Архитектура системы компьютерной телефонии SmartServer

А.В.Корявко
http://oasis.apmath.spbu.ru/~kav/

А.А.Тихонов
talex@tepkom.ru

Санкт-Петербургский государственный университет
198504, Университетский пр., 28
Санкт-Петербург, Россия

Аннотация

Сегодня на рынке присутствует множество систем компьютерной телефонии. Хорошо справляясь с типовыми задачами, они не предоставляют пользователю возможность легко создать свой собственный, уникальный сервис.

В работе описана архитектура системы компьютерной телефонии, которая лишена этого недостатка. Наиболее подробно рассмотрены пользовательские интерфейсы и внутреннее представление бизнес-логики.

Введение

В последние несколько лет наблюдается непрерывное вторжение компьютерных технологий в различные сферы человеческой деятельности. В частности, одним из проявлений этого процесса в области телекоммуникаций является бурное развитие компьютерной телефонии.

Системы компьютерной телефонии призваны, с одной стороны, взять на себя те задачи, которые раньше выполнялись человеком, например, в справочных системах. С другой стороны, появилось множество новых сервисов, присущих только компьютерной телефонии, например голосовая почта.

Однако существующие на данный момент системы компьютерной телефонии не лишены недостатков. Самым существен-

ным из них является неуниверсальность предлагаемых решений, их жесткая привязка к конкретным схемам работы.

В данной работе описывается архитектура системы компьютерной телефонии, разрабатываемой в департаменте телефонии ЗАО "Ланит-Терком". Статья построена следующим образом: в следующем разделе дано краткое введение в предметную область и обзор уже существующих систем, затем описана архитектура предложенной универсальной системы компьютерной телефонии, а в конце обозначены пути ее дальнейшего развития.

1 Обзор технологий

Главным компонентом системы компьютерной телефонии является *центр обработки вызовов* (call center, contact center). Центр обработки вызовов — это программно-аппаратный комплекс, предназначенный для автоматической и полуавтоматической обработки телефонных звонков. Сфера применения центров обработки вызовов лежит там, где основная деятельность заключается в автоматической или полуавтоматической обработке звонков. В качестве примера можно привести справочные службы, центры опроса общественного мнения, отделы продаж и поддержки крупных торговых организаций и пр.

Основной задачей центра обработки вызовов является эффективная обработка потока телефонных вызовов в соответствии с бизнес-процессами компании, пользующейся услугами центра. Прежде всего сюда входит технология автоматизации интерактивных взаимодействий клиентов с центром обработки вызовов и оптимизация процесса общения клиента с оператором (агентом) в тех случаях, когда сценарий обработки звонка требует его участия.

Поскольку затраты на создание и поддержку системы компьютерной телефонии велики, а многим организациям ее услуги необходимы только эпизодически, появилось такое явление, как *внешний центр обработки вызовов* (outsourcing call center), который поддерживается сторонней компанией, у которой фирмы-пользователи центра покупают "виртуальные офисы". При этом выгодно использовать ресурсы (специальное оборудование, операторов) совместно, и такая система является более общим вариантом центра обработки вызовов. Таким образом, можно вы-

делить четыре роли, которые участвуют в функционировании системы компьютерной телефонии.

1. *Разработчик* — организация, которая разработала систему, имеет возможность ее модифицировать почти произвольным образом и продает ее “Администратору”.

2. *Администратор* — организация, осуществляющая эксплуатацию системы, ее администрирование и продажу услуг системы “Пользователям”.

3. *Пользователь* — организация, осуществляющая эксплуатацию отдельного сервиса системы, имеет возможность настройки этого сервиса.

4. *Клиент* — организация или частное лицо, потребляющее сервис, предоставляемый “Пользователем”.

С точки зрения разработки системы можно выявить ряд достаточно самостоятельных задач, а именно:

- построение телефонного сервера;
- разработка баз данных и доступа к ним как для хранения информации, относящейся непосредственно к системе, так и для клиентов, пользующихся услугами центра;
- разработка модели описания сценариев, а также систем их генерации и исполнения;
- реализация некоторого универсального способа задания клиентами сценариев обработки относящихся к ним звонков и многое другое.

Далее мы рассмотрим ряд промышленных систем компьютерной телефонии, их возможности и архитектуру, базируясь на обзоре, данном в работе [2]. К сожалению, центры обработки вызовов обычно являются коммерческими продуктами, а потому документация по ним довольно скудна и часто ограничивается документами рекламного характера. Более подробное введение в компьютерную телефонию можно найти в [1].

