ЗАО «АЙТИ»

УТВЕРЖДАЮ

Генеральный директор
ЗАО «АйТи».

_____ Бакиев О.Р.

"__" _____ 2011 г.

Создание высокотехнологичного производства комплексных решений в области предметно-ориентированных облачных вычислений для нужд науки, промышленности, бизнеса и социальной сферы

ПРОГРАММНЫЙ КОМПОНЕНТ МОДЕЛИРОВАНИЯ ИСПОЛНЕНИЯ WF CLAVIRE/SIM

ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.80066-06 13 65-ЛУ

2011

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

ЗАО «АЙТИ»

УТВЕРЖДЕН
RU.СНАБ.80066-06 13 65-ЛУ

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

ПРОГРАММНЫЙ КОМПОНЕНТ МОДЕЛИРОВАНИЯ ИСПОЛНЕНИЯ WF CLAVIRE/SIM

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 65

ЛИСТОВ 19

2011

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

АННОТАЦИЯ

Документ содержит описание программного компонента моделирования исполнения WF CLAVIRE/Sim RU.СНАБ.80066-06 01 65, предназначенного для проведения экспериментов, моделирующих выполнение WF и задач в рамках системы CLAVIRE. Программный компонент разработан в рамках проекта «Создание распределенной вычислительной среды на базе облачной архитектуры для построения и эксплуатации высокопроизводительных композитных приложений» 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 моделирования исполнения WF CLAVIRE/Sim RU.СНАБ.80066-06 01 65 предназначен для проведения экспериментов, в ходе которых моделируется выполнение WF и задач в среде CLAVIRE – моделирование затрагивает весь жизненный цикл WF и задач с момента поступления в систему скрипта на EasyFlow и до замеров времени выполнения задач с учётом накладных расходов.

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).

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Основное функциональное назначение данного компонента заключается в предоставлении инструментария для описания и проведения экспериментов по моделированию выполнения WF и задач в среде CLAVIRE. Результаты таких экспериментов можно использовать в разных целях, таких как, например:

- Исследование и сравнение различных стратегий планирования выполнения WF;
- Более детальное информирование пользователей CLAVIRE о прогрессе выполнения запущенного WF (путём соотнесения текущего состояния WF с результатом моделирования его выполнения, полученным перед запуском).

Функциональность компонента включает в себя следующие возможности:

- Предоставление набора абстрактных классов и интерфейсов, которые можно использовать для описания экспериментов;
- Предоставление среды выполнения экспериментов (моделирование жизненного цикла WF и задач, работа с сервисами базы ресурсов CLAVIRE и планировщика).

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Структура и общие схемы работы компонента моделирования

Компонент моделирования исполнения workflow состоит из единственного модуля, содержащего (в первом приближении) структуры данных, необходимые для описания экспериментов, а также классы, отвечающие за собственно моделирование жизненного цикла WF и задач. 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 обеспечения возможности более гибкой работы с ней со стороны прочих компонентов CLAVIRE планируется реализовать WCF-службу. Более детально структура компонента и его связей с другими компонентами CLAVIRE изображена на рис. 3.1 – стрелки указывают направления от зависимых компонентов к тем, от которых существует зависимость.

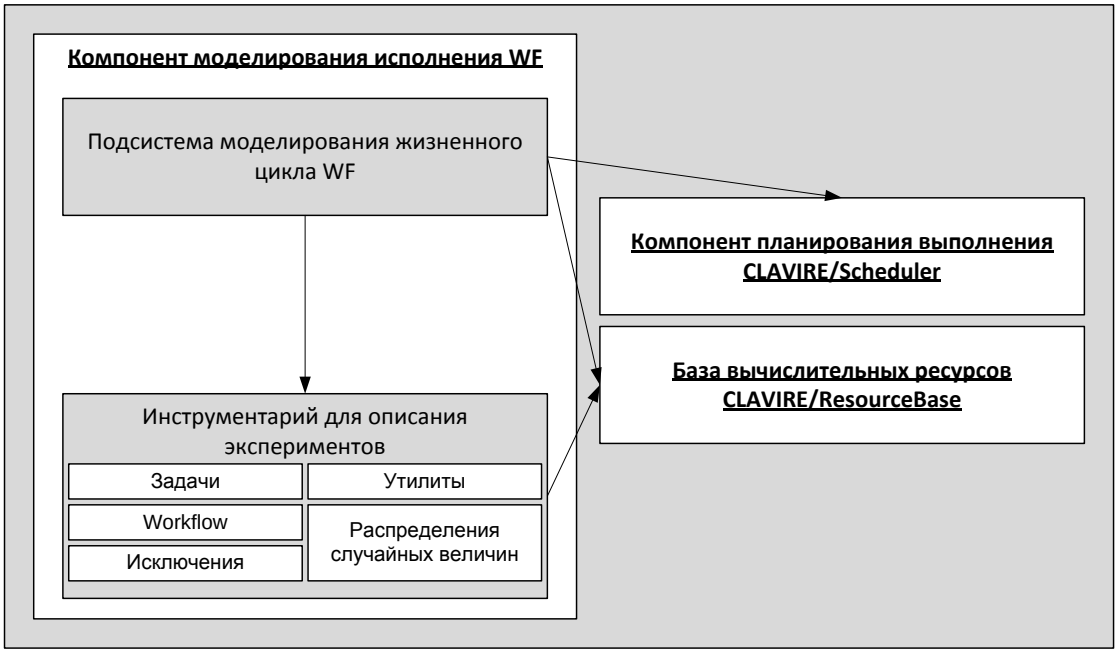

Рисунок 3.1 - Структура компонента моделирования исполнения WF и его связей с другими компонентами CLAVIRE

Работа компонента моделирования исполнения WF строится вокруг ***экспериментов***, и это определяет типовую схему работы с ним. В текущем контексте экспериментом называется процедура, инициализируемая на основе некоторых входных параметров, в рамках которой в соответствии с неким алгоритмом производится моделирование выполнения одного или нескольких WF, чей состав и структура определяются спецификой эксперимента; также в соответствии со смыслом и логикой

данного эксперимента формируется отчёт о его результатах. Также алгоритм эксперимента должен, помимо прочего, описывать условие его завершения.

Как было сказано выше, в ходе эксперимента моделируются запуски WF – теперь стоит обратить внимание на то, каким образом производится моделирование отдельных шагов WF. Работа компонента моделирования в этом аспекте основана на концепции дискретного времени – иными словами, течение времени представляется в виде отрезков равной длины (например, 5 секунд), и на каждом таком отрезке производится обновление состояния системы, выполняющихся в данный момент задач и, соответственно, эксперимента, к которому относятся эти задачи. Величина дискретного отрезка времени должна определяться спецификой эксперимента. Процедура отслеживания состояния задач и изменения их состояния моделирует работу компонента управления выполнением CLAVIRE/Executor (последовательный запуск задач в соответствии с зависимостями между ними, взаимодействие с компонентом планирования CLAVIRE/Scheduler и т.п.).

С учётом вышесказанного, алгоритм выполнения эксперимента можно представить в виде блок-схемы, изображённой на рис. 3.2 – подчёркиванием выделены те элементы алгоритма, которые определяются спецификой конкретного эксперимента.

3.2. Описание экспериментов

Описание эксперимента можно разделить на два этапа:

1. Описание прикладных пакетов, чьи запуски будут моделироваться в ходе эксперимента;
2. Описание собственно алгоритма эксперимента.

Первый этап выполняется путём задания структур данных, инкапсулирующих знания о моделях производительности и накладных расходов на выполнение прикладных пакетов – по одному типу данных на прикладной пакет. С технической точки зрения эти типы данных должны являться потомками абстрактного класса **ActiveTask**, определённого среди прочих в базовом наборе классов для описания экспериментов. У этих новых классов необходимо определить методы **EstimateTime** и **EstimateOverheads**, отвечающие за оценку соответственно времени выполнения данного прикладного пакета на заданном ресурсе и за оценку временны**х** накладных расходов на его выполнение. Также возможно использовать уже имеющиеся описания пакетов, созданные ранее, поскольку один эксперимент может использовать типы данных, определённые другим экспериментом.

