МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

СОГЛАСОВАНО
Генеральный директор
ЗАО «АйТи»

_____ Бакиев О.Р.
"29" декабря 2011 г.

УТВЕРЖДАЮ
Ректор НИУ ИТМО

_____ Васильев В.Н.
"29" декабря 2011 г.

МНОГОПРОФИЛЬНАЯ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА СОЗДАНИЯ И УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СРЕДОЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ CLAVIRE

ПРОГРАММНЫЙ КОМПОНЕНТ ПЛАНИРОВАНИЯ ИСПОЛНЕНИЯ WF CLAVIRE/SCHEDULER

ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.80066-06 13 28-ЛУ

Представители
Организации-разработчика

Руководитель разработки,
профессор НИУ ИТМО

_____ Бухановский А.В.
"29" декабря 2011 г.

Ответственный исполнитель,
с.н.с. НИУ ИТМО

_____ Луценко А.Е.
"29" декабря 2011 г.

Нормоконтролер
инженер НИУ ИТМО

_____ Позднякова Л.Г.
"29" декабря 2011 г.

2011

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

УТВЕРЖДЕН
RU.СНАБ.80066-06 13 28-ЛУ

МНОГОПРОФИЛЬНАЯ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА СОЗДАНИЯ И УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СРЕДОЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ CLAVIRE

ПРОГРАММНЫЙ КОМПОНЕНТ ПЛАНИРОВАНИЯ ИСПОЛНЕНИЯ WF CLAVIRE/SCHEDULER

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 28

ЛИСТОВ 17

2011

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

АННОТАЦИЯ

Документ содержит описание программного компонента планирования исполнения WF CLAVIRE/Scheduler RU.СНАБ.80066-06 01 28, обеспечивающего поддержку процесса выполнения цепочек заданий в автоматическом режиме с подбором оптимального плана исполнения. Программный компонент разработан в ходе выполнения проекта «Создание распределенной вычислительной среды на базе облачной архитектуры для построения и эксплуатации высокопроизводительных композитных приложений» (Договор № 21057 от 15 июля 2010 г., шифр 2010-218-01-209) в рамках реализации постановления Правительства РФ № 218 «О мерах государственной поддержки развития кооперации российских высших учебных заведений и организаций, реализующих комплексные проекты по созданию высокотехнологичного производства».

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Компонент планирования исполнения цепочек заданий (WF) МИТП CLAVIRE/Scheduler RU.СНАБ.80066-06 01 28, обеспечивающий поддержку процесса выполнения WF в автоматическом режиме с подбором оптимального плана выполнения, предназначен для генерации планов исполнения workflow на заданном множестве вычислительных ресурсов в соответствии с экстренной или неэкстренной эвристикой распределения задач по ресурсам.

Компонент функционирует на аппаратных системах с архитектурой процессора x86, x86_64 и IA64. Для его работы необходимо следующее системное программное обеспечение: ОС семейства Windows NT (версии старше Windows 2000), .NET 4.0, Microsoft Internet Information Services (с версией старше 6.0), либо ОС Linux (с ядром 2.6.22 и выше) с установленной средой Mono Framework с поддержкой библиотек .NET 2.0 и выше (рекомендуется версия Mono Framework 2.6 или выше), а также web-сервером с поддержкой ASP .NET Web Services.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Основное функциональное назначение данного компонента заключается в координации работы других элементов МИТП (таких как, например, подсистема управления запуском задач на вычислительных ресурсах CLAVIRE/Execution) путем предоставления по запросу построенных по некоторой эвристике планов выполнения задач на вычислительных ресурсах. Отличие экстренных эвристик от неэкстренных состоит в том, что целью работы экстренной эвристики является получение плана, время выполнения которого будет наиболее близким к желаемому (используется при необходимости выполнения экстренных вычислений).