1.1 Definity

Ядром решений для операторского центра AVAYA Communications является УПАТС (учрежденческо-производственная АТС) типа Definity ECS. Компания AVAYA[1] предлагает три пакета программного обеспечения, которые позволяют создать центры с разной функциональностью:

- Call Center Basic для создания простых центров с базовым набором функций;
- Call Center Deluxe, имеющий подсистему гибкой маршрутизации вызовов;
- Call Center Elite, в который добавлены функции маршрутизации вызовов с учетом квалификации операторов.

Особенностью данной системы является использование гибких методов маршрутизации, получивших общее название Call Vectoring. Если обычная система обеспечивает распределение вызовов между операторами одной группы, то данная система позволяет задать гибкую маршрутизацию вызовов к разным операторским группам на каждом шаге. Схема обслуживания состоит из 32 шагов, которые задаются администратором. На каждом шаге можно работать с тремя группами операторов.

Для распределения нагрузки между операторами используются два основных алгоритма — Most-Idle Agent и Least Occupied Agent. В первом случае вызов будет передан тому оператору, который дольше всего оставался свободен. Во втором случае учитывается, сколько времени работал оператор в течение дня, и звонок будет передан наименее занятому оператору из всех свободных.

Одна из функций маршрутизации в Definity ECS, называемая выбором оптимального маршрута, позволяет при отсутствии свободных операторов оценить время ожидания обслуживания в разных группах и выбрать ту из них, время ожидания в которой будет наименьшим.

Система содержит средства для воспроизведения и записи звуковых сообщений, умеет синтезировать речь. Имеется возможность оставить сообщение оператору с просьбой связаться с абонентом.

[1] http://www.avaya.com/

Для поддержки функций компьютерно-телефонной интеграции центр на базе УПАТС Definity имеет в своем составе сервер компьютерной телефонии CentreVu Computer Telephony Server, который обеспечивает взаимодействие с внешними приложениями. Сервер подключается к Definity ECS через специальную плату, обеспечивающую связь через локальную сеть на основе протокола IP (далее IP-сеть) по протоколу ASAI. На сервере CentreVu Computer Telephony устанавливается программное обеспечение, поддерживающее большинство открытых API: TSAPI, JTAPI, TAPI, CallVisor PC. Клиентская часть сервера может быть реализована для различных платформ, например Windows, UnixWare, Solaris.

Центр поддерживает возможность приема и обработки звонков по IP-сетям. Для этого в структуре системы предусмотрен сервер Internet Call Manager, который, работая вместе с сервером компьютерно-телефонной интеграции, обеспечивает возможность приема речевых вызовов из сети с использованием технологии Voice Over IP. Кроме того, поддерживается обработка сообщений, поступающих по электронной почте, по факсу, а также в режиме обмена текстовыми сообщениями.

Рабочее место оператора оборудовано компьютером, подключенным к серверу и базе данных, и телефонным аппаратом, подключенным к телефонной станции.

1.2 NEAX 2400/7400

Центр NEAX 2400/7400 корпорации NEC[2] имеет блочную конструкцию, которая позволяет базовой 64-портовой системе связи быть расширенной посредством дополнительных модулей по 64 порта до емкости в 512 портов. Система имеет полностью не блокируемую архитектуру и 32-разрядный процессор, которые обеспечивают необходимую гибкость и производительность.

Основными возможностями системы являются:

- минимизация расходов на связь — возможность распознавать и использовать самые дешевые из имеющихся каналов;
- поддержка Primary и Basic Rate ISDN интерфейсов;
- автоматическое распределение вызовов;

[2] http://www.nec.com

- наличие цифровой консоли оператора;
- речевое приветствие;
- переадресация входящих вызовов;
- составление отчетов;
- поддержка групп пользователей;
- предоставление специальных возможностей для гостиниц и больниц.

1.3 ПРОТЕЙ-РВ

Контакт-центр ПРОТЕЙ-РВ научно-технического центра ПРОТЕЙ[3] реализован на базе интеллектуальной платформы ПРОТЕЙ. Система имеет распределенную архитектуру, позволяющую в определенных пределах наращивать ее емкость и функциональные возможности. Обеспечивается прием вызовов как по обычным телефонным линиям, так и из вызовов по IP-сетям.

Система поддерживает несколько групп операторов, в каждой из которых имеется своя очередь звонков, ожидающих обслуживания. Система маршрутизации ПРОТЕЙ-РВ обеспечивает возможность обслуживать входящие вызовы в ответном и предответном состоянии, направлять вызовы в ту или иную группу в зависимости от ряда параметров, заданных администратором системы, например номера абонента, времени суток, числа вызовов и др. Во время ожидания обслуживания оператором абоненту может сообщаться различная информация, например его номер в очереди и ориентировочное время ожидания. Для распределения нагрузки между операторами реализованы три алгоритма — циклическое распределение, выбор оператора с наибольшим временем бездействия и выбор наименее занятого оператора.