Второй этап описания эксперимента осуществляется путём задания класса, реализующего интерфейс **ISimCase**, определённый в базовом наборе классов для

описания экспериментов. Реализация этого интерфейса должна определять следующие методы:

- **Init** – инициализация эксперимента;
- **CreateState** – создание нового объекта состояния системы;
- **ShouldContinue** – проверка эксперимента на завершение;
- **Update** – обновление состояния системы (запуск новых задач и т.п.);
- **WriteReport** – формирование отчёта о выполнении эксперимента.

Методы **Init**, **CreateState** и **WriteReport** принимают аргументы в виде набора пар «ключ-значение», где значением может являться объект произвольного типа.

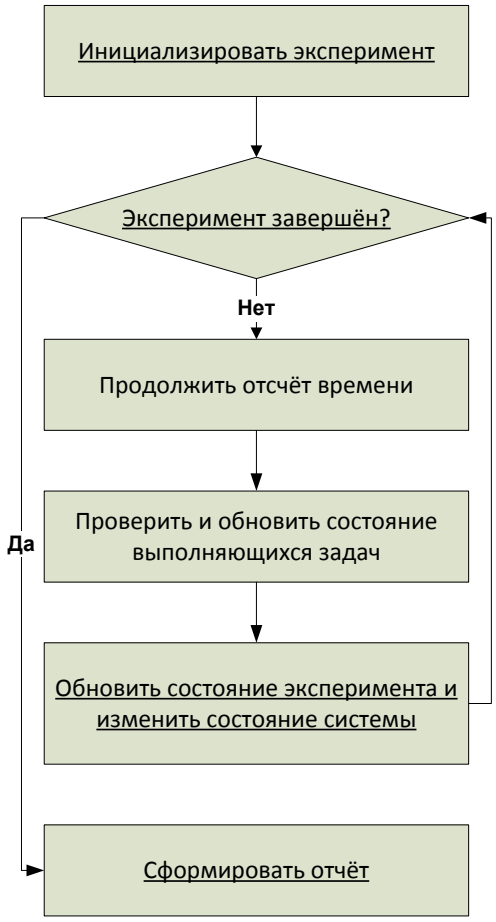

Рисунок 3.2 - Блок-схема алгоритма выполнения эксперимента по моделированию

3.3. Задание характеристик пакетов с помощью распределений случайных величин

Как было указано выше, классы, описывающие модели производительности прикладных пакетов и связанных с их выполнением накладных расходов, должны определять соответствующие методы, алгоритм работы которых является для

пользователей этих классов «чёрным ящиком», то есть метод может возвращать как значение, детерминировано зависящее от входных параметров (то есть производить моделирование в соответствии с жёстко заданной формулой), так и значение некоторой случайной величины, заданной каким-либо распределением. Для задания распределений и их привязки к описаниям конкретных пакетов рекомендуется пользоваться пространством имён **Sim.Random**, содержащим соответствующие абстрактные классы и интерфейсы.

3.4. Основные классы компонента моделирования исполнения WF

Ниже приводятся сокращенные описания структуры и методов основных классов компонента моделирования исполнения WF. Диаграмма их отношений между собой изображена на следующем рисунке.