Функциональность компонента включает в себя следующие возможности:

1) планирование WF на заданном множестве вычислительных ресурсов в соответствии с некоторой эвристикой и с использованием оценок времени выполнения задач на различных ресурсах, построенных на основе базы моделей производительности прикладных пакетов;
2) планирование WF повышенного приоритета, состоящего из экстренных задач (urgent computing), – предусматривается возможность преждевременного завершения уже

запущенных низкоприоритетных задач, если они занимают ресурсы, необходимые для выполнения экстренных задач (наибольшей актуальностью обладает при реализации МИТП-Э). Этот процесс планирования также осуществляется в соответствии с некоторой эвристикой и с использованием оценок времени выполнения задач на различных ресурсах, построенных на основе базы моделей производительности прикладных пакетов.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Структура и общие схемы работы компонента

Компонент планирования WF состоит из единственного модуля, осуществляющего собственно генерацию плана выполнения workflow на основе оценок времени счета прикладных пакетов на вычислительных ресурсах. Этот модуль представляет собой изолированную библиотеку .NET, которую при необходимости легко можно интегрировать в любые приложения. В качестве внешнего интерфейса доступа к этой библиотеке и для связи ее с прочими компонентами комплекса была реализована соответствующая WCF-служба.

Таким образом, компонент Scheduler выполняет важную роль в интерпретации абстрактного workflow (AWF) и превращении его в конкретный (CWF). Более детально структура компонента изображена на рис. 3.1.

Собственно процедура планирования workflow такова: при поступлении запроса на планирование необходимо в первую очередь определить, является планируемый WF экстренным или обычным.

При построении плана для обычного workflow порядок выполнения отдельных его блоков определяется зависимостями между задачами и заданной эвристикой использования вычислительных ресурсов (реализованные эвристики будут рассмотрены ниже). Входными данными эвристик служат оценки времени выполнения всех задач workflow на всех ресурсах системы (из них в соответствии с правилами выбранной эвристики выбирается оптимальная по некоторому критерию), а выходными – информация о том, какую задачу и на каком ресурсе следует запустить. Результатом планирования в этом случае является список задач с указанием того, какая задача и на каком ресурсе будет запущена, а также примерная оценка занятости вычислительных ресурсов, которая строится на основе полученных данных планирования задач текущего WF и данных о занятости ресурсов на момент начала планирования.

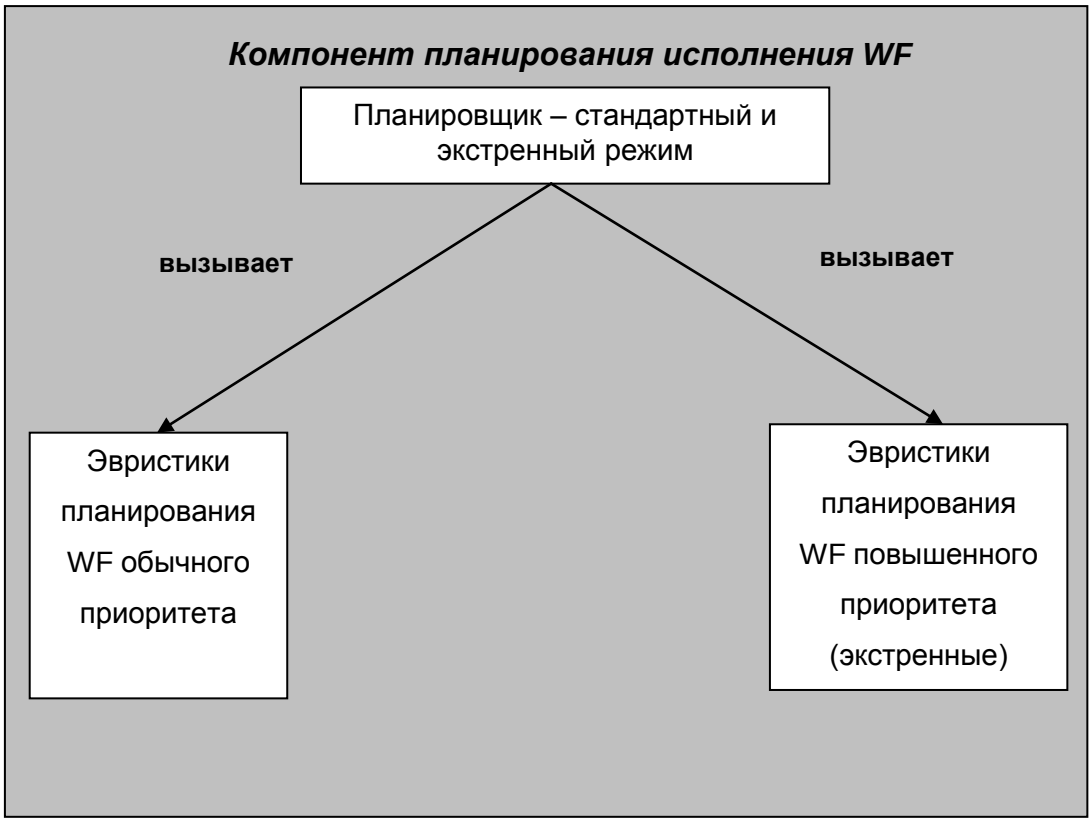

Рисунок 3.1 – Структура компонента планирования исполнения WF

Задачи экстренного workflow получают повышенный приоритет по сравнению с обычными задачами, что уже запущены на ресурсах и конкурируют лишь между собой, а также уже запущенными экстренными задачами, если такие имеются. Вместе с экстренным workflow на вход алгоритма поступают желаемые временные рамки его выполнения (минимальное и максимальное время), и на основе этих данных в соответствии с эвристикой экстренного планирования подбирается такая конфигурация запуска задач на доступных ресурсах, что примерное время выполнения WF будет либо лежать в заданных границах, либо, если такой вариант подобрать не удается, будет выбран самый быстрый план выполнения. В работе экстренных эвристик, как и в случае с WF обычного приоритета, используются оценки времени выполнения элементов workflow на доступных вычислительных ресурсах. Результатом планирования в этом случае являются те же данные, что и в случае с неэкстренным WF, а также, возможно, список низкоприоритетных запущенных задач, которые необходимо остановить для освобождения ресурсов в пользу задач планируемого экстренного workflow.

Результирующим временем выполнения WF в обоих случаях является время окончания выполнения его последней (согласно зависимостям) задачи, которое

складывается из времени ожидания запуска задачи (зависит от времени освобождения необходимых задаче вычислительных ресурсов и времени, необходимого на разрешение всех зависимостей этой задачи) и собственно времени ее выполнения.

3.2. Эвристики неэкстренного планирования

Для построения плана запуска неэкстренного WF, как было сказано выше, используются эвристики планирования использования ресурсов, конкретно – алгоритмы Max-Min, Min-Min и Sufferage. Псевдокод этих алгоритмов приведен на рис. 3.2–3.4. В описаниях алгоритмов используются следующие сокращения:

- EET(t, r) – Estimated Execution Time, количество времени, требуемого непосредственно для расчета задачи t на ресурсе r;
- EAT(r) – Estimated Availability Time, время, к которому ресурс r освободится для расчета;
- DRT(t, r) – Dependencies Resolution Time, время, к которому будут выполнены все задачи, от которых зависит задача t, а все файлы, необходимые для ее счета, будут скопированы на ресурс r;
- ECT(t, r) – Estimated Completion Time, время, к которому задача t посчитается на ресурсе r.

```
 текущие_задачи := незапланированные задачи, чьи родители посчитались;
Пока существуют текущие_задачи
     Для каждой задачи t из текущих_задач
          Для каждого ресурса r из доступных для задачи t
               ECT(t, r) := EET(t, r) + max(EAT(r), DRT(t, r));
          Конец_для
          лучший_ресурс_для_задачи(t) := ресурс r с мин-ным ECT(t, r) для задачи t;
     Конец_для
     выбранная_задача := задача t с минимальным ECT(t, лучший_ресурс_для_задачи(t));
     выбранный_ресурс := лучший_ресурс_для_задачи(выбранная_задача);
     определить выбранную_задачу на выбранный_ресурс;
     убрать выбранную_задачу из текущих_задач;
     обновить EAT(выбранный_ресурс);
Конец_пока
```

Рисунок 3.2 – Алгоритм Min-min

Алгоритм Max-min (рис. 3.3) схож с алгоритмом Min-min, но в отличие от последнего, отдает приоритет задаче, которая будет считаться максимальное, а не минимальное, время.

```
текущие_задачи := незапланированные задачи, чьи родители посчитались;
Пока существуют текущие_задачи
     Для каждой задачи t из текущих_задач
          Для каждого ресурса r из доступных для задачи t
               ECT(t, r) := EET(t, r) + max(EAT(r), DRT(t, r));
          Конец_для
          лучший_ресурс_для_задачи(t) := ресурс r с мин-ным ECT(t, r) для задачи t;
     Конец_для
     выбранная_задача := задача t с максимальным ECT(t, лучший_ресурс_для_задачи(t));
     выбранный_ресурс := лучший_ресурс_для_задачи(выбранная_задача);
     определить выбранную_задачу на выбранный_ресурс;
     убрать выбранную_задачу из текущих_задач;
     обновить EAT(выбранный_ресурс);
Конец_пока
```

Рисунок 3.3 – Псевдокод алгоритма Max-min

Алгоритм Sufferage (рис. 3.4) не отдает предпочтения задачам с минимальным или максимальным ECT на лучшем для задачи ресурсе. Вместо этого выбирается задача, у которой максимальна разница между временем счета на лучшем ресурсе и следующим за ним значением.

```
текущие_задачи := незапланированные задачи, чьи родители посчитались;
Пока существуют текущие_задачи
     Для каждой готовой задачи t из текущих_задач
          Для каждого ресурса r из доступных для задачи t
               ECT(t, r) := EET(t, r) + max(EAT(r), DRT(t, r));
          Конец_для
          лучший_ресурс(t) := ресурс r с минимальным ECT(t, r) для задачи t;
          второй_ресурс(t) := ресурс r с мин-ным ECT(t, r) после лучший_ресурс(t);
     Конец_для
выбранная_задача := задача t с максимальной разницей ECT(t, лучший_ресурс(t)) и ECT(t,
второй_ресурс(t));
выбранный_ресурс := лучший_ресурс(выбранная_задача);
определить выбранную_задачу на выбранный_ресурс;
убрать выбранную_задачу из текущих_задач;
обновить EAT(выбранный_ресурс);
Конец_пока
```

Рисунок 3.4– Псевдокод алгоритма Sufferage

3.3. Эвристики экстренного планирования

Для построения плана запуска экстренного WF, как было сказано выше, используются экстренные эвристики планирования использования ресурсов, конкретно – алгоритмы Greedy, Best First, Best Free First, Busiest First и Busiest Free First.

Алгоритм Greedy схож по смыслу с алгоритмом Min-Min, описанным в разделе 3.2, – он последовательно выбирает доступные задачи с минимальным временем выполнения, ставя им в соответствие самые быстрые ресурсы, таким образом минимизируя общее время выполнения WF.

Алгоритм Best First схож с алгоритмом Greedy – отличие их состоит в том, что данный алгоритм, получив план с минимальным временем, последовательно «ухудшает» его, перенося задачи на менее производительные ресурсы, чтобы оставить их не

используемыми, стремясь при этом, однако, удержать время выполнения WF внутри заданных рамок.

Алгоритм Best Free First схож с алгоритмом Best First и отличается от него тем, что при расчете EAT(r) для ресурса r он учитывает не только экстренные задачи, выполняющиеся на нем, но и неэкстренные.

Алгоритм Busiest First фактически представляет собой алгоритм Max-Min, описанный в разделе 3.2, с той разницей, что он выбирает для каждой задачи не лучшие, а худшие ресурсы, и, получив план выполнения задач на наименее производительных ресурсах, последовательно «улучшает» его, перенося задачи на более производительные ресурсы, стремясь при этом, однако, удержать время выполнения WF внутри заданных рамок.

Алгоритм Busiest Free First имеет сходство с алгоритмом Best First и отличается от него тем, что при расчете EAT(r) для ресурса r учитывает не только экстренные задачи, выполняющиеся на нем, но и неэкстренные.

3.4. Основные классы компонента планирования WF

Ниже приводятся сокращенные описания структуры и методов основных классов компонента событийного взаимодействия.

3.4.1. Класс SchedulerService

Основной класс WCF-сервиса компонента планирования. Реализует интерфейс IScheduler. Является интерфейсом всего компонента.

Открытые методы

- GetUHNames – получение имен доступных эвристик экстренного планирования
    a) возвращает массив имен доступных эвристик экстренного планирования (тип: String[]).
- GetDefaultUHName – получение имени экстренной эвристики, используемой по умолчанию
    a) возвращает имя активной экстренной эвристики (тип: String).
- SetDefaultUHName – назначение имени экстренной эвристики, используемой по умолчанию
    a) входной параметр name (тип: String) – имя новой активной экстренной эвристики;
    b) возвращаемое значение отсутствует (void).

- GetHNames – получение имен доступных эвристик неэкстренного планирования
    a) возвращает массив имен доступных эвристик неэкстренного планирования (тип: String[]).
- GetDefaultHName – получение имени неэкстренной эвристики, используемой по умолчанию
    a) возвращает имя активной неэкстренной эвристики (тип: String).
- SetDefaultHName – назначение имени неэкстренной эвристики, используемой по умолчанию
    a) входной параметр name (тип: String) – имя новой активной неэкстренной эвристики;
    b) возвращаемое значение отсутствует (void).
- RescheduleEstimated – спланировать WF
    a) входной параметр workflow (тип: EstimatedWorkflow) – WF, для которого необходимо получить план выполнения;
    b) возвращает план выполнения (тип: LaunchPlan).

3.4.2. Класс EstimatedWorkflow

Планируемый WF. Содержит данные о задачах, выполнение которых необходимо спланировать, оценках времени их выполнения на различных ресурсах и их зависимостях между собой, а также задачах других WF, активных в данный момент.

Свойства

- IsUrgent (тип: bool) – флаг, показывающий, является ли данный WF экстренным. Свойство возвращает true лишь у экстренных WF.

Поля

- Tasks (тип: List<EstimatedTask>) – список задач WF, выполнение которых необходимо спланировать.
- ActiveTasks (тип: List<ActiveEstimatedTask>) – список уже выполняющихся экстренных и неэкстренных задач.
- Dependencies (тип: List<TasksDependency>) – список зависимостей между задачами WF.
- Name (тип: string) – идентификатор WF.

Открытые методы

- UpdateDependencies – обновить зависимости всех задач в соответствии со списком Dependencies
    a) Возвращаемое значение отсутствует (void).

Закрытые методы

- DFS – провести топологическую сортировку списка задач
    a) входной параметр tasks (тип: IEnumerable<EstimatedTask>) – список задач, находящихся на одном уровне «глубины» графа зависимостей между задачами;
    b) входной параметр level (тип: int) – текущий уровень «глубины» графа зависимостей между задачами.

3.4.3. Класс EstimatedUrgentWorkflow

Потомок класса EstimatedWorkflow (см. раздел 3.4.2).

Поля

- MinExecutionTime (тип: double) – минимальное время выполнения WF.
- MaxExecutionTime (тип: double) – максимальное время выполнения WF.