В ПРОТЕЙ-РВ интегрирована система Interactive Voice Response, которая позволяет во время ожидания не только обеспечить абонента необходимой информацией и подсказками, но и строить многоэтапные диалоги, поддерживая меню произвольной структуры с навигацией путем набора цифр в тоновом режиме (DTMF). В ходе такого диалога с системой абонент может

[3] http://www.protei.ru

получить часть интересующей его информации без задействования оператора.

В системе ПРОТЕЙ-РВ предусмотрена система упреждающего набора номера. Функции этой подсистемы требуются, когда в процессе работы системы создаются исходящие соединения. В этом случае, в зависимости от назначения центра, либо сама система генерирует список оповещения абонентов, либо этот список формируется персоналом. Система автоматически производит вызовы по сформированным спискам оповещения, определяет состояние линии, после чего связывает ответившего абонента с оператором.

В ходе разговора оператор может обратиться к базе данных за дополнительной информацией, может произвести исходящий звонок, отключить абонента, переадресовать абонента другой группе операторов, а также записать разговор.

Операторские места оборудуются компьютерами с аппаратурой для записи и воспроизведения звука. Компьютеры связаны с сервером системы через IP-сеть, по которой и передается звук. Также к этой сети подключена база данных.

1.4 Simposium

Операторские центры Simposium компании Nortel Networks[4] используют в качестве коммутационного ядра АТС типа Meridian-1, к которой по IP-сети подключается сервер, реализующий прикладные функции системы распределения вызовов. При выборе оператора для обслуживания абонента учитываются знания и специализация операторов. Кроме того, на выбор оператора может повлиять то, кто звонит, на какой номер он звонит, что ответил абонент на заданные ему вопросы. Система сохраняет информацию о предыдущих результатах работы с этим номером в базе данных и оператор получает ее при приеме вызова.

В центр также встроена система автоматизированного предоставления справок по часто задаваемым вопросам. Центр поддерживает прием заявок по электронной почте. Операторский интерфейс может быть настроен на работу с несколькими базами данных.

[4] http://www.nortelnetworks.com

1.5 LanHello PBX

LanHello PBX[5] компании Ланит является коммуникационным сервером, обеспечивающим построение современного телефонного офиса с интеграцией телефонных, факсимильных и вычислительных ресурсов. При этом коммуникационный сервер становится частью локальной сети организации и обеспечивает взаимодействие телефонного и почтового сервиса с рабочими местами сотрудников и базами данных.

LanHello PBX использует в качестве аппаратной платформы оборудование фирмы Dialogic, поддерживает информационно-логическое сопряжение со встречными станциями как по аналоговым, так и по цифровым соединительным линиям (в соответствии с протоколами R2, SS7), и ориентирована на ОС Windows NT. Разработана оригинальная надстройка над Windows NT, позволяющая обеспечить эффективную диспетчеризацию событий от драйверов аппаратных средств, запуск и управление реализацией соответствующих компонент ПО при асинхронном режиме выполнения стандартных (библиотечных) функций прикладного уровня.

Операторские места подключаются к серверу и базе данных через IP-сеть, которая используется для передачи данных. Для передачи голоса используется либо отдельная телефонная линия, либо IP-сеть.

Система поддерживает возможность донабора номера в тональном режиме, переадресации абонента на другой номер, записи телефонных разговоров, организации конференций до 32-х абонентов одновременно, создания голосовых меню любой сложности, предоставляет неограниченное число почтовых ящиков, обеспечивает автоматическое оповещение о поступивших сообщениях на пейджер, факс или по телефону, прием факсимильных сообщений операторами.

Итак, несмотря на массу дополнительной функциональности, предоставляемой отдельными системами, в качестве общего недостатка можно выделить следующий момент: все варианты манипуляций с вызовом предопределены заранее. Нет возможности создавать гибкие сценарии по обслуживанию звонков на каждый входящий номер индивидуально.

[5] http://www.lanit.ru/comptelephone_contact.html

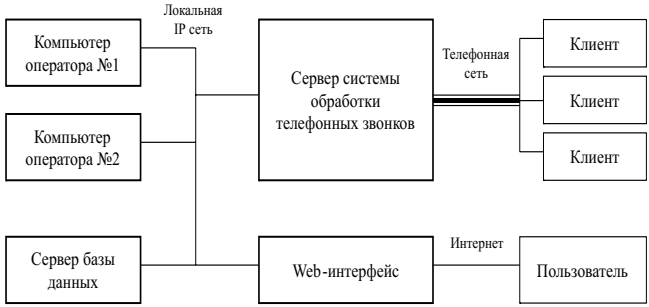

Рис. 1: Общая схема системы компьютерной телефонии

2 Архитектура системы SmartServer

В этом разделе описывается решение, частично избавляющее от изложенных выше недостатков.