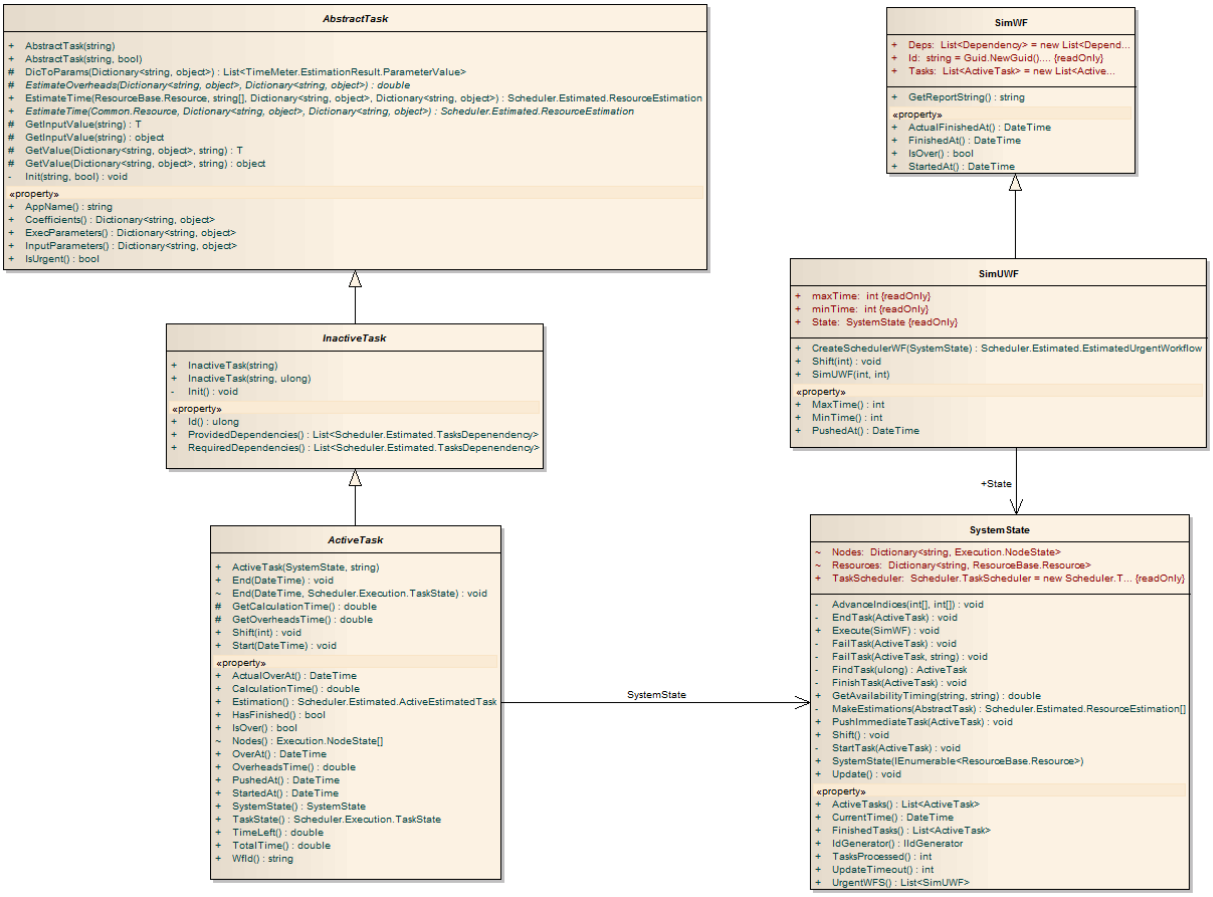

Рисунок 3.3 - Диаграмма наиболее важных классов компонента моделирования исполнения WF

3.4.1. Класс AbstractTask

Этот абстрактный класс определяет основные атрибуты задачи в системе – имя пакета, наборы входных параметров, коэффициентов и параметров вычислительной среды.

Свойства

- AppName (тип: String) – идентификатор прикладного пакета.
- Coefficients (тип: Dictionary<String, Object>) – набор коэффициентов для использования в модели производительности.
- ExecParameters (тип: Dictionary<String, Object>) – набор параметров среды исполнения.
- InputParameters (тип: Dictionary<String, Object>) – набор параметров работы пакета, извлечённых из входных файлов.
- IsUrgent (тип: bool) – флаг, показывающий, является ли данная задача экстренной.

Открытые методы

- EstimateTime – оценка времени выполнения задачи на некотором наборе узлов.
    a. Входной параметр resource (тип: Sim.ResourceBase.Resource) – описание ресурса, полученное из базы.
    b. Входной параметр nodesNames (тип: String[]) – имена узлов, для которых строится оценка.
    c. Входной параметр execParameters (тип: Dictionary<String, Object>) – набор параметров среды исполнения.
    d. Входной параметр coeffs (тип: Dictionary<String, Object>) – набор коэффициентов для использования в модели производительности.
    e. Возвращает оценку времени выполнения на данном ресурсе (тип: Scheduler.Estimated.ResourceEstimation).

3.4.2. Класс InactiveTask

Этот абстрактный класс определяет расширенный набор атрибутов абстрактной задачи в системе. Является потомком предыдущего класса.

Свойства

- Id (тип: ulong) – уникальный идентификатор задачи.
- ProvidedDependencies (тип: List<Scheduler.Estimated.TasksDependency>) – перечень зависимостей, предоставляемых данной задачей.

- RequiredDependencies (тип: List<Scheduler.Estimated.TasksDependency>) – перечень зависимостей, требуемых данной задачей.

3.4.3. Класс ActiveTask

Этот абстрактный класс определяет набор атрибутов активной задачи, а также добавляет несколько методов, имеющих смысл лишь для активной задачи. Является потомком предыдущего класса.

Свойства

- ActualOverAt (тип: DateTime) – время «реального» завершения задачи (с учётом ошибки, вызванной дискретизацией времени).
- CalculationTime (тип: double) – количество времени в секундах, оставшегося до завершения задачи.
- Estimation (тип: Scheduler.Estimated.ActiveEstimatedTask) – активная оценка, в соответствии с которой производится запуск.
- HasFinished (тип: bool) – флаг, показывающий, завершился ли расчёт в рамках данной задачи.
- IsOver (тип: bool) – флаг, показывающий, имеет ли задача статус завершённой (успешно либо нет).
- OverAt (тип: DateTime) – время завершения задачи (без учёта ошибки, вызванной дискретизацией времени).
- OverheadsTime (тип: double) – количество времени в секундах, которое составляют накладные расходы на выполнение данной задачи.
- PushedAt (тип: DateTime) – время ввода задачи в систему.
- StartedAt (тип: DateTime) – время запуска задачи.
- SystemState (тип: Scheduler.Execution.SystemState) – объект состояния системы, в котором существует данная задача.
- TaskState (тип: Scheduler.Execution.TaskState) – системный статус задачи.
- TimeLeft (тип: double) – количество времени в секундах, которое осталось до ожидаемого завершения данной задачи.
- TotalTime (тип: double) – общее количество времени в секундах, которое требуется для выполнения данной задачи (с учётом накладных расходов).
- WfId (тип: String) – идентификатор WF, к которому относится задача.

Открытые методы

- End – пометить задачу как завершённую с определённым статусом.
    a. Входной параметр endTime (тип: DateTime) – время завершения задачи.
    b. Входной параметр endState (тип: Scheduler.Execution.TaskState) – статус, с которым завершилась задача.
    c. Возвращаемое значение отсутствует (Void).
- Shift – обновить состояние задачи (увеличить текущее время)..
    a. Входной параметр interval (тип: int) – длина дискретного отрезка времени в секундах.
    b. Возвращаемое значение отсутствует (Void).
- Start – пометить задачу как запущенную.
    a. Входной параметр startTime (тип: DateTime) – время запуска задачи.
    b. Возвращаемое значение отсутствует (Void).

3.4.4. Класс SystemState

Основной класс подсистемы моделирования жизненного цикла WF, моделирующий течение времени, изменение статуса ресурсов и задач, а также работу компонента CLAVIRE/Executor.

Свойства

- ActiveTasks (тип: List<ActiveTask>) – список активных в данный момент задач.
- CurrentTime (тип: DateTime) – текущее время.
- FinishedTasks (тип: List<ActiveTask>) – список завершившихся задач.
- IdGenerator (тип: IIdGenerator) – генератор уникальных идентификаторов задач (по умолчанию выдаёт номера последовательно, начиная с единицы).
- TasksProcessed (тип: int) – количество завершённых за время существования системы задач.
- UpdateTimeout (тип: int) – длина дискретного отрезка времени в секундах.
- UrgentWfs (тип: List<SimUWF>) – список активных в данный момент экстренных WF.

Открытые методы

- Execute – запуск WF на исполнение.
    a. Входной параметр workflow (тип: SImWF) – WF, который необходимо запустить.

b. Возвращаемое значение отсутствует (void).

