Профессор, др. наук Синиша Срблич

канд. наук Деян Шкворц

канд. наук Даниел Скробо

канд. наук Мирослав Попович

канд. наук Иван Гавран

Иван Жужак

канд. наук Иван Бенц

канд. наук Иван Скулибер

др. наук Андро Миланович

канд. наук Матия Подравец

Профессор, доктор наук Синиша Срблич, канд. наук Деян Шкворц, канд. наук Даниел Скробо, канд. наук Мирослав Попович, канд. наук Иван Гавран. Иван Жужак

Факультет электротехники и информатики,
Университет г. Загреба, Загреб, Хорватия
Faculty of Electrical Engineering and Computing,
University of Zagreb, Zagreb, Croatia

Канд. наук Иван Бенц, канд. наук Иван Скулибер
Эрикссон Никола Тесла А.О., Загреб, Хорватия
Ericsson Nikola Tesla d.d., Zagreb, Croatia

Доктор наук Андро Миланович
Combis K.O.O., Загреб, Хорватия
Combis d.o.o, Zagreb, Croatia

Кандидат наук Матия Подравец
Телекоммуникации Хорватии А.О., Загреб, Хорватия
Hrvatske telekomunikacije d.d., Zagreb, Croatia

Ключевые слова:

Программируемое Интернет окружение

GRID- технология, ориентированная на виртуализацию вычислений

Вычислительная техника, базирующаяся на услугах

Интеграция и формирование услуг

Полипроект CRO-GRID

Key words:

Programmable Internet Environment

GRID computing

Service Oriented Computing

Service integration and composition

CRO-GRID polyproject

Резюме:

Быстрое развитие приложений, основанных на услугах, является одним из самых важных требований современного рынка информатики и телекоммуникаций. В этой статье представлено программируемое Интернет окружение (PIE – *Programmable Internet Environment*) - программная система для быстрого построения распределенных приложений посредством объединения доступных услуг. Программируемое Интернет окружение построено в соответствии с образцами и концептами вычислительной техники, базирующейся на услугах (*Service Oriented Computing*), и технологии GRID, в области распределенных вычислительных систем, важность которых растет особенно в последние годы. Система PIE обеспечивает возможность объединения удаленных компьютеров в единственную систему, а также программирования распределенных приложений посредством объединения услуг, доступных из частных сетей, или из сети Интернет. Система программируемого Интернет окружения осуществлена в рамках проекта "CRO-GRID Mediator – Посредник GRID сети Хорватии", являющейся частью полипроекта "CRO-GRID". Целью полипроекта было построение национальной GRID сети Хорватии, с необходимой программной поддержкой и прототипами приложений. Исследования в рамках проекта "CRO-GRID Mediator ", являются частью исследований проекта "Middleware Architecture (MidArc) – Межплатформенная архитектура", и проводятся Институтом телекоммуникаций компании Эрикссон Никола Тесла в сотрудничестве с учреждением электроники, микроэлектроники, вычислительных и интеллектуальных систем факультета электротехники и информатики, Университет г. Загреба.

Abstract:

Rapid development of applications based on available services is one of the most important demands of today's information and telecommunications market. This paper presents the Programmable Internet Environment (PIE), a software system for composing distributed applications from publicly exposed services. The PIE system has been created according to the paradigms and concepts of the Service Oriented Computing (SOC) and GRID. The PIE system enables connection of separate computers into a system and programming of distributed applications based on the composition of services that are located either on the computers of the system or on the computers accessible by the Internet. The PIE system has been developed as a part of the project entitled CRO-GRID Mediator, within a larger polyproject entitled CRO-GRID. The goals of the polyproject were the creation of the Croatian national GRID network with necessary software and prototype applications. The research done within the project CRO-GRID Mediator is a part of the research done within the project entitled Middleware Architecture (MidArc), conducted in collaboration between the Research Department at Ericsson Nikola Tesla's Research and Development Center, and the Department of Electronics, Microelectronics, Computer and Intelligent Systems at the Faculty of Electrical Engineering and Computing, University of Zagreb.

1. GRID окружение и вычислительная техника, базирующаяся на услугах

GRID окружение и вычислительная техника, базирующаяся на услугах (SOC - *Service Oriented Computing*), понятия из области распределенной вычислительной техники, значение которой значительно возросло в течение последних несколько лет. Большинству пользователей, которые используют компьютеры, в основном, для простых деловых приложений или мультимедиа, эти понятия все еще не очень привлекательны. Однако академические заведения и фирмы начинают использовать их возможности. Решение проблем, представляющих огромный вызов и с точки зрения математических операций, и с точки зрения емкости памяти, (*Grand Challenge Problems*), непрактично даже при применении сверхмощных компьютеров, т.к. решения нужно ждать несколько дней, или даже несколько месяцев. Примерами таких сложных проблем являются: анализ больших баз данных (порядка терабайтов и больше, например, анализ поведения пользователей или групп пользователей), автоматизированное принятие деловых решений, предвидение поведения рынка, моделирование изменений климата, метеорологическое моделирование погоды, поведение сложных биологических соединений, и т.д.

Именно применение концептов из мира GRID и вычислительной техники, основанной на услугах, обеспечивает значительное упрощение и ускорение решений перечисленных проблем. Основная идея заключается в объединении пространственно удаленных компьютеров и т.н. кластеров (*cluster* - группа взаимодействующих компьютеров, объединенных локальной сетью) в единственную систему, которая для оконечного пользователя представляет видимость одного суперкомпьютера. Способ создания виртуального суперкомпьютера и способы его использования ясно разграничивают GRID концепты и концепты вычислительной техники, базирующейся на услугах.

GRID системы объединяют компьютеры в единственную виртуальную систему, названную "Виртуальная организация" (*virtual organization*). В такой виртуальной организации записываются данные каждого компьютера, связанные с состоянием структурных параметров в данный момент. Речь идет о нагрузке процессора, емкости свободной рабочей памяти, пространстве на рабочих дисках, состоянии локальной сети и т.д. В зависимости от этих записей, задачи, посланные на решение системе GRID, распределяются компьютерам, которые в данный момент самые пригодные для их решения. Значит, в системах GRID особое внимание посвящается рабочим характеристикам аппаратных средств, с целью максимального ускорения решения задач, посланных системе. Нужно подчеркнуть отличие между кластерами и GRID системой, т.к. эти два понятия часто отождествляются - кластеры размещены на одной локации и, в основном, объединены одной локальной сетью, а одна система GRID может состоять из взаимно очень удаленных компьютеров, размещенных в различных сетях доступа.

Системы, построенные на концептах вычислительной техники, основанной на услугах, нежестко связывают (*loose coupling*) компьютеры и кластеры компьютеров в сети Интернет. Взаимосвязь компьютеров осуществляется посредством информации в центральных регистрах, или на основании данных в локальных запоминающих устройствах.

В отличие от систем GRID, в которых информация относится на аппаратные средства, в системах, основанных на услугах, информация содержит данные обо всех явно доступных программных услугах, которые может выполнять определенный компьютер.

Данные о программной услуге состоят из технических параметров (например, точка доступа услуги, требуемые параметры, категория выходных параметров, какой протокол коммуникации нужно использовать, и т.д.) и не технических параметров (семантическое описание услуги, ожидаемая скорость ответа, качество услуги, и т.д.). Пользователь объединяет и составляет услуги в более крупную смысловую целостность - процессы, или распределенное приложение, которое выполняет определенную сложную задачу. Следовательно, системы GRID очень упрощенно можно задумать как пространственно распределенный суперкомпьютер, а системы, базирующиеся на услугах, можно задумать как большой архив услуг, объединением которых приобретается дополнительное достоинство – распределенное приложение.

Важно подчеркнуть, что концепты систем GRID и вычислительной техники, базирующейся на услугах, технологически полностью независимые. Оба концепта представляют собой ряд стандартов и директив для объединения различных компьютеров в большие целостности определенного назначения. Поэтому еще недавно существовало большее число различных систем, которые осуществляли перечисленные концепты, но в различных технологиях (например, REST, RPC, DCOM, CORBA, и т.д.). Однако несколько лет назад, благодаря всюду присутствующей сети Интернет, начали преобладать системы GRID и системы вычислительной техники, базирующейся на услугах, построенные с помощью группы стандартов для Интернет-услуг (*Web Services Standards*). Так как группа стандартов для Интернет-услуг предназначена для различных видов объединения разнородных компьютеров, именно эта технология представляет основу для полной платформенной независимости GRID систем и систем, базирующихся на услугах.

В конце 2000 года исследовательский отдел Института телекоммуникаций компании А.О. Эрикссон Никола Тесла начал сотрудничество с учреждением электроники, микроэлектроники, вычислительных и интеллектуальных систем (ZEMRIS) Факультета электротехники и информатики (FER) в области распределенной вычислительной техники. В рамках этого сотрудничества доныне разработано несколько прототипов распределенных систем, последний из которых, программируемое Интернет окружение (PIE – *Programmable Internet Environment*), представляет прототип системы, базирующейся на услугах. Кроме сотрудничества компании Эриссон Никола Тесла и Факультета электротехники и информатики, программируемое Интернет окружение частично разработано и в рамках полипроекта CRO-GRID, национальной GRID инициативы Хорватии.

Программируемое Интернет окружение можно задумать как простую распределенную операционную систему, которая позволяет объединение компьютеров и их публично доступных услуг, размещенных где-либо в сети Интернет. Оконечному пользователю программируемое окружение сети Интернет предлагает возможность программирования распределенных приложений посредством объединения услуг, размещенных на этих компьютерах, словно эти услуги находятся на настольном компьютере пользователя.

2. CRO-GRID

В соответствии с общими тенденциями распределенной вычислительной техники, в течение 2003 года несколько различных исследовательских групп в Хорватии (FER и ETK, IRB, SRCE) опубликовали технологические проекты, тематически связанные с областью GRID. Почти одновременно, Министерство наук, образования и спорта также опознало важность технологии GRID в дальнейшей информатизации Хорватии. Так как проекты были взаимно дополняющими, предложено их объединение в полипроект CRO-GRID, который будет представлять национальную GRID инициативу Хорватии, подобную национальным GRID проектам в других странах.

В конце 2003 года Министерство наук, образования и спорта одобрило полипроект CRO-GRID как комплексный технологический проект в рамках программы HITRA (Новаторское технологическое развитие Хорватии). Программа HITRA представляет собой особый вид интеграции научной и технологической политики, направленной на объединение общественного научно-исследовательского сектора с хозяйством страны, которую проводит Министерство с целью способствования развитию национальной экономики, базирующейся на знании. Кроме компании Эрикссон Никола Тесла, которая активно участвовала в полипроекте CRO-GRID, в отдельные его части были включены и многие другие хорватские фирмы различных деятельностей, например, судостроительная верфь г. Риеки, Хорватская почта, предприятие “Клара”, и другие.

На рис.1. представлено устройство полипроекта CRO-GRID в рамках программы HITRA. Полипроект состоял из трех проектов: Инфраструктура, Посредник и Приложения.

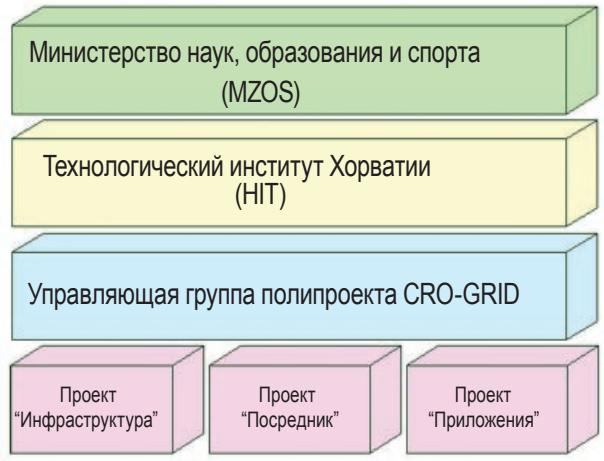


Рис. 1. Устройство полипроекта CRO-GRID в рамках программы HITRA

Руководители отдельных проектов информировали Управляющую группу о развитии и результатах проектов. Управляющая группа, которая контролировала и направляла целый полипроект, состояла из представителей Министерства наук, образования и спорта, Технологического института Хорватии (HIT), четырех университетов Хорватии и факторов национального хозяйства, включенных в полипроект. Управляющая группа о работе и развитии целого полипроекта информировала Технологический институт

Хорватии, который надзирает реализацию программы HITRA. Технологический институт также информировал Министерство наук, образования и спорта о состоянии полипроекта и целой программы HITRA.

Цели полипроекта можно разделить на технологические и не технологические цели. Главной технологической целью являлось построение национальной GRID инфраструктуры. Это не подразумевает лишь аппаратное обеспечение, но также и программную поддержку, необходимую для работы системы GRID и соответствующих приложений в реальном времени. Главной не технологической целью было объединение хорватских академических заведений в единственную GRID систему Хорватии, предлагаемую в распоряжение и академической и хозяйственной общественности. Из этой цели произошла потребность предоставления работы и образования молодых кадров, которые свои приобретенные знания и искусство перенесут хорватским исследователям, и продолжат осуществление системы GRID Хорватии. Кроме того, замечена необходимость объединения CRO-GRID с европейскими и мировыми GRID инициативами, т.е. объединения хорватской системы GRID с существующими GRID системами.

Своим включением с полипроект CRO-GRID компания Эрикссон Никола Тесла продолжила и расширила свое ранее сотрудничество с академической общественностью Хорватии. Технические вызовы, с которыми мы встретились в течение реализации полипроекта при осуществлении различных прототипов, представляют очень ценное искусство и знания, применимые в следующем поколении сетей (не только GRID). Использование различных экспериментальных технологий обеспечило нам возможность оценки полезности и выгодности приближающихся технологий.

2.1. Проект Инфраструктура

Проектом Инфраструктура управлял Вычислительный центр Университета (аббревиатура - SRCE). Основной задачей этого проекта являлось построение инфраструктуры GRID сети Хорватии, которая объединяет все хорватские университеты в единственной GRID системе. Кроме академической общественности, обеспечена возможность использования национальной GRID системы и хозяйственным факторам. Параллельно с осуществлением инфраструктуры, вычислительный центр SRCE, вместе со своими партнерами, занимался образованием научных работников в области применения новых GRID технологий.

Благодаря корректно выполненным заданиям, хорватская GRID система в 2006 году принята в среднеевропейскую федерацию GRID систем, а также в международный проект EGEEII (*Enabling Grids for E-SciencE II*), в котором договорены деятельности и обязанности до 2008 года. В рамках этих деятельностей Хорватии предоставляется возможность активного участия в дальнейшем определении и развитии GRID концептов, а также доступ к самым последним достижениям в этой области.

2.2. Проект Посредник

Факультет электротехники и информатики и компания Эрикссон Никола Тесла сотрудничали на проекте Посредник. Самой важной целью проекта было осуществление программной поддержки для построения и объединения взаимно независимых и пространственно разделенных услуг в распределенное приложение. Особое внимание уделялось приближению концептов распределенной вычислительной техники оконечному пользователю. Разработанный интерфейс пользователя, т.е. программируемое Интернет окружение и программная поддержка, детальнее представлен в главе 3.

Сам проект и постигнутые результаты представлены в многочисленных научных работах и на конференциях. Особым признанием проекту являются приглашения к чтению лекций в американских фирмах (Intel corp., Google inc., и др.), а также в университетах (Berkeley, Santa Clara, Irvine и др.).

2.3. Проект Приложения

Проект Приложения состоит из четырех подпроектов, целью которых было построение четырех прототипов приложений GRID. Ведущими организациями в этом проекте были Институт имени Руджера Бошковича, Факультет транспортных наук в Загребе, Физико-математический факультет в Загребе и Факультет электротехники, машиностроения и судостроения в Сплите. Кроме академических заведений в проекте участвовали и некоторые фирмы из Хорватии.

В рамках этого проекта построены следующие приложения: Оптимизация организации транспорта, Индексация web и локальных сетей, Моделирование и имитация сгибания протеинов, и CRO GRID портал.

Оптимизация организации транспорта решает проблему маршрутов и числа доставочных автомашин в определенном районе города, принимая во внимание параметры городского транспорта. В рамках этого приложения осуществлено сотрудничество нескольких фирм, а здесь перечислим лишь некоторые из них – Почта Хорватии, Предприятие общественного транспорта Загреба, Пресса, Коммунальные предприятия Загреба и др.

Индексацию web и локальных сетей проще всего описать как хорватский вариант поискового сервера Google, который принимает во внимание все специфичности хорватского языка и грамматики. Моделирование и имитация сгибания протеинов это пример научно-исследовательского приложения, с исключительно сложными математическими исчислениями, предназначенного пониманию поведения и пространственной структуры протеинов, с потенциальным применением в фармацевтической промышленности. Приложение CRO-GRID портал предназначено научным работникам, которые не обладают достаточными знаниями о распределенной вычислительной технике и GRID. Приложение обеспечивает им возможность использования различных GRID приложений.

Все эти приложения послужили в качестве направляющих для понимания и применения новых технологий, а также для определения способа программирования распределенных приложений в ближайшем будущем.

Кроме перечисленных приложений, в рамках проекта установлено сотрудничество с CERN (Европейский центр физики высоких энергий) и проектом SEE GRID (*South-Eastern Europe* GRID – Сеть GRID Юго-Восточной Европы), который объединяет страны юго-восточной Европы в отдельную GRID федерацию.

3. Программируемое Интернет окружение

Программируемое Интернет окружение в качестве программной системы обеспечивает быстрое и эффективное оформление распределенных приложений посредством объединения доступных услуг. Например, в GRID окружении программируемое Интернет окружение позволяет объединение услуг существующих GRID подсистем, а также построение и выполнение распределенных приложений с помощью сетевых емкостей и вычислительной мощности, предлагаемой GRID услугами. Подобно этому, в телекоммуникационном окружении программируемое Интернет окружение можно использовать для построения приложений, которые интегрируют услуги различных подсистем операторов, например, системы заботы о пользователях (*customer care*), системы оплаты и подсистемы надежности сети оператора.

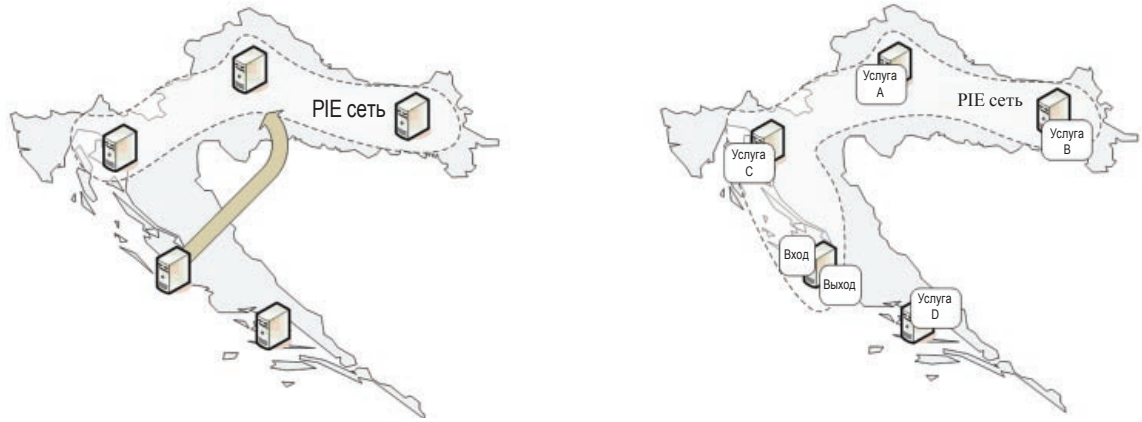
Распределенные приложения, построенные посредством программируемого Интернет окружения, состоят из трех типов услуг: прикладных услуг, услуг синхронизации и коммуникации, и соединительных программ. Прикладные услуги осуществляют специфическую логику распределенного приложения. Услуги синхронизации и коммуникации обеспечивают различные модели коммуникации и синхронизации прикладных услуг, например, соединение посредством почтовых ящиков или посредством модели “объявление-подписка” (*publish-subscribe*). Соединительные программы осуществляют логику объединения прикладных услуг и услуг синхронизации и коммуникации в полностью распределенное приложение.

Процесс построения распределенного приложения из отдельных услуг посредством программируемого Интернет окружения состоит из пяти этапов: (1) включение компьютера в сеть равноправных участников в программируемом Интернет окружении, (2) поиск и установка прикладных услуг, (3) установление услуг коммуникации и синхронизации, (4) разработка соединительных программ, (5) установка системы перевода и перевод соединительных программ.

На первом этапе пользователь, который желает использовать программируемое Интернет окружение, включает собственный компьютер в сеть равноправных участников в программируемом Интернет окружении *(Рис. 2.a)*. Это осуществляется извлечением и установкой программной поддержки для программируемого Интернет окружения на компьютере. Первый этап не обязателен при построении каждого приложения, он должен быть выполнен для каждого компьютера только один раз.

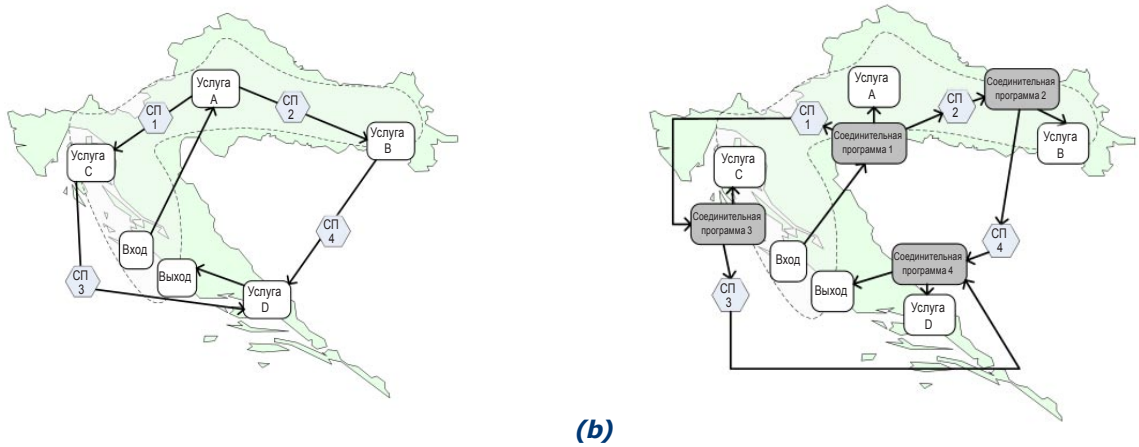
На втором этапе пользователь начинает построение распределенного приложения. Приложение строится объединением группы прикладных услуг в последовательность работ (*workflow*). Если прикладные услуги, необходимые для выполнения приложения, уже существуют, пользователь находит их внутри или вне сети программируемого Интернет окружения. Если прикладные услуги не существуют, пользователь может их построить сам и поставить в сеть. Например, на *Рис. 2.b* пользователь задумал приложение из шести услуг. Пользователь находит услуги A, B и C, поиском в программируемом Интернет окружении. Услугу D находит где-то в другом месте сети Интернет, а услуги Вход и Выход, представляющие вход и выход нового приложения, пользователь создает сам и устанавливает на собственный компьютер.

Целью третьего этапа является осмысление способов коммуникации и согласование работы услуг распределенного приложения. Программируемое Интернет окружение поддерживает четыре способа соединения и согласования работы услуг – соединение и согласование с помощью непосредственного вызова услуг, и соединение и согласование посредством семафора, почтового ящика или маршрутизатора событий. Каждый способ соединения услугосуществляет иную модель коммуникации между услугами. Непосредственный вызов услуги поддерживают стандарты для услуг сети Интернет, а остальные три модели коммуникации требуют установки дополнительных услуг коммуникации и синхронизации. Так как эти дополнительные услуги стандартные и повторяются во всех приложениях, они уже заранее построены и содержатся в программируемом Интернет окружении. Кроме того, такая



(a) *(b)*

Рис. 2. Первый и второй этапы построения распределенного приложения, основанного на программируемом Интернет окружении



(a) *(b)*

Рис. 3. Третий и четвертый этапы построения распределенного приложения, основанного на программируемом Интернет окружении

система позволяет передовое управление перечисленными услугами коммуникации и синхронизации. Например, на Рис. 3.a, пользователь решил осуществить коммуникацию между услугами A и D и услугой Выход непосредственными вызовами услуг, а вся остальная коммуникация между услугами осуществляется посредством почтовых ящиков.

На четвертом этапе, *Рис. 3.b*, пользователь создает соединительные программы, которые осуществляют передачу значений между услугами. При передаче значений соединительные программы используют предварительно установленные услуги коммуникации и синхронизации. Соединительные программы обязательны, т.к. прикладные услуги построены независимо друг от друга и от услуг коммуникации и синхронизации. Точнее, прикладные услуги не содержат встроенную логику соединения.

Типично, соединительные программы извлекают входные значения из услуг синхронизации и коммуникации, вызывают прикладные услуги и выходные значения направляют другой услуге синхронизации и коммуникации. Например, логика соединительной программы 1 (СП 1) на *Рис. 3.b* следующая:

после принятия непосредственного вызова от услуги Вход, соединительная программа 1 направляет входные параметры услуге A и принимает ответ. Полученный ответ направляется в почтовые ящики 1 и 2, из которых их далее обрабатывают другие услуги и соединительные программы. Кроме того, соединительные программы могут осуществлять и трансформирование выходных значений параметров одной услуги во входные параметры другой услуги. В программируемом Интернет окружении соединительные программы определяются с помощью языка CL или SSCL. Язык CL основан на расширяемом языке разметки, XML, и подобен языку WS-BPEL (стандартный язык для определения соединительных программ). Язык SSCL это простой язык сценариев, приспособленный использованию в программируемом Интернет окружении. Программирование соединительных программ с использованием языка SSCL упрощает и ускоряет процесс, а использование языка CL позволяет определение более сложных отношений между услугами. Кроме того, программируемое Интернет окружение обеспечивает поддержку созданию SSCL соединительных программ посредством Web интерфейса.

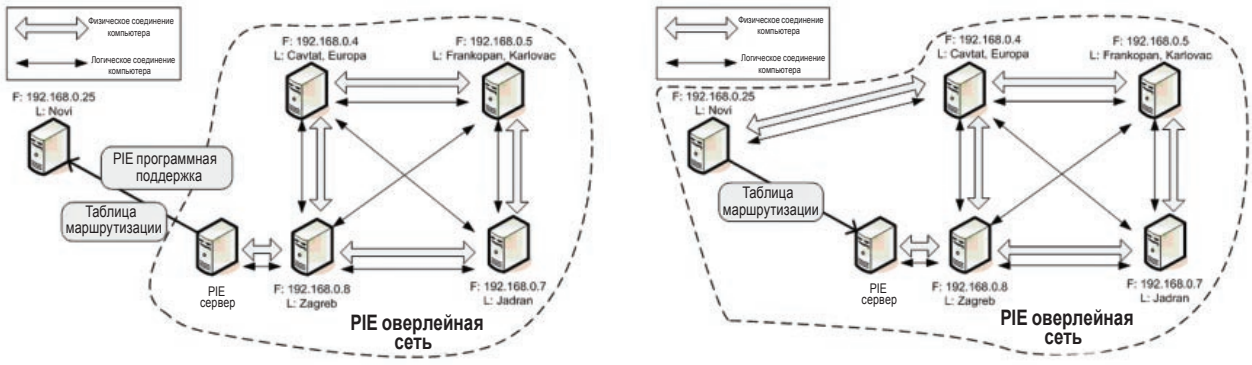
Последним этапом при построении приложения является определение системы перевода соединительных программ. А именно, так как соединительные программы, написанные языком SSCL, это программы высокого уровня, прежде выполнения их нужно перевести в формат, удобный для выполнения на компьютерах программируемого Интернет окружения. Перевод и интерпретацию соединительных программ выполняет особая подсистема, которую можно конфигурировать и таким образом приспособить каждому построенному приложению. Перевод и интерпретация соединительных программ происходит в реальном времени и обеспечивает пропорциональное наращивание распределенных приложений на большое число компьютеров. В продолжение этой главы детальнее описан каждый из перечисленных этапов осуществления распределенного приложения посредством программируемого Интернет окружения.

3.1. Включение в оверлейную сеть программируемого Интернет окружения

Основа управления инфраструктурой компьютеров в окружении развития программируемого Интернет окружения это система перекрывающей или оверлейной сети (*overlay network*). Оверлейная сеть обеспечивает возможность коммуникации компьютеров программируемого Интернет окружения внутри надежного коммуникационного окружения, логически отделенного от остатка сети общего пользования. Кроме того, оверлейная сеть позволяет осуществление адресации компьютеров программируемого Интернет окружения на основании логических адресов.

Использование логических адресов оверлейной сети, как основы коммуникации, позволяет построение приспосабливаемой системы маршрутизации, динамическое присоединение и отсоединение компьютера из системы, а также повышение надежности и устойчивости системы.

На Рис. 4. представлен процесс присоединения нового компьютера в оверлейную сеть программируемого Интернет окружения.



(a) *(b)*

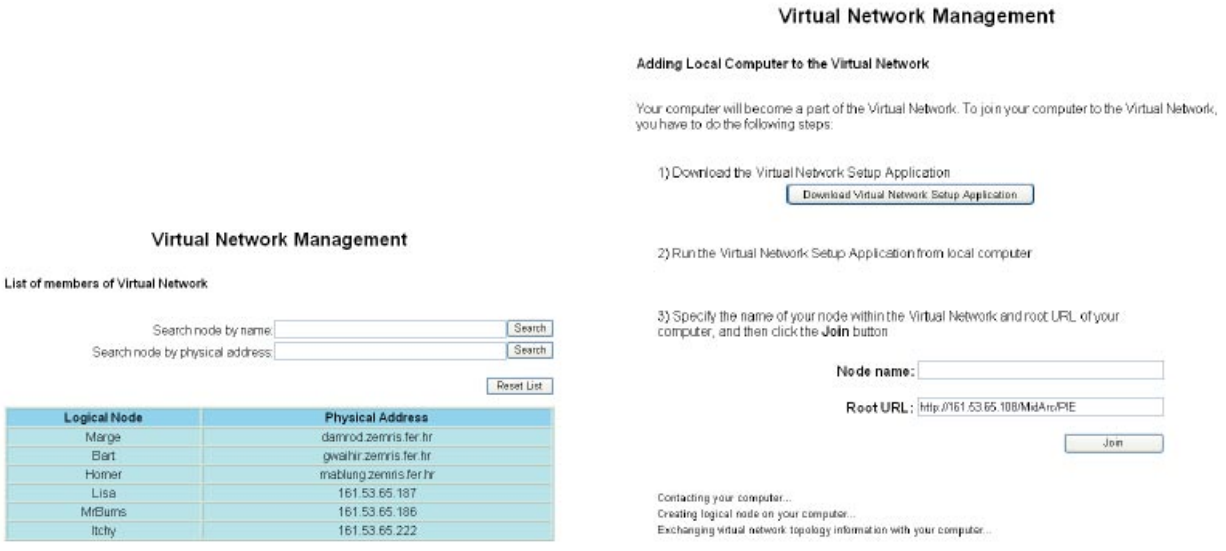
Рис. 4. Включение нового компьютера в оверлейную сеть программируемого Интернет окружения

На *Рис. 4.a* представлен пример оверлейной сети, которая состоит из четырех компьютеров. Каждому компьютеру выделен один физический адрес (F), и один или несколько логических адресов (L). Коммуникация между компьютерами системы PIE происходит с применением логических адресов.