2.1 Общее описание

Система состоит из четырех основных компонент — центра обработки вызовов, операторских приложений, базы данных и Web-интерфейса для доступа к базе данных. Ее общая схема приведена на рисунке 1.

Главной компонентой, как уже было сказано, является центр обработки вызовов. Далее для краткости будем называть центр обработки вызовов просто сервером, так как это соответствует его роли в рамках данной системы. Его схема приведена на рисунке 2. При построении сервера была выбрана многопоточная модель, соответственно модули, которым выделяется собственный поток управления, выделены двойной рамкой. HTTP-сервер, обеспечивающий работу Web-интерфейса, и сервер базы данных исполняются отдельно, возможно, удаленно, и не являются составной частью сервера. Общие схемы использования модулей обозначены стрелками, при этом пунктирные стрелки обозначают посылку сообщения, механизм сообщений обеспечивает асинхронность взаимодействия.

Рассмотрим кратко назначение модулей.

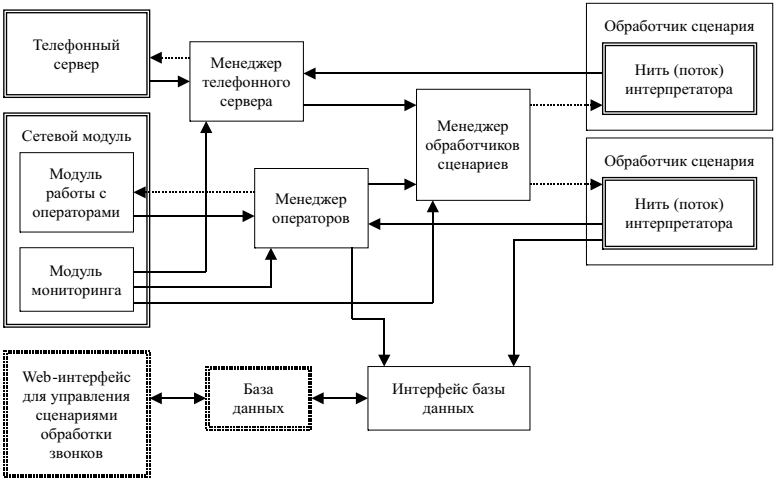

Рис. 2: Общая схема центра обработки вызовов

2.1.1 Телефонный сервер

Телефонный сервер является важным элементом сервера. Он работает непосредственно с телефонным оборудованием и выполняет простейшую обработку сообщений от него, тем самым абстрагируя все остальные модули от конкретного специализированного оборудования и его API. Это важно, поскольку различные производители оборудования, такие, как Dialogic, Aculab, Cisco (из отечественных можно упомянуть Агат-РТ (http://www.agatrt.ru/), Спецсвязь 2001 (http://www.granit-k.ru/)), используют различные API, так как несмотря на наличие стандартов, например, TAPI, эти стандарты слишком бедны и не позволяют использовать возможности компьютерной телефонии в полной мере. В настоящее время в прототипе системы используется плата собственной разработки, обеспечивающая передачу данных по 4-м трактам E1 (120 телефонных линий) и превосходящая аналоги по соотношению цена/качество.

2.1.2 Менеджер телефонного сервера

Все модули системы общаются с телефонным сервером через диспетчер. Диспетчер отвечает за механизм работы виртуальных устройств. Виртуальные устройства позволяют избежать коллизий при работе множества нитей со множеством устройств, уменьшить поток сообщений и частично защитить от случайного сбоя.

2.1.3 Менеджер операторов

Менеджер операторов предоставляет всей системе единый доступ к операторам. Его отличительной особенностью является передача звука и остальных данных по разным каналам, что обусловлено повышенными требованиями к каналу передачи звука. На данный момент каналы передачи реализованы одинаково, но для предотвращения *эффекта дрожания фазы* (jitter) разработана система буферизации звуковых пакетов.

Также в нашей системе реализована возможность определять для операторов некие абстрактные свойства, на основе которых осуществляется выбор оператора по критериям, задаваемым создателем сервиса. Более подробно эта тема раскрыта в [3].

2.1.4 Сетевой модуль

Основной задачей модуля является периодическая обработка сетевых соединений. В описываемