительности работ должно быть решено следующее интервальное уравнение:

$$effort(t) = n(t) * c(t),$$

где $n(t) = [\underline{n}_t, \overline{n}_t]$ – интервальное число, характеризующее допустимое количество исполнителей задачи;

$c(t) = [\underline{c}_t, \overline{c}_t]$ – интервальное число, характеризующее время, необходимое для выполнения задачи.

Задачи могут быть связаны между собой по времени. Связь по времени отображает логическую зависимость между работами в реальном мире. Наиболее частой причиной таких зависимостей являются технологические ограничения (начало одних работ зависит от результатов других), хотя возможны и ограничения, диктуемые другими соображениями.

В соответствии с установленными связями, работы делятся на предшествующие и последующие. Предшествующая работа является обеспечивающей для последующей работы, т.е. для начала выполнения последующей работы требуется выполнение всех предшествующих.

Модель структуры проекта представим в виде ориентированного ациклического графа:

$$\Gamma_{project} = \langle T, R \rangle ,$$

где T – множество задач проекта;

R – множество бинарных отношений, характеризующих связи по времени между задачами.

Элементы множества R представляют собой тройки вида:

$TaskDependency = \langle task_{prev}, task_{post}, dependency \rangle$,

где $task_{prev}$ представляет собой предшествующую задачу;

$task_{post}$ – последующую задачу;

*dependency* – это ограничение, характеризующее временную зависимость между задачами.

Данное ограничение представляет собой выражение вида:

$time\_begin(task_{prev}) \geq time\_end(task_{post}) + \Delta t$,

либо

$time\_end(task_{prev}) \geq time\_begin(task_{post}) + \Delta t$

Модель проектной команды 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.

Данная модель представляет собой граф $\Gamma_{team} = \langle E, T, B, P \rangle$, который состоит из трех подграфов.

1) Подграф опыта участников, который для каждого исполнителя определяет его способности выполнять указанный тип работы, представляет собой двудольный взвешенный граф:

$$\Gamma_{experience} = \langle E, WorkType, F_{experience} \rangle,$$

где $E$ – множество участников проекта;

*WorkType* – множество всех типов работ на проекте;

$F_{experience} \subseteq E \times WorkType$ – множество ребер, характеризующих для каждого участника имеющийся опыт, измеряемый количеством предыдущих исполнений конкретного типа работы $wt \in WorkType$ с конкретным исполнителем $exc \in E$.

2) Подграф ролевых запретов, который определяет взаимное отображение ролей и характеризует запрет на одновременное исполнение двух определенный ролей любым участником проекта:

$$\Gamma_{ban} = \langle WorkType, B \rangle,$$

где $B \subseteq \{\langle wt_1, wt_2 \rangle \mid wt_1, wt_2 \in WotkType, wt_1 \neq wt_2\}$ – множество ребер, характеризующих запрет на выполнение двух ролей.

3) Подграф назначений, отображающий фактическую возможность выполнения конкретной задачи определенным исполнителем, представляет собой двудольный граф:

$\Gamma_{actual} = \langle E, T, F_{actual} \rangle$, где $F_{actual} \subseteq E \times T$ – множество ребер, характеризующих для каждого участника фактическую возможность исполнения им конкретной задачи.

Таким образом, модель проектной команды уточняет классическую постановку «задачи о назначениях» – поиска на двудольном графе паросочетаний, удовлетворяющих требованиям некоторых критериев за счет учета ограничений, присущих данной задаче.

Решением данной задачи для характерного отрезка времени реализации программного проекта $\tau_i$, то есть допустимым парасочетанием <участники проекта> - <исполняемые роли>, является граф:

$\Gamma_{match}^{\tau_i} = \langle E, T^{\tau_i}, F_{match}^{\tau_i} \rangle$, при ограничениях:

$$\forall \tau_i, \forall t_i \in T^{\tau_i}, F_{match} \subseteq F_{actual},$$

$$\forall \tau_i, \forall t_i \in T^{\tau_i}, F_{match} \subseteq F_{experience},$$

$$\forall \tau_i, \forall t_j, t_k \in T^{\tau_i}, (t_j, t_k) \in F_{ban} \Rightarrow F_{match}^{\tau_i}(t_j) \cap F_{match}^{\tau_i}(t_k) = \varnothing$$