При включении компьютера в оверлейную сеть *(Рис. 4.a)*, пользователь подключаемого компьютера соединяется с сервером программируемого Интернет окружения, выбирает программную поддержку и устанавливает ее на собственный компьютер. Программная поддержка программируемого Интернет окружения содержит подсистему для разрешения (*resolving*) логических адресов и маршрутизации вызова услуг в оверлейной сети. Кроме того, извлекается действующая в данный момент таблица маршрутизации оверлейной сети. После установления программной поддержки на компьютере *(Рис. 4.b)*, компьютер меняет таблицу маршрутизации оверлейной сети и направляет ее серверу программируемого Интернет окружения. Сервер обновляет таблицы маршрутизации остальных компьютеров оверлейной сети данными о новоподключенном компьютере.

Графический Интернет интерфейс сервера программируемого Интернет окружения обеспечивает более простую процедуру просмотра структуры оверлейной сети *(Рис. 5a.)*, т.е. существующих компьютеров, логических и физических адресов, и таблиц маршрутизации.

На *Рис. 5.b* представлен Интернет интерфейс для присоединения нового компьютера и выделения ему логического адреса, а также для устранения старого логического адреса или полностью компьютера из оверлейной сети.



(a) *(b)*

Рис. 5. Интернет-интерфейсы оверлейной сети: (a) обзор существующих узлов и таблицы маршрутизации, (b) подключение и отключение компьютеров из оверлейной сети

3.2. Управление услугами программируемого Интернет окружения

Управление услугами программируемого Интернет окружения состоит из ряда функций, обеспечивающих возможность включения в сеть новых услуг, установку и устранение услуг с компьютеров в программируемом Интернет окружении, удаление экземпляров услуг с компьютеров, доступ к услугам вне программируемого Интернет окружения и поиск услуг. На *Рис. 6.* представлена архитектура системы управления услугами. Архитектура состоит из архива услуг, регистра услуг и программной поддержки для установки услуг на компьютеры.

Архив услуг служит для хранения установочных пакетов (*installation package*) услуг, доступных в программируемом Интернет окружении. Регистр услуг служит для регистрации локаций установленных услуг, описания услуг и локаций существующих экземпляров услуг. Архив и регистр услуг являются инфраструктурными подсистемами программируемого Интернет окружения, и они установлены на сервере. Программная поддержка для установки услуг использует управляющие пакеты и на основании их устанавливает услуги на компьютеры программируемого Интернет окружения. Программная поддержка для установки услуг является частью программной поддержки программируемого Интернет окружения, которая устанавливается на компьютеры при подключении компьютера к окружению.

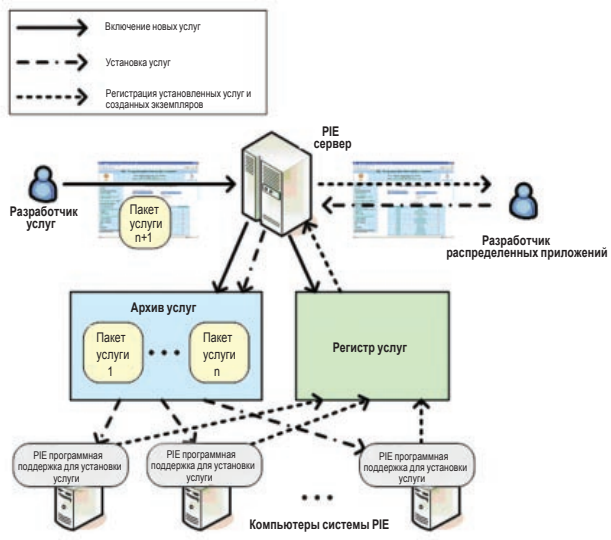


Рис. 6. Архитектура системы управления услугами

3.2.1. Включение, установка и запуск услуги

Новые услуги в программируемое Интернет окружение включают разработчики услуг, используя для этого Интернет-интерфейс, представленный на *Рис. 7.a.* В вопросник Интернета разработчики услуг доставляют файлы выполнимого программного кода новой услуги, описание интерфейса новой услуги языком WSDL (язык описания Web-услуг) и сценарий установления новой услуги на компьютеры1. На основании доставленных файлов строится установочный пакет (*installation package*) новой услуги, который вносится на хранение в архив услуг. Кроме того, регистру услуг сообщается о существовании новой услуги, которая готова для установки. Установочный пакет услуги вносится в архив только один раз, при каждой установке услуги на компьютере этот пакет извлекается из архива и передается компьютеру, на котором услуга устанавливается.

Услуги, которые сохраняются в архиве услуг, можно устанавливать на компьютеры оверлейной сети программируемого Интернет окружения. Установка услуги осуществляется с помощью интерфейса, представленного на *Рис. 7.b.* Услуга устанавливается посредством выбора логического адреса и названия услуги.

Процедура установки полностью автоматизирована и состоит из извлечения установочного пакета из архива услуг и передачи пакета программной поддержке для установки услуг удаленного компьютера. Программная поддержка для установки услуги, на основании доставленного пакета, устанавливает и конфигурирует услугу на компьютере, и сообщает о новой установленной услуге регистру услуг.

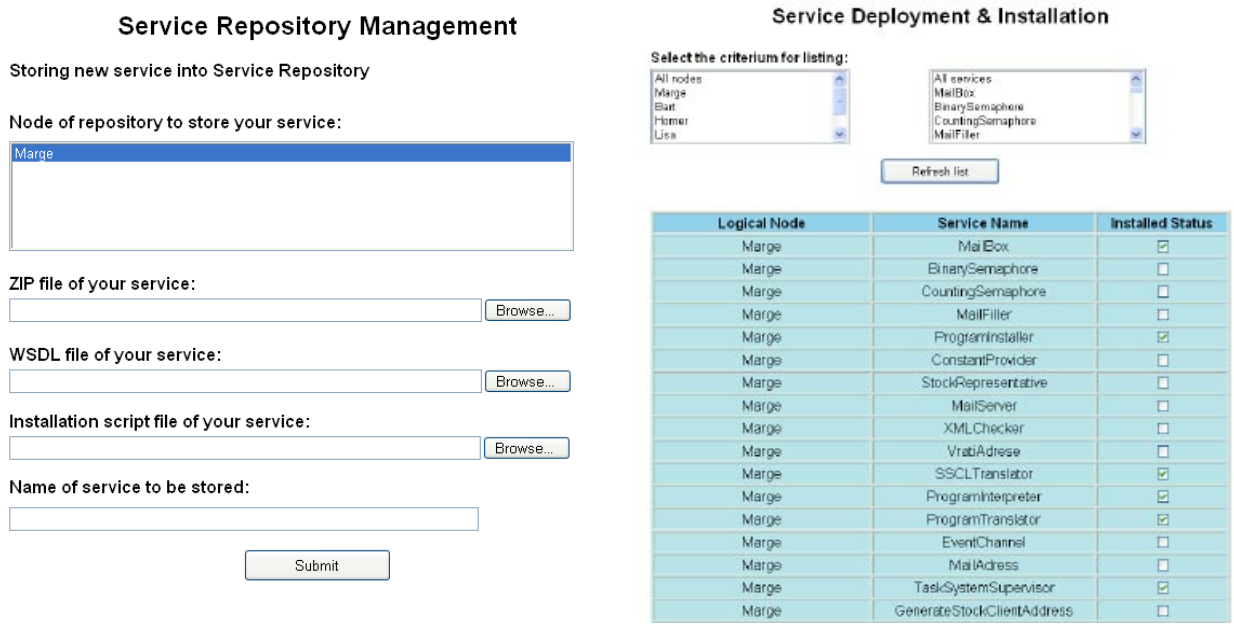
3.2.2. Управление услугами вне программируемого Интернет окружения

Программируемое Интернет окружение обеспечивает возможность управления и внешними услугами. Речь идет об услугах, осуществленных на компьютерах, которые не являются частью программируемого Интернет окружения, но доступные посредством сети общего пользования. Значит, внешние услуги невозможно установить на компьютеры программируемого Интернет окружения, их можно только вызвать. Есть целый ряд инициатив для построения услуг, которые будет возможно регистрировать как внешние услуги. Например, набор стандартов Parlay X определяет Интернет-услуги для использования услуг телекоммуникационной сети,

Исследовательский проект Com2MonSense компании Эрикссон определяет Интернет-услуги для доступа к сетям сенсоров, а инициатива Globus GRID определяет Интернет-услуги для доступа и управления подсистемами GRID.

Для регистрации внешней услуги достаточно подключиться к

[1]Нынешнее выполнение системы PIE позволяет установку услуг, осуществленных в рамках .NET программирования и предусмотренных для IIS серверов (IIS - информационный Интернет-сервер).



(a) ***(b)***

Рис. 7. Интернет-интерфейсы программируемого Интернет окружения: (a) включение услуги в систему, (b) установка услуг на компьютеры системы

Интернет-интерфейсу, представленному на *Рис.8.*, и доставить WSDL описание интерфейса внешней услуги.

Описание интерфейса услуги языком WSDL содержит все данные, необходимые для доступа к услуге. Зарегистрированная внешняя услуга и ее WSDL описание записываются в регистр услуг программируемого Интернет окружения.

3.2.3. Поиск услуг

В процессе построения распределенного приложения пользователи программируемого Интернет окружения ищут внутри системы услуги, требуемые им для осуществления приложения. Поиск услуг выполняется посредством особого Интернет-интерфейса, который позволяет им просмотр

Рис. 8. Интерфейс для регистрации новой внешней услуги

регистра услуг программируемого Интернет окружения. Если существующие там услуги не удовлетворяют запросам пользователей,

они могут, посредством ранее описанных интерфейсов, разработать новую услугу и включить ее и установить в программируемое Интернет окружение, или регистрировать внешнюю услугу.

3.3. Управление услугами коммуникации и синхронизации

Окружение развития программируемого Интернет окружения содержит набор обобщенных услуг коммуникации и синхронизации. Услуги коммуникации и синхронизации служат для осуществления различных моделей согласования и коммуникации между услугами распределенного приложения. Нынешний вариант программируемого Интернет окружения осуществляет три механизма синхронизации и коммуникации: почтовый ящик, маршрутизатор событий и семафор. Почтовый ящик обеспечивает асинхронный обмен сообщениями между услугами. Маршрутизатор событий обеспечивает коммуникацию услуг на основании механизма объявление-подписка (*publish-subscribe*). Семафор позволяет согласование выполнения множества услуг.

Все осуществленные услуги коммуникации и синхронизации позволяют создание и удаление большего числа экземпляров услуг на одном компьютере (*service instance*). Отдельные

экземпляры услуг синхронизации и коммуникации можно использовать одновременно и независимо друг от друга в различных приложениях.

Услуги синхронизации и коммуникации осуществлены с применением основных стандартов для Интернет-услуг, а также стандартов *WS-ResourceFramework* (Web-услуги – описание ресурсов) и *WS-Addressing* (Web-услуги – адресация). Осуществление этих услуг в качестве внутренних услуг программируемого Интернет окружения позволяет их установку на все компьютеры в системе.

3.3.1. Почтовый ящик

Почтовый ящик это стандартная услуга программируемого Интернет окружения, которая обеспечивает обмен сообщениями между соединительными программами и услугами приложений. Коммуникация с применением почтового ящика не зависит от времени, т.к. в почтовом ящике сообщения сохраняются так долго, пока их не извлечет адресат. Эта услуга служит для обмена сообщениями, написанными языком XML. Преимуществом использования

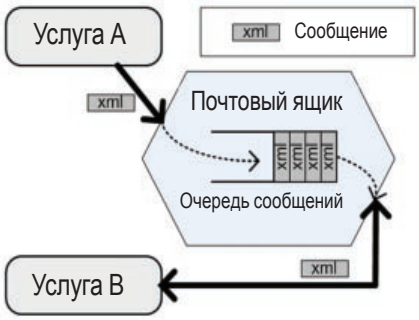


Рис. 9. Применение почтового ящика в коммуникации услуг

сообщений в XML формате является возможность передачи структур данных произвольной сложности и независимых от аппаратной, программной и разрабатывающей платформы услуг, обменивающихся сообщениями. При передаче сообщений почтовый ящик не анализирует содержание передаваемых сообщений, а только их передает.
На *Рис. 9.* представлен пример обмена сообщениями с помощью почтового ящика. Услуга А передает XML сообщения в почтовый ящик, а услуга В извлекает сообщения из почтового ящика. Если в момент подключения услуги В к почтовому ящику в очереди сообщений почтового ящика нет сообщений, услуга В входит в состояние ожидания. Когда в почтовом ящике появится сообщение, почтовый ящик автоматически направляет ее услуге В, которая продолжает с работой.

Программируемое Интернет окружение осуществляет графический Интернет-интерфейс для управления почтовыми ящиками *(Рис.10.)*. Интерфейс для управления почтовыми ящиками позволяет установку и устранение услуги почтового ящика с отдельных компьютеров программируемого Интернет окружения, а также создание и удаление экземпляров услуги на отдельных компьютерах сети. Таким образом, строителю распределенного приложения обеспечена возможность установки и запуска собственных экземпляров услуги почтового ящика на компьютерах в сети.

3.3.2. Маршрутизатор событий

Маршрутизатор событий самая сложная услуга синхронизации и коммуникации, осуществленная в нынешнем варианте программируемого Интернет окружения. Маршрутизатор событий осуществляет передовую конфигурацию коммуникационной модели “объявление - подписка”. Эта передовая коммуникационная модель основывается на XML документах, представляющих события, и на услугах “абонент”, “объявление” и “интерпретирование”. Услуги “объявление” объявляют новые события в маршрутизатор событий.

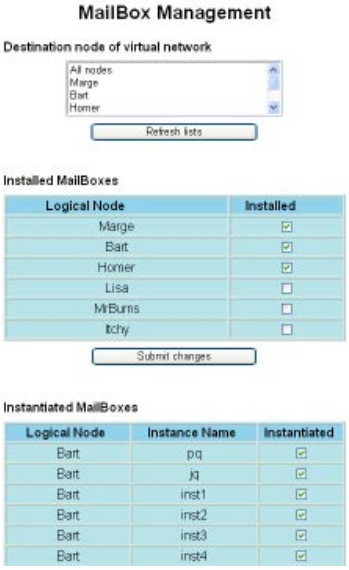


Рис. 10. Интерфейс для управления почтовыми ящиками

Услуги “Абонент” подписываются на события в маршрутизаторе событий и выбирают услугу, которую используют для интерпретирования события. Роль услуги “интерпретирование”, на основании объявленных в данный момент событий, утвердить, нужно ли информировать услугу “Абонент”. Если услугу абонентам нужно информировать, услуга интерпретирования создает интерпретированное событие, которое направляет услуге абоненту.

Маршрутизатор событий по-разному поступает с различными типами событий. Маршрутизатор событий различает четыре типа событий: постоянные события, расходуемые события, события объявления и интерпретированные события *(Рис. 11.)*. Постоянные события, расходуемые события, события объявления и интерпретированные события объявляют услуги “объявление”. После объявления постоянное событие вносится в группу событий маршрутизатора и остается в группе до тех пор, пока не будет отозвано.

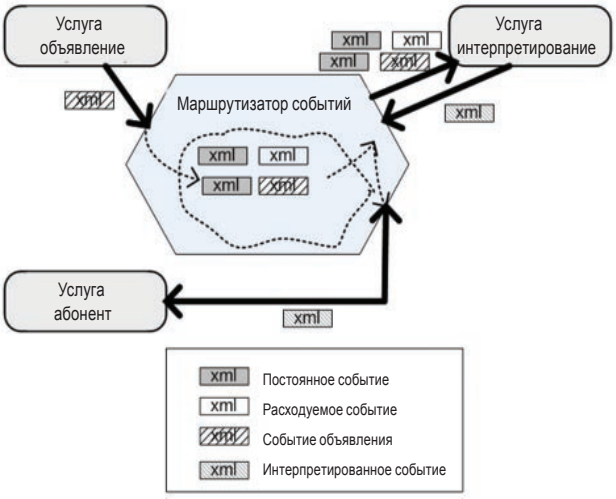


Рис. 11. Применение маршрутизатора событий в коммуникации услуг

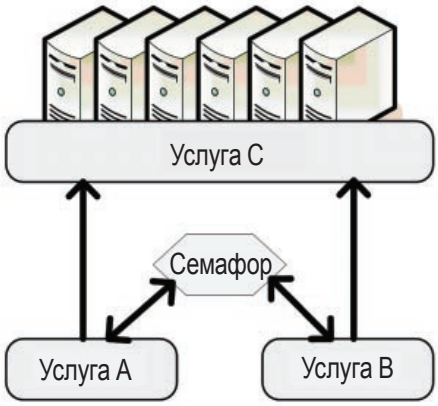


Рис. 12. Применение бинарного семафора для согласования работы услуг

Расходуемые документы вносятся в группу объявленных событий, но из группы автоматически удаляются, когда первая услуга интерпретирования подтвердит выполнение условий подписки. События объявления один раз записываются в группу событий и посылаются всем услугам интерпретирования. Интерпретированные события это события, которые создают услуги интерпретирования на основании группы событий маршрутизатора, когда выполнены условия подписки.

После каждого объявления события (постоянного, расходуемого или объявления), маршрутизатор вызывает все присоединенные услуги интерпретирования и направляет им группу событий, которые в данный момент существуют в маршрутизаторе. Услуги интерпретирования проверяют, исполняет ли данная группа событий условия подписки. Если какой-нибудь интерпретатор событий утвердит, что условия подписки выполнены, он создает интерпретированное событие и возвращает его маршрутизатору. Принятое интерпретированное событие маршрутизатор направляет той услуге абонентов, чей интерпретатор создал интерпретированное событие.

Управление маршрутизаторами событий проводится посредством графического Интернет-интерфейса. Графический интерфейс позволяет установку и устранение маршрутизатора событий с компьютеров программируемого Интернет окружения, а также запуск и выключение экземпляров услуги “маршрутизатор событий” на этих компьютерах. Графический Интернет-интерфейс для управления маршрутизаторами событий равнозначен интерфейсу для управления почтовыми ящиками.

3.3.3. Бинарный и общий семафор

Третьим механизмом синхронизации и коммуникации являются семафоры. Семафоры, прежде всего, предназначены для синхронизации выполнения услуг и соединительных программ. Существуют два типа семафоров: бинарный и общий. Бинарный семафор обеспечивает возможность взаимного выключения соединительных программ и услуг, т.к. в любой момент его может занимать только одна услуга, или одна соединительная программа. После занятия бинарного семафора,

услуга или соединительная программа должна его освободить, чтобы другая услуга или соединительная программа могла снова его занять. Общий семафор является расширением бинарного семафора и служит для защиты ресурсов (*resources*). По сравнению с бинарным семафором, общий семафор могут одновременно занять несколько соединительных программ или услуг. Максимальное число услуг, которые одновременно могут занять семафор, задается при создании экземпляра общего семафора.

Самым частым применением семафора в окружении развития PIE является ограничение одновременного доступа к компьютеру посредством заданного числа распределенных программ. На *Рис.12.* представлен пример использования бинарного семафора для ограничения одновременного доступа к услуге С, которая осуществляет доступ и использование группы компьютеров. Прежде доступа к услуге С, услуга А подключается к семафору и просит разрешения для доступа к услуге С. Так как семафор не занят, услуге А

разрешается доступ к услуге С и услуга А подключается к услуге С. Когда услуга В потребует доступ к семафору, доступ ей запрещается, т.к. услуга А уже заняла семафор. После окончания использования услуги С, услуга А освобождает семафор. После освобождения семафор сообщает услуге В, что доступ к средству разрешен, и услуга В может использовать услугу С.

Управление бинарным и общим семафорами осуществляется с помощью графического Интернет-интерфейса, равнозначного интерфейсу управления почтовым ящиком. Этот интерфейс позволяет установку и устранение услуги семафор с компьютеров программируемого Интернет окружения, а также запуск и выключение экземпляров услуги семафор на этих компьютерах.