- PushImmediateTask – попытаться запустить задачу на указанном наборе узлов в обход системы планирования и т.п. (возможно лишь в том случае, если узлы свободны – иначе возникнет ошибка времени выполнения).
    a. Входной параметр task (тип: ActiveTask) – задача, которую необходимо запустить.
    b. Возвращаемое значение отсутствует (void).
- Shift – осуществить сдвиг текущего момента времени на величину дискретного отрезка, обновить соответствующим образом состояния активных задач.
    a. Возвращаемое значение отсутствует (void).
- Update – обновить состояния активных задач и узлов.
    a. Возвращаемое значение отсутствует (void).

3.4.5. Класс SimWF

Workflow, чей ход выполнения необходимо промоделировать.

Поля

- Deps (тип: List<Dependency>) – список зависимостей, существующих между задачами данного WF.
- Id (тип: String) – идентификатор WF.
- Tasks (тип: List<ActiveTasks>) – список задач данного WF.

Свойства

- ActualFinishedAt (тип: DateTime) – время «реального» завершения WF (с учётом ошибки, вызванной дискретизацией времени).
- IsOver (тип: bool) – флаг, показывающий, имеют ли все задачи данного WF статус завершённых (успешно либо нет).
- FinishedAt (тип: DateTime) – время завершения WF (без учёта ошибки, вызванной дискретизацией времени).
- StartedAt (тип: DateTime) – время запуска WF.

3.4.6. Класс SimUWF

Экстренный workflow, чей ход выполнения необходимо промоделировать. Является потомком предыдущего класса

Поля

- minTime (тип: int) – нижняя граница времени выполнения данного WF.
- maxTime (тип: int) – верхняя граница времени выполнения данного WF.
- State (тип: SystemState) – ссылка на объект состояния системы, в чьей среде выполняется данный WF.

Свойства

- MinTime (тип: int) – нижняя граница времени выполнения данного WF с учётом времени, прошедшего с момента его попадания в систему.
- MaxTime (тип: int) – верхняя граница времени выполнения данного WF с учётом времени, прошедшего с момента его попадания в систему.
- PushedAt (тип: DateTime) – время попадания данного WF в систему.

3.4.7. Интерфейс ISimCase

Интерфейс, используемый для описания и выполнения экспериментов.

Методы

- CreateState – создание нового объекта состояния системы.
    a. Входной параметр parameters (тип: Dictionary<String, Object>) – словарь параметров, в котором должны иметься все объекты, необходимые для создания нового системного состояния для выполнения данного эксперимента. Набор этих объектов и ключей словаря определяются спецификой эксперимента.
    b. Возвращает объект системного состояния (тип: SystemState).
- Init – инициализация внутреннего состояния эксперимента.
    a. Входной параметр parameters (тип: Dictionary<String, Object>) – словарь параметров, в котором должны иметься все объекты, необходимые для инициализации данного эксперимента. Набор этих объектов и ключей словаря определяются спецификой эксперимента.
    b. Возвращаемое значение отсутствует (void).
- ShouldContinue – проверка эксперимента на завершенность.
    a. Входной параметр state (тип: SystemState) – системное состояние, в среде которого выполняется эксперимент.
    b. Возвращает логическое значение (bool).

- Update – обновление системного состояния в соответствии со спецификой эксперимента.
    a. Входной параметр state (тип: SystemState) – системное состояние, в среде которого выполняется эксперимент и которое необходимо обновить.
    b. Возвращаемое значение отсутствует (void).
- WriteReport – формирование отчета о проведении эксперимента.
    a. Входной параметр parameters (тип: Dictionary<String, Object>) – словарь параметров, в котором должны иметься все объекты, необходимые для формирования отчёта. Набор этих объектов и ключей словаря определяются спецификой эксперимента и логикой формирования отчёта, которая может быть произвольной, и, строго говоря, может не заключаться в записи простого текстового файла или не ограничиваться ей.
    b. Возвращаемое значение отсутствует (void).

3.4.8. Класс SimExecutor

Класс, инкапсулирующий типовой алгоритм выполнения эксперимента.