**Общепроектные ограничения**

Перечислим дополнительные общепроектные ограничения, которые, как правило, должны выполняться в любом проекте:

1. Ограничение на фонд заработной платы:

$$\sum_i \left( effort(t_i) \sum_j \left( \gamma_{t_i, exc_j} \cdot Z_{exc_j} \right) \right) \leq Z_\Sigma,$$

где $effort(t_i)$ – трудоемкость выполнения работы $t_i$ исполнителями;

$\gamma_{t_i, exc_j}$ – коэффициент, характеризующий относительный объем работы $t_i$, выполненной *j*-м исполнителем $exc_j$, такой, что:

$$\forall exc_j \in E, \sum_i \gamma_{t_i, exc_j} = 1;$$

$Z_{exc_j}$ – заработная плата *j*-го исполнителя;

$Z_\Sigma$ – предельная величина фонда заработной платы

2. Ограничение на общую трудоемкость работ

$$\sum_i effort(t_i) \leq T_\Sigma,$$

где $T_\Sigma$ – предельно допустимая общая трудоемкость работ.

3. Ограничения на средний уровень занятости исполнителей в проекте предполагают, что могут быть назначены нижняя $\underline{\beta}$ и верхняя $\overline{\beta}$ границы, характеризующие среднюю занятость исполнителей:

$$\underline{\beta} \le \frac{\sum_{i} \text{effort}(t_i)}{\|E\| \cdot \tau_{T_\Sigma}} \le \overline{\beta}.$$

где $\tau_{T_\Sigma}$ – общая продолжительность работ;

4. Ограничение на степень участия *j*-го исполнителя в конкретных работах проекта устанавливает предельно допустимые значения относительной занятости любого человека в проектных работах :

$$\forall j, \underline{\beta_j} \le \sum_{i} \frac{\text{effort}(t_i, \text{exc}_j)}{\tau_{T_\Sigma}} \le \overline{\beta_j},$$

5. Ограничение на временное участие *j*-го исполнителя в проекте предполагает, что может быть задан некоторый период времени $t = (t_s .. t_f)$, в течение которого исполнитель будет задействован в проекте:

$$\forall i, \forall j, \begin{cases} t_s \le \min_i(\tau_{start_{i,j}}), \\ t_f \ge \max_i(\tau_{finish_{i,j}}) \end{cases},$$

где $\tau_{start_{i,j}}$, $\tau_{finish_{i,j}}$ – время начала, окончания работ, в которых планируется участие *j*-го исполнителя.

6. Ограничение на максимальную продолжительность интервала времени $\tau_{free_{don}}$, в течение которого простаивание *j*-го исполнителя (отсутствие загрузки проектными работами) считается допустимым:

$$\forall j, \Delta_{\tau_j} = \max_r\left(\overline{\tau_{free_{r,j}}} - \underline{\tau_{free_{r,j}}}\right) \le \tau_{free_{\text{äîï}}},$$

где $\overline{\tau_{free_{r,j}}}$, $\underline{\tau_{free_{r,j}}}$ – время окончания, начала r-того интервала времени, когда *j*-го исполнитель не принимает участия в проектных работах, то есть

$$\forall \tau_j, \overline{\tau_{free_{r,j}}} \le \tau_j \prec \underline{\tau_{free_{r,j}}},$$

$$\neg \exists i, \tau_{start_{i,j}} \le \tau_j \prec \tau_{finish_{i,j}}$$

7. Ограничение на планирование работ, предотвращающее появление дефектов плана работ, типа «бутылочное горло», когда существует работа, начало выполнения которой, задерживается в течение значительного промежутка времени из-за неготовности незначительной доли ее входных продуктов:

$$\exists T_i \in T, \exists \Delta\tau_j \ge \Delta\tau_{\text{äîï}}, \frac{\left\|WP_{in_{i,\Delta\tau_j}}\right\|}{\left\|WP_{in_i}\right\|} \le \sigma_{i,\Delta\tau_j},$$

где $T_i$ – работа;

$\Delta\tau_{\text{äîï}}$ – максимально допустимый интервал времени ожидания недостающих для начала $T_i$ работы входных продуктов $WP_{in_{i,\Delta\tau_j}}$;

$\sigma_{i,\Delta\tau_j}$ - минимально допустимая доля входных продуктов, которые необходимы для начала $T_i$ работы.