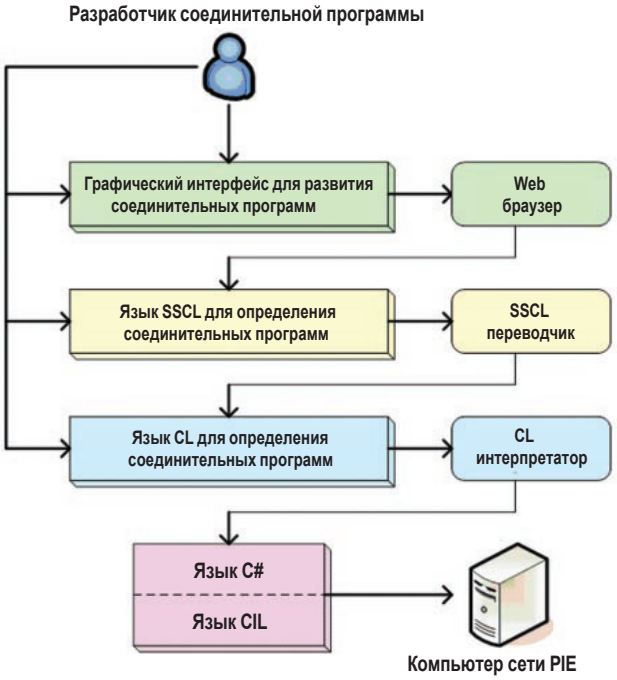


Рис. 13. Иерархия языков развития в окружении развития PIE

3.4. Построение соединительных программ

Соединительные программы объединяют внутренние и внешние услуги программируемого Интернет окружения, а также услуги синхронизации и коммуникации в единственное распределенное приложение. Соединительные программы можно осуществить с помощью нескольких языков, связанных по иерархии. На *Рис.13.* представлена иерархия языков, доступных в программируемом Интернет окружении. В самом верху иерархии находится графический Интернет-интерфейс для развития соединительных программ. Графический интерфейс обеспечивает возможность простого развития соединительных программ посредством Интернет-браузера (программа просмотра). Развитие соединительных программ с помощью графического интерфейса происходит быстро и просто, но возможность построения более сложных соединительных программ очень малая.

Графический интерфейс для развития соединительных программ создает программный код на языке SSCL (*Simple Service Composition Language*). Язык SSCL это простой язык сценариев, приспособленный для развития соединительных программ, которые предназначены для соединения прикладных услуг с помощью услуг коммуникации и синхронизации. Кроме автоматического создания SSCL команд посредством графического Интернет-интерфейса, более опытные программисты могут использовать язык SSCL непосредственно и, таким образом, осуществить лучший надзор над структурой соединительной программы.

Язык SSCL переводится на язык CL (*Coopetition Language*) с помощью переводчика SSCL. Основными преимуществами языка CL являются совместимость со стандартами Интернет-услуг, и возможность простой передачи и выполнения на всех компьютерах программируемого Интернет окружения.

Язык CL определяет управление вызовами услуг на уровне SOAP сообщений, и управление данными на уровне XML структур данных. Указанные свойства обеспечивают полный контроль над выполнением соединительных программ. Однако, из-за обширности и сложности XML структуры, язык CL не удобен для непосредственного применения. Поэтому язык CL чаще всего применяется в качестве выхода SSCL переводчика, а очень редко используется опытными программистами для небольших вмешательств.

В практическом выполнении программируемого Интернет окружения в рамках .NET, язык CL переводится двухэтапным процессом CL переводчика на язык CIL. На первом этапе процесса перевода язык CL переводится на язык C#, один из основных языков развития услуг в рамках .NET (.NET *framework*). На втором этапе перевода язык C# переводится на язык CIL (*Common Intermediate Language* – Общий промежуточный язык) с помощью встроенного переводчика в рамках .NET. Программы, написанные языком CIL, непосредственно выполняются с помощью виртуального механизма в рамках .NET.

3.4.1. Язык CL

С помощью языка CL определяются соединительные программы на уровне управления SOAP (Простой протокол доступа к объектам) сообщениями, которые проходят через соединительную программу. SOAP сообщения придерживаются XML синтаксиса и представляют основную

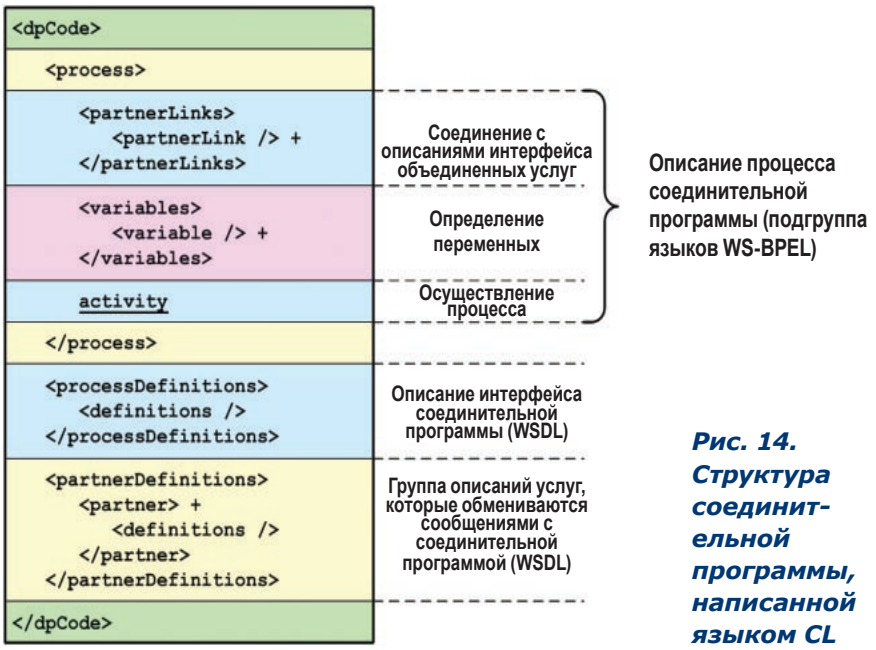


Рис. 14. Структура соединительной программы, написанной языком CL

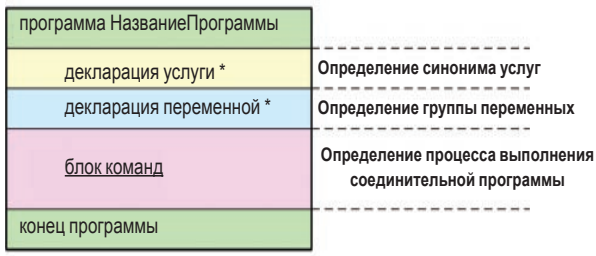


Рис. 15. Структура программ, написанных языком SSCL

единицу коммуникации между услугами сети Интернет. Языком CL определяется, что соединительная программа должна сделать после принятия определенного SOAP сообщения, точнее, какие трансформации сообщения нужно выполнить, и куда его послать.

Структура соединительной программы, написанной на языке CL, представлена на *Рис. 14.* Язык CL основан на XML структуре и состоит из описания процессов соединительной программы, описания интерфейса соединительной программы и группы описаний интерфейсов услуг, с которыми соединительная программа обменивается SOAP сообщениями. Описания интерфейса соединительной программы и интерфейсов всех услуг, с которыми соединительная программа обменивается SOAP сообщениями, выполнены стандартным языком WSDL. Описание процесса основывается на языке WS-BPEL, однако поддержана только сжатая подгруппа функций этого языка. Описание процесса содержит определения всех сообщений, обменивающихся между соединительной программой и другими услугами. Кроме того, определяется группа локальных переменных, которые могут сохранять промежуточные значения параметров вызова при вызове большего числа услуг. И, наконец, описывается ряд активностей, определяющих процесс. Типичная последовательность активностей – принятие SOAP сообщения, передача сообщения другой услуге (вызов другой услуги), извлечение ответного значения и передача его третьей услуге.

3.4.2. Язык SSCL

Язык SSCL разработан из-за сложности языка CL. Структура языка SSCL не основывается на XML, она подобна языкам сценариев. Языковые конструкции языка SSCL формированы согласно группе языковых конструкций языка WS-BPEL, со значительно упрощенным синтаксисом. Язык SSCL скрывает от пользователя сложность управления XML данными, определения соединений с WSDL интерфейсами, и управления SOAP сообщениями. Кроме того, язык SSCL содержит специально приспособленные языковые конструкции, которые дополнительно облегчают определение взаимодействия соединительной программы с услугами синхронизации и коммуникации. Поэтому применение языка SSCL позволяет построение соединительных программ со значительно сокращенным кодом, более простым и понятным человеку.

На Рис. 15. представлена структура программы, написанной языком SSCL. Тело программы разделено на три основные части. Первую часть составляют декларации услуг, вызываемых из программы. В декларациях услуг определяются псевдонимы (*alias*), под которыми будут вызываться услуги в соединительной программе. Декларации услуг используют имена услуг, перечисленных в регистре услуг программируемого Интернет окружения, избегая сложного внесения WSDL описания интерфейса услуги. Вторую часть тела программы составляет декларация переменных, в которой перечислены переменные, которые использует соединительная программа. Последнюю часть тела составляет блок команд для вызова услуг и управления процессом выполнения соединительной программы. На Рис. 16. представлена разница в определении соединительных программ языками SSCL и CL на примере перевода одной команды языка SSCL в равнозначную, сложную XML структуру языка CL.

3.4.3. Графический интерфейс для развития соединительных программ

Графический интерфейс ведет пользователя при развитии соединительных программ, осуществляя полуавтоматическое редактирование программных файлов соединительных программ.

Полуавтоматический поступок помогает пользователю при построении программы, предлагая выбор языковых

Выполнение вызова услуги языком SSCL

```
invoke serviceURL, operation, inputVar, outputVar
```

Равнозначное выполнение вызова услуги языком CL

```
<assign>
   <copy>
     <from>
        ...
     </from>
     <to partnerLink="ServicePL" />
   </copy>
</assign>
<assign>
   <copy>
     <from>
        ...
     </from>
     <to variable="srvcRqstMsg" part="params" />
   </copy>
</assign>
<invoke partnerLink="ServicePL" portType="..."
 operation="..." inputVariable="srvcRqstMsg"
 outputVariable="srvcRsltMsg" />
```

Рис. 16. Пример SSCL команды invoke и равнозначного CL кода

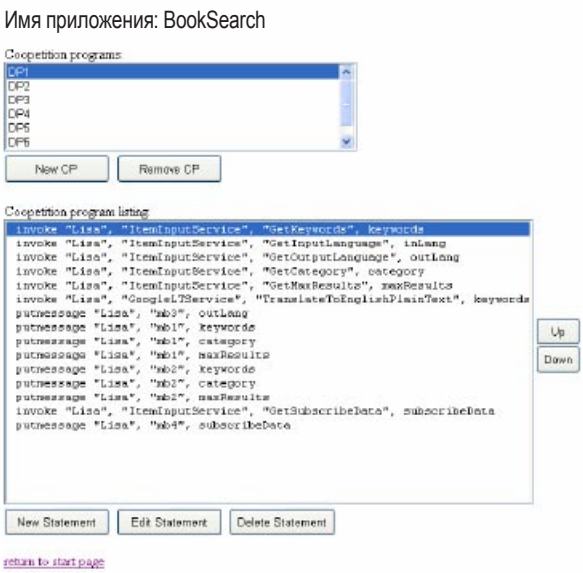


Рис. 17. Интерфейс для развития соединительных программ

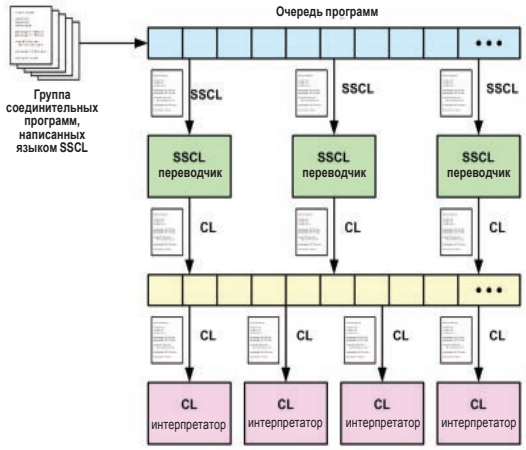


Рис. 18. Двухуровневая архитектура перевода, распределения и выполнения

конструкций языка SSCL, которые пользователь может использовать в программе. После выбора пользователем языковой конструкции, графический интерфейс автоматически строит синтаксическую структуру, а пользователь лишь вносит значения отдельных параметров.