Открытые методы

- Execute (статический) – запуск эксперимента.
    a. Входной параметр parameters (тип: Dictionary<String, Object>) – словарь параметров, в котором должны иметься все объекты-параметры, необходимые для выполнения эксперимента. Набор этих объектов и ключей словаря определяются спецификой эксперимента.
    b. Входной параметр simCase (тип: ISimCase) – собственно объект, описывающий эксперимент.
    c. Возвращаемое значение отсутствует (void).

3.4.9. Перечисление ExType

Возможные типы ошибок (исключений) компонента моделирования.

Возможные значения

- ARGUMENT (неверные аргументы операции).
- CONFIG (проблема с файлом конфигурации).
- EXECUTION (неожиданная ситуация в подсистеме моделирования жизненного цикла).

- INTERPRETER (ошибка при обработке зависимостей и помещении WF в подсистему моделирования жизненного цикла).
- OTHER (прочее).
- RANDOM (ошибка, связанная со случайными величинами).
- REMOTE (ошибка обращения к внешнему сервису – например, базе ресурсов или планировщику).
- RESOURCES (ошибка при работе с ресурсами, полученными из базы – например, на узлах отсутствует необходимый пакет).
- SCHEDULER (ошибка в планировщике).
- TASKS (ошибка при определении или изменении статуса задачи).

3.4.10. Класс SimException

Базовый класс исключения компонента моделирования. Потомок системного класса Exception.

Свойства

- Type (тип: ExType) – флаг, показывающий, спланировано ли выполнение данной задачи.

3.4.11. Интерфейс IDistribution

Интерфейс объекта, прелставляющего собой распределение случайной величины.

Методы

- GetNextSample (тип: double) – Получить очередное значение случайной величины из заданного распределения.

3.4.12. Класс ConfiguredNormalDistribution

Объекты этого класса позволяют получать значения из нормального распределения случайной величины, при этом параметры этого распределения извлекаются из конфигурационного файла. Данный класс не является исключительно важным в работе всего компонента, но приведение его описания представляется не лишним, так как на его примере можно понять, каким образом можно задавать распределения, настраиваемые конфигурационными файлами. Данный класс является потомком класса NormalDistribution, представляющего собой, в свою очередь, нормальное распределение случайной величины, и расширяет его функциональность лишь тем, что получает среднее

значение случайной величины и её среднеквадратичное отклонение из конфигурационного файла.

Свойства

- Alias (тип: String) – идентификатор данного распределения, используемый в конфигурационном файле.
- Mean (тип: double) – среднее значение данной случайной величины.
- StDev (тип: double) – среднеквадратичное отклонение данной случайной величины.

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Компонент моделирования 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 программную библиотеку, предназначенную для исполнения в среде Microsoft .NET 4.0 и выше.

Компонент может функционировать:

- 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
- На вычислительной системе под управлением ОС Windows (версии XP и выше) с установленной средой .NET Framework версии 3.5 или выше.

5. ВЫЗОВ И ЗАГРУЗКА

Компонент моделирования исполнения WF реализован в виде библиотеки для платформы .NET. Для его использования приложение-клиент должно ссылаться в коде на его классы. Загрузка библиотеки будет осуществлена системой управления сборок .NET автоматически при её нахождении в том же каталоге, что и приложение-клиент.

6. ВХОДНЫЕ ДАННЫЕ

Входными данными для библиотеки моделирования исполнения WF являются конфигурационные файлы, а также входные параметры методов входящих в неё классов.

Конфигурационные файлы представляют собой обычные файлы настроек приложения, характерные для .NET. Они должны включать в себя данные о привязке к внешним сервисам, а также (в секции appConfig) могут включать настройки распределений случайных величин (см. п. 3.4.12) – имена параметров в таком случае представляют собой строки вида «Идентификатор распределения»(точка)«Имя параметра». Например:

```
<add key=”CNMOverheads.Mean” value=”3.76” />
```

7. ВЫХОДНЫЕ ДАННЫЕ

Выходными данными компонента моделирования исполнения WF являются сформированные отчёты об экспериментах.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

МИТП	Многофункциональная инструментально-технологическая платформа

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводи-тельного докум. и дата	Подп.	Дата
	изме-ненных	заме-ненных	но-вых	Аннули-рован-ных					