3.4.4. Класс TasksDependency

Пара идентификаторов «предок-потомок», инкапсулирующая информацию о зависимости между двумя задачами WF.

Поля

- ProviderId (тип: ulong) – идентификатор задачи-предка.
- ConsumerId (тип: ulong) – идентификатор задачи-потомка.

3.4.5. Класс EstimatedTask

Задача WF, выполнение которой необходимо спланировать, и оценки времени ее выполнения на ресурсах.

Свойства

- IsScheduled (тип: bool) – флаг, показывающий, спланировано ли выполнение данной задачи.
- IsReady (тип: bool) – флаг, показывающий, готова ли данная задача к выполнению.

Поля

- Estimations (тип: ResourceEstimation[]) – список оценок времени выполнения этой задачи на различных ресурсах.
- RequiresDependencies (тип: List<EstimatedTask>) – список задач, от которых зависит данная.
- ProvidesDependencies (тип: List<EstimatedTask>) – список задач, от которых зависит данная.
- DepthLevel (тип: int) – «глубина» нахождения этой задачи на графе зависимостей.
- ScheduledInstance (тип: ActiveEstimatedTask) – объект спланированной для выполнения на конкретном ресурсе задачи, соответствующий данному.

3.4.6. Класс ActiveEstimatedTask

Задача WF, уже выполняющаяся или спланированная для выполнения на определенном ресурсе.

Поля

- TaskState (тип: State) – текущее состояние данной задачи (варианты: ACTIVE запущена, SCHEDULED – запланирована к выполнению, ABORTED – экстренно завершена).
- Estimation (тип: ActiveEstimation) – описание ресурса и узлов, где исполняется задача, плюс данные об ожидаемом времени завершения расчета и ожидаемом времени запуска, если задача еще не запущена.
- IsUrgent (тип: bool) – флаг, показывающий, является ли эта задача частью экстренного WF.

3.4.7. Класс ResourceEstimation

Оценка времени выполнения задачи на ресурсе.

Поля

- Resource (тип: Resource) – описание ресурса (имя ресурса, параметры узлов).
- Result (тип: EstimationResult) – собственно оценка (время и параметры выполнения).

3.4.8. Класс LaunchPlan

План запуска WF на ресурсе.

Поля

- Plan (тип: List<ActiveEstimatedTask>) – список запущенных, отложенных и экстренно завершенных задач.
- NodesTimings (тип: List<NodeAvailabilityTime>) – список объектов с данными о времени доступности используемых вычислительных узлов с учетом времени, необходимого на выполнение запланированного WF.
- EstimatedExecutionTime (тип: double) – примерное общее время выполнения WF (в секундах).

3.4.9. Интерфейс ITaskBasedHeuristics

Интерфейс класса неэкстренной эвристики.

Методы

- ChooseTask – выбор очередной задачи для запуска
    a) входной параметр workflow (тип: EstimatedWorkflow) – планируемый WF;
    b) Estimations (тип: List<IEnumerable<ActiveEstimatedTask>>) – набор оценок задач WF на конкретных наборах узлов вычислительных ресурсов;
    c) возвращает готовую к запуску задачу (тип: ActiveEstimatedTask).

3.4.10. Интерфейс IWFBasedUrgentHeuristics

Интерфейс класса неэкстренной эвристики.

Методы

- MakeInitialPlan – создать базовый план запуска
    a) входной параметр workflow (тип: EstimatedUrgentWorkflow) – планируемый WF;
    b) возвращает план выполнения (тип: UrgentPlan).
- OptimizePlan – оптимизировать полученный ранее план
    a) входной параметр workflow (тип: EstimatedUrgentWorkflow) – планируемый WF;
    b) входной параметр previousPlan (тип: UrgentPlan) – полученный ранее план;
    c) возвращает флаг, показывающий, удалось ли оптимизировать переданный план (тип: bool).