Кроме того, интерфейс для развития программы объединен с данными из регистра услуг и при определении параметров языковых конструкций предлагает возможные значения имен услуг и экземпляров услуг.

Распределенное приложение состоит из группы соединительных программ, которые строятся независимо друг от друга. Построение соединительных программ основывается на интерфейсе, представленном на *Рис. 17.* Интерфейс позволяет редактирование соединительной программы с помощью введения новых команд, а также изменений и стираний существующих команд. Построенный графический интерфейс окружения развития программируемого Интернет окружения обеспечивает возможность простого и быстрого развития соединительных программ и распределенных приложений.

3.5. Перевод соединительных программ

Распределенное приложение состоит из группы прикладных услуг, группы услуг синхронизации и коммуникации, и группы соединительных программ. Прикладные услуги и услуги синхронизации и коммуникации устанавливаются на компьютеры с помощью системы управления услугами. Соединительные программы осуществляют процесс выполнения приложения посредством вызова прикладных услуг и взаимной синхронизацией и коммуникацией. Выполнение соединительных программ, а значит и распределенного приложения, в программируемом Интернет окружении основывается на двухуровневой архитектуре перевода, распределения и выполнения, представленной на *Рис. 18.*

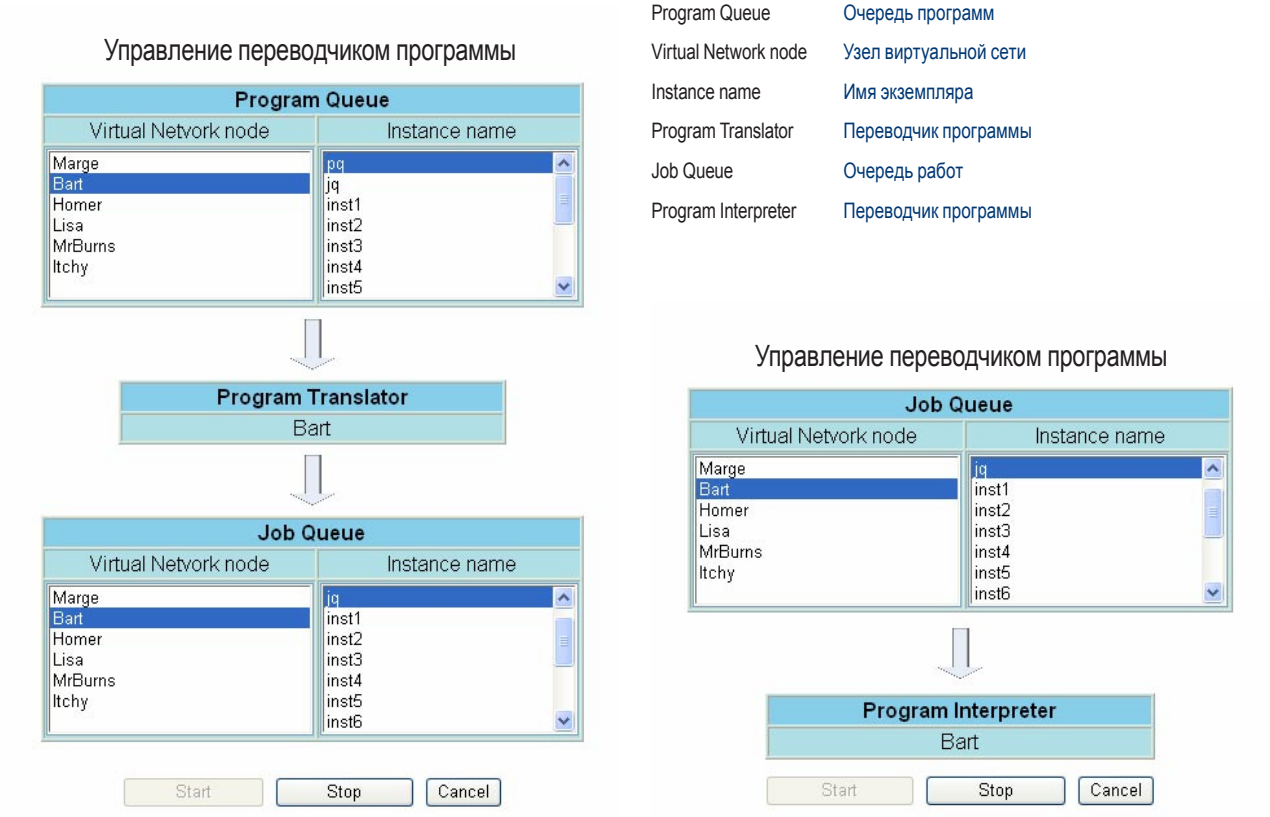
Двухуровневая архитектура перевода, распределения и выполнения состоит из очереди программ, очереди работ, переводчиков языка SSCL и переводчиков языка CL. Очереди программ и работ осуществлены с помощью услуги почтового ящика. После запуска приложения, все соединительные программы, из которых состоит приложение, ставятся в очередь программ. Переводчики языка SSCL извлекают соединительные программы, написанные языком SSCL, из очереди программ, переводят их в соединительные программы, написанные языком CL, и ставят в очередь работ. Каждый SSCL переводчик работает в цикле, состоящем из извлечения одной SSCL программы из очереди программ, ее перевода на язык CL и добавления в очередь работ. Переводчики SSCL повторяют описанный цикл до тех пор, пока их не остановит пользователь.

В очереди работ находятся соединительные программы, написанные языком CL. Переводчики CL установлены на компьютеры программируемого Интернет окружения. Каждый переводчик соединяется с очередью работ, извлекает одну CL соединительную программу и выполняет ее на компьютере, на котором она установлена. Описанный цикл извлечения и выполнения повторяется до тех пор, пока пользователь не остановит выполнение переводчика.

3.5.1. Управление процессом перевода

Управление двухуровневой архитектурой перевода, распределения и выполнения соединительных программ в окружении развития программируемого Интернет окружения разделено на управление SSCL переводчиками и управление CL переводчиками. Управление SSCL переводчиками обеспечивает возможность установки и устранения SSCL переводчика с компьютера сети PIE, а также управление способом соединения SSCL переводчика с очередями программ и работ.

Управление CL переводчиками позволяет установку и устранение CL переводчика с компьютера программируемого Интернет окружения и определение очереди работ, из которой отдельные переводчики извлекают соединительные программы.



(a) ***(b)***

Рис. 19. Определение перевода соединительных программ: (a) соединение переводчика с очередью программ и очередью работ, (b) соединение переводчика с очередью работ

На *Рис. 19.a* представлен интерфейс для управления соединением SSCL переводчика с очередями программ и работ, а на *Рис. 19. b* представлен интерфейс для определения соединения CL переводчика с очередью работ.

Для каждого SSCL переводчика определяется почтовый ящик, из которого переводчик извлекает SSCL программы, и почтовый ящик, в который направляет CL программы. Для каждого CL переводчика определяется почтовый ящик, из которого переводчик извлекает CL программы, которые он будет выполнять.

Благодаря наличию очередей и возможности динамического конфигурирования SSCL и CL переводчиков, построенная архитектура перевода, распределения и выполнения обеспечивает пропорциональный рост распределенного приложения и высокую степень приспособляемости потребностям пользователей.

4. Вывод

В данной статье представлена система программируемого Интернет окружения (*Programmable Internet Environment*). Система предусмотрена для объединения общедоступных услуг взаимно удаленных компьютеров в комплексное распределенное приложение. Услуги могут предлагать различные функции, от использования различных ресурсов компьютеров и сетей, до доступа к знаниям научных заведений. Система программируемого Интернет окружения разработана как часть проекта “CRO-GRID посредник”, в рамках которого представлена на многих конференциях и семинарах, а также в элитных американских университетах и фирмах. Совокупным полипроектом CRO-GRID создана хорватская национальная сеть GRID, которая включена в идентичные европейские GRID системы. Разработаны прототипы приложений и представлены возможности их коммерческого применения.

С помощью программируемого Интернет окружения устанавливается сеть равноправных участников, которая соединяет удаленные заведения, и создает надежное и выделенное окружение для выполнения распределенного приложения. Кроме того, программируемое Интернет окружение устанавливает особые услуги синхронизации и коммуникации, которые позволяют осуществление различных моделей согласования и обмена сообщений между услугами, расположенными в удаленных заведениях. Также, с помощью программируемого Интернет окружения строятся соединительные программы, которые с помощью услуг синхронизации и коммуникации объединяют услуги удаленных заведений в смышленое единственное приложение. Управление всеми функциями программируемого Интернет окружения и создание новых распределенных приложений осуществляется с помощью Интернет-браузера и сервера программируемого Интернет окружения. Несложный Интернет-интерфейс обеспечивает простой и приспособленный пользователю надзор системы и построение распределенных приложений.