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 $\text{Ò} \times T \to F_{ban}$ 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.

**Практическая апробация**

Применение новой усовершенствованной модели плана программного продукта позволит контролировать сроки разработки проекта,

определять эффективность работы команды и оптимизировать планирование программного проекта в целом.

Разработанные модели реализованы в прототипе программного обеспечения, позволяющем решать задачи планирования на основе недоопределенных моделей. Интерфейс программы (рис. 3) состоит из двух вкладок: «план проекта» и «команда», панели инстументов, меню. На рисунке 3 представлена вкладка «план проекта», которая состоит из двух панелей: список задач (слева) и диаграмма Ганта (справа). Задачи в проект добавляются с помощью команд меню.

Вкладка «команда» имеет структуру, подобную вкладке «план проекта». На левой панели отображается список исполнителей, а справа на календаре отображается их загруженность.

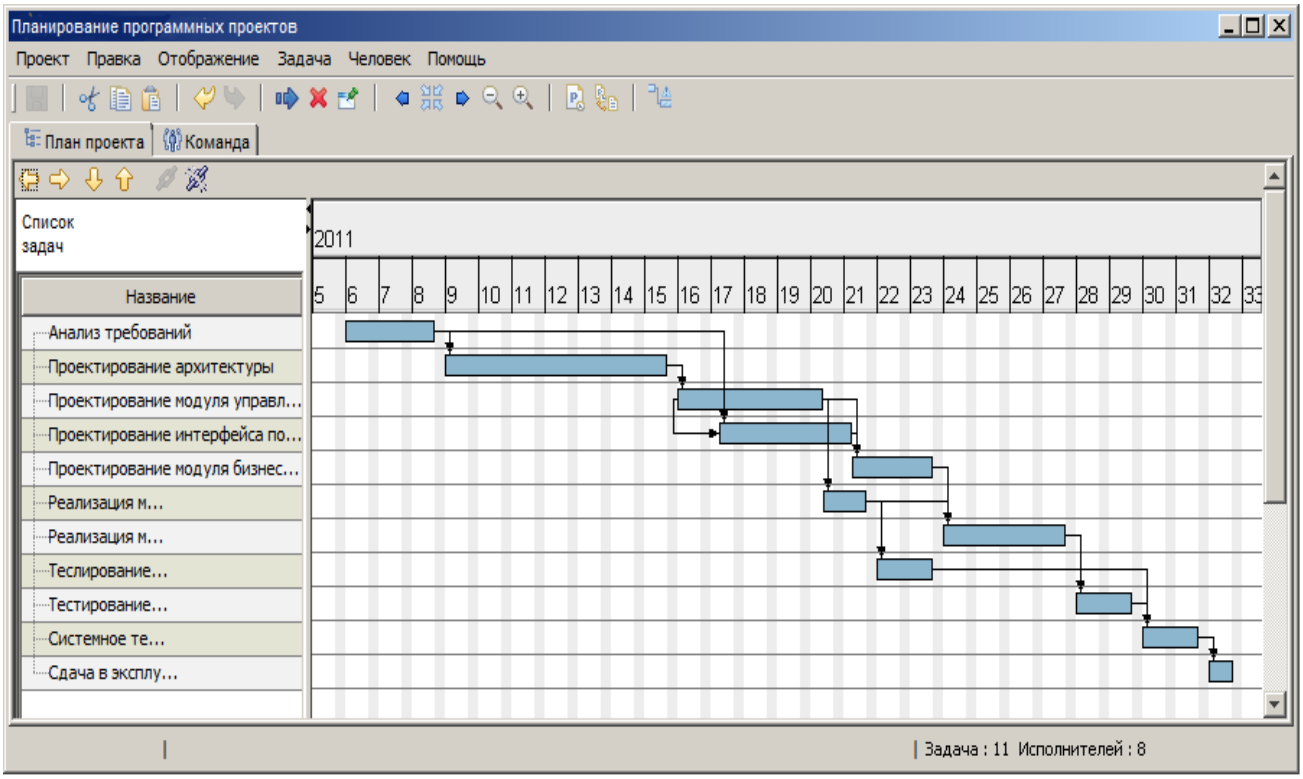


Рис. 3. Главное окно прототипа ПО

В качестве способа представления плана проекта выбран XML-формат. XML-файл с описанием плана программного проекта содержит корневой элемент с именем schedule, который состоит из четырех элементов:

– projectDates – содержит даты начала и окончания проекта;

– taskList – содержит список задач проекта;

– taskDependencies – содержит зависимости между задачами;

– executorList – содержит список исполнителей.