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Компонент планирования исполнения WF предназначен для функционирования на аппаратных системах с характеристиками:

- архитектура процессора – x86, x86_64, IA64;
- минимальный объем оперативной памяти – 1 ГБ;
- минимальный объем свободного пространства на жестком диске – 1 ГБ;
- минимальная тактовая частота процессора – 1 ГГц.

Компонент представляет собой набор программных библиотек, предназначенных для исполнения в среде Microsoft .NET 4.0 и выше, и WCF-сервис, разработанные на языке C# 4.0.

Компонент может функционировать:

1) на вычислительной системе под управлением ОС Linux (с ядром 2.6.22 и выше) с установленной средой Mono Framework с поддержкой библиотек .NET 2.0 и выше (рекомендуется версия Mono Framework 2.6 или выше). Для корректного функционирования необходимо наличие установленного web-сервера с поддержкой технологии ASP .NET WebServices (рекомендуется использование web-сервера XSP, поставляющегося в составе среды Mono Framework);
2) на вычислительной системе под управлением ОС Windows (версии XP и выше) с установленной средой .NET Framework версии 3.5 или выше. Для корректного функционирования необходимо наличие установленного web-сервера с поддержкой технологии ASP .NET WebServices

5. ВЫЗОВ И ЗАГРУЗКА

Компонент планирования исполнения WF реализован в виде SOAP WCF-сервиса платформы .NET. Для запуска сервиса используется стандартный механизм используемого web-сервера. Загрузка сервиса в этом случае выполняется web-сервером автоматически по мере поступления запросов от клиентов сервиса.

Компонент планирования исполнения WF предоставляет интерфейс в виде WCF-сервиса, реализованного классом SchedulerService (раздел 3.4.1). Вызов сервиса производится стандартным для технологии WCF способом: по опубликованному описанию сервиса (WSDL) необходимо создать прокси-класс, через который осуществляется взаимодействие с сервисом путем вызова необходимых операций

(методов). Процедура создания прокси-класса зависит от того, на базе каких технологий строится клиентское приложение. Если выбраны язык программирования C# и платформа .NET, построение клиента производится за счет вызова служебного программного средства svcutil.exe, распространяемого в составе платформы .NET, либо за счет создания ссылки на сервис в среде Microsoft VisualStudio.

6. ВХОДНЫЕ ДАННЫЕ

Входными данными для компонента планирования исполнения WF являются:

1) конфигурационные файлы, описывающие параметры запуска WCF-сервиса. В частности, в конфигурационном файле указываются эвристики, используемые при планировании по умолчанию. При старте системы конфигурация считывается и активируется;
2) параметры планирования:
   a. описание WF, подлежащего планированию в экстренном или неэкстренном режиме – указывается в качестве параметра метода RescheduleEstimated;
   b. эвристика, выбранная для использования по умолчанию для планирования WF в экстренном и неэкстренном режимах.

7. ВЫХОДНЫЕ ДАННЫЕ

Выходными данными компонента планирования исполнения WF является план выполнения WF в форме экземпляров класса LaunchPlan (см. раздел 3.4.8), возвращаемого методом RescheduleEstimated.

Кроме того, сервис может предоставлять информацию о доступных и используемых по умолчанию эвристиках планирования WF в различных режимах (см. методы GetHNames, GetUHNames, GetDefaultHName, GetDefaultUHName).

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

SOAP	Simple Object Access Protocol (простой протокол доступа к объектам)
URI	Uniform Resource Identifier (унифицированный идентификатор ресурса)
WCF	Windows Communication Foundation
XML	eXtensible Markup Language (расширяемый язык разметки)
БД	База данных
МИТП	Многопрофильная инструментально-технологическая платформа
ОС	Операционная система
ПО	Программное обеспечение

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Изм.	**Номера листов (страниц)**				**Всего листов (страниц) в докум.**	**№ документа**	**Входящий № сопроводительного докум. и дата**	**Подп.**	**Дата**
	измененных	**замененных**	**новых**	**аннулированных**					