5. Список сокращений

SOC	Service Oriented Computing ***Вычислительная техника, базирующаяся на услугах***
IP	Internet Protocol ***Интернер Протокол***
XML	Extensible Markup Language ***Расширяемый язык разметки***
WSDL	Web Service Description Language ***Язык описания Web-услуг***
SOAP	Simple Object Access Protocol ***Простой протокол доступа к объектам***
UDDI	Universal Description, Discovery and Integration ***Универсальное описание, поиск и взаимодействие***
WS-BPEL	Web Services – Business Process Execution Language ***Язык описания выполнения деловых процессов Web-услуг***
WS-Addressing	Web Services – Addressing ***Web-услуги - Адресация***
WS-RF	Web Services – Resource Framework ***Web-услуги – Описание ресурсов***
PIE	Programmable Internet Environment ***Программируемое Интернет окружение***
CL	Composition Language ***Язык переводчика соединительных программ***
SSCL	Simple Service Composition Language ***Простой язык сценариев***
HIT	Hrvatski institut za tehnologiju ***Хорватский технологический институт***
HITRA	Hrvatski inovacijski tehnologijski razvitak ***Новаторское технологическое развитие Хорватии***

6. Литература

[1] Д. Шкворц: „Prividna mreža računalnih sustava zasnovanih na uslugama – Виртуальная сеть вычислительных систем, базирующихся на услугах”, Кандидатская диссертация, Факультет электротехники и информатики, Университет в Загребе, октябрь 2005 года.

[2] М. Подравец: „Otkrivanje i postavljanje usluga u sustavima zasnovanim na uslugama – Создание и установка услуг в системах, базирующихся на услугах“, Кандидатская диссертация, Факультет электротехники и информатики, Университет в Загребе, 2005 год.

[3] А. Милановић, С. Срблић, Д. Скробо, Д. Чапалија, С. Решковић: “Coopetition Mechanisms for Service-Oriented Distributed Systems – Мехинизмы сотрудничества распределенных систем, базирующихся на услугах”, Proceedings of the CCCT 2005 (The 3rd International Conference on Computing, Communications and Control Technologies), Austin, Texas, USA, July 2005, Vol. 1, pp. 118-123.

[4] С. Решковић: „Sinkronizacijski i komunikacijski mehanizmi za sustave zasnovane na uslugama – Механизмы синхронизации и коммуникации для систем, базирующихся на услугах“, дипломная работа, Факультет электротехники и информатики, Университет в Загребе, сентябрь 2005 года.

[5] Д. Чапалија: „Objava/pretplata mehanizmi za ostvarivanje mreža zasnovanih na sadržaju – Механизмы объявление/подписка для осуществления сетей, базирующихся на содержании“, дипломная работа, Факультет электротехники и информатики, Университет в Загребе, май 2005 года.

[6] Д. Скробо: „Raspodijeljeno usporedno interpretiranje programa u arhitekturama zasnovanim na uslugama – Распределенное параллельное интерпретирование программ в архитектурах, основанных на услугах“, Кандидатская диссертация, Факультет электротехники и информатики, Университет в Загребе, октябрь 2005 года.

[7] И. Гавран: „Korisnički jezik programskog modela zasnovanog na uslugama – Пользовательский язык программной модели, основанной на услугах“, Кандидатская диссертация, Факультет электротехники и информатики, Университет в Загребе, 2005 год.

[8] Д. Скробо, А. Милановић, С. Срблић: “Distributed Program Interpretation in Service-Oriented Architectures – Интерпретация распределенных программ в архитектурах, основанных на услугах”, Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2005), Orlando, Florida, USA, July 2005, Vol. 4, pp. 193-197.

[9] А. Милановић: „Programski model zasnovan na uslugama – Программная модель, основанная на услугах“, докторская диссертация, FER, Университет в Загребе, 2006.

[10] World Wide Web consortium: „Extensible Markup Language (XML) 1.0“, http://www.w3.org/TR/REC-xml/

[11] World Wide Web consortium: „SOAP Version 1.2“, http://www.w3.org/TR/soap/

[12] World Wide Web consortium: „Web Services Description Language (WSDL) 1.1“, http://www.w3.org/TR/wsdl

[13] OASIS: „UDDI Spec Technical Committee Draft“, http://uddi.org/pubs/uddi_v3.htm

[14] OASIS: „Business Process Execution Language for Web Services“, http://www.oasis-open.org/committees/wsbpel/charter.php

[15] World Wide Web consortium: „Web Services Addressing (WS-Addressing)“, http://www.w3.org/Submission/wsaddressing/

[16] OASIS: „Web Services Resource 1.2 (WS-Resource)“, http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf

[17] The Parlay Group: „Parlay X specifications“, http://www.parlay.org/en/specifications/pxws.asp

Адреса авторов:

Синиша Срблич
e-mail: sinisa.srbljic@fer.hr
Факультет электротехники и информатики,
Университет в Загребе
Unska 3
HR-10000 Zagreb
Хорватия

Андро Миланович
e-mail: andro.milanovic@combis.hr
Combis d.o.o.
Baštijanova 52a
HR-10000 Zagreb
Хорватия

Деян Шкворц
e-mail: dejan.skvorc@fer.hr
Факультет электротехники и информатики,
Университет в Загребе
Unska 3
HR-10000 Zagreb
Хорватия

Даниел Скробо
e-mail: daniel.skrobo@fer.hr
Факультет электротехники и информатики,
Университет в Загребе
Unska 3
HR-10000 Zagreb
Хорватия

Мирослав Попович
e-mail: miroslav.popovic@fer.hr
Факультет электротехники и информатики,
Университет в Загребе
Unska 3
HR-10000 Zagreb
Хорватия

Иван Гавран
e-mail: ivan.gavran@fer.hr
Факультет электротехники и информатики,
Университет в Загребе
Unska 3
HR-10000 Zagreb
Хорватия

Матия Подравец
e-mail: matija.podravec@fer.hr
Телекоммуникации Хорватии А.О.
Savska cesta 32
HR-10000 Zagreb
Хорватия

Иван Жужак
e-mail: ivan.zuzak@fer.hr
Факультет электротехники и информатики,
Университет в Загребе
Unska 3
HR-10000 Zagreb
Хорватия

Иван Скулибер
e-mail: ivan.skuliber@ericsson.com
Ericsson Nikola Tesla d.d.
Krapinska 45
p.p. 93
HR-10002 Zagreb
Хорватия

Иван Бенц
e-mail: ivan.benc@ericsson.com
Ericsson Nikola Tesla d.d.
Krapinska 45
p.p. 93
HR-10002 Zagreb
Хорватия

Редакция приняла рукопись 3 мая 2007 года.

Перевод: Надежда Племенчич

Перевод текста рисунков 5a и 5b, стр. 11

Virtual Network Management	Управление виртуальной сетью
List of members of Virtual Network	Список членов виртуальной сети
Search node by name	Поиск узла по имени
Search node by physical address	Поиск узла по физическому адресу
Search	Поиск
Reset list	Обновление списка
Logical Node	Логический узел
Physical Address	Физический адрес
Virtual Network Management	Управление виртуальной сетью
Adding Local Computer to the Virtual Network	Присоединение локального компьютера к виртуальной сети
Your computer will become a part of the Virtual Network. To join your computer to the Virtal Network, you have to do following steps:	Ваш компьютер станет частью виртуальной сети. Для присоединения к виртуальной сети вы должны сделать следующее:
1) Download the Virtual Network Setup Application	1) Скачать программную поддержку для виртуальной сети
2) Run the Virtual Network Setup Application from local computer	2) Выполнение программной поддержки для виртуальной сети на присоединяемом компьютере
3) Specify the name of your node within the Virtual Network and root URL of your computer, and then click the Join button	3) Специфицировать имя вашего узла внутри виртуальной сети и URL (адрес) вашего компьютера, а затем щелкнуть мышью Join
Node name:	Имя узла:
Root URL:	Унифицированный указатель ресурса, URL адрес:
Join	Присоединить
Contacting your computer	Контактирование вашего компьютера
Creating logical node on your computer	Создание логического узла на вашем компьютере
Exchanging virtual network topology information with your computer	Изменение информации о топологии виртуальной сети

Перевод текста рисунков 7a и 7b, стр. 13

Service Repository Management	Управление архивом услуг
Storing new service into Service Repository	Сохранение новой услуги в архиве услуг
Node of repository to store your service:	Узел архива для сохранения вашей услуги
ZIP file of your service:	ZIP файл вашей услуги
WSDL file of your service:	WSDL файл вашей услуги
Installation script file of your service:	Установочный файл сценария вашей услуги
Name of service to be stored:	Название сохраняемой услуги
Browse...	Просмотр
Service Deployment & Installation	Внедрение и установка услуги
Select the criterium for listing:	Выбор критериев для листинга
All nodes	Все услуги
All services	Все узлы
Refresh list	Обновить список
Logical Node	Логический узел
Service Name	Имя услуги
Installed Status	Установленный статус

External Service Registration	Регистрация внешней услуги
Registration of new external service	Регистрация новой внешней услуги
To register external serice, you should provide its WSDL description.	Для регистрации внешней услуги вы должны предоставить ее WSDL описание.
Use the form below to upload the WSDL file of the service you want to register.	Используйте нижний бланк для загрузки WSDL файла услуги, которую желаете регистрировать.
You should use an out-of-band method to retrieve the service WSDL. For example, you can visit official web site of the required service and download the service WSDL.	Используйте внеполосный (out-of-band) метод для извлечения услуги WSDL. Например, вы можете посетить официальный web сайт требуемой услуги и скачать услугу WSDL.
WSDL file:	WSDL файл
Browse	Просмотр
Optionally, you can use the form below to set the link to the detailed service description.	По вашему выбору, можете использовать нижний бланк для установки соединения с подробным описанием услуги.
Link to service description:	Ссылка к описанию услуги
Register Service	Регистрация услуги

Перевод текста рисунка 8, стр. 13

MailBox Management	Управление почтовым ящиком
Destination node of vitual network	Узел назначения виртуальной сети
Refresh lists	Обнови списки
Installed MailBoxes	Установленные почтовые ящики
Logical node	Логический узел
Installed	Установлен
Submit changes	Предложить изменения
Instantiated MailBoxes	Обработанные почтовые ящики
Instance Name	Имя экземпляра
Instantiated	Подвергнуто обработке

Перевод текста рисунка 10, стр. 14

Coopetition programs (CP):	Взаимодействующие программы:
New CP	Новая CP
Remove CP	Устранить CP
Coopetition programm listing:	Листинг взаимодействующих программ
New Statement	Введение новой команды
Edit Statement	Редактирование команды
Delete Statement	Стирание команды
return to start page	возвращение на начальную страницу

Перевод текста рисунка 17, стр. 18

Program Queue	Очередь программ
Virtual Network node	Узел виртуальной сети
Instance name	Имя экземпляра
Program Translator	Переводчик программы
Job Queue	Очередь работ
Program Interpreter	Переводчик программы

Перевод текста рисунка 19, стр. 19