**Выводы**

При использовании недоопределенных моделей исходная постановка оптимизационной задачи заменена на поиск и определение такого множества вариантов работ, при которых все частные показатели эффективности проекта удовлетворяют соответствующим ограничениям. При этом возможность оптимизации плана сохраняется, но сама оптимизация осуществляется как пошаговый интерактивный процесс, в котором производится уточнение интервалов показателей в направлении, улучшающем значения используемых критериев оценки проекта. Уменьшение запасов делает требования плана намного более жесткими, а его выполнение более проблематичным.

Перспективой настоящей работы является дальнейшее развитие математических моделей планирования программных проектов, а также разработка программного продукта, реализующего планирование с учетом существующих международных стандартов жизненного цикла разработки программного обеспечения, который позволит существенно повысить эф-

фективность планирования и управления программных проектов.

**Литература**

1. Соммервилл И., Инженерия программного обеспечения. / И. Соммервилл, пер. с англ – М. : Издательский дом «Вильямс», 2002. – 624с.

2. Rational Unified Process. Best Practices for Software Development Teams. A Rational Software Corporation White Paper. [электронный ресурс] // – режим доступа: http://www.augustana.ab.ca/~mohrj/courses/2000.winter/csc220/papers/rup_best_practices/rup_bestpractices.pdf.

3. MSF for Agile Software Development Process Guidance. [электронный ресурс] // – режим доступа: http://go.microsoft.com/ fwlink/ ?linkid=63524].

4. Нариньяни А. С. Недоопределенность в системах представления и обработки знаний. / А. С. Нариньяни // Техническая кибернетика. / Известия АН СССР. – 1986. – Вып. 5. - С. 3-28.

5. Интеллектуальная технология недоопределенного календарно-ресурсного планирования и управления проектами Time-EX / А. С. Нариньяни, И. Д. Гофман, А. А. Липатов, Д. А. Инишев // Информационные технологии. - 2010. - N 2. - С. 2-32.

6. Напреенко В.Г. Методика оптимального планирования и управления комплексными инновационными разработками для условий затрудненного прогнозирования хода работ // Альманах «Наука. Инновации. Образование» - 2008. – Вып. 4. –С. 374-383.

7. Моделирование ресурсных ограничений при структурном неопределенном планировании в технологии TIME-EX / И.Д. Гофман, Д.А. Инишев, А.В. Шурбаков, Л.Г. Романов // Труды VI-й международной конференции "Проблемы управления и моделирования в сложных системах" - Самара: Самарский Научный Центр РАН - 2004. -С. 176-182.

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

9. Петров Е.С Модуль для решения линейных ограничений в системе UniCalc / Е.С. Петров, Ю.В. Костов, Е.Ю. Ботоева // «Проблемы управления и моделирования в сложных системах»: труды 8-й междунар. конф. – Самара: Самарский научный центр РАН, 2006. - С. 270-277.

10. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG / И. Братко – М. : Издательский дом «Вильямс», 2004. - 640 с.

11. ISO/IEC 13211-1:1995. Information Technology. Programming Languages. Prolog. Part 1: General Core. [электронный ресурс] // – режим доступа: http://www.iso.org /iso / iso_catalogue/ catalogue_tc/catalogue_detail.htm?csnumber=21413

12. Телерман В. В. Недоопределенные модели: формализация подхода и перспективы развития. / В. В. Телерман, Д. М. Ушаков // Проблемы преставления и обработки не полностью определенных знаний, знаний. - РосНИИ ИИ. - М.-Новосибирск, 1996. С. 7-32.

**Відомості про авторів**:

**Туркин Игорь Борисович** – д.т.н., професор, зав каф. инженерии программного обеспечения Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина.
e-mail: energy@d4.khai.edu.

**Дереженец Денис Сергеевич –** студент Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина.
e-mail: 9deniz@gmail.com.

**Соколов Никита Сергеевич –** студент Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина.
e-mail: 9sokolov@gmail.com.

Статья поступила в редакцію 15.06.2011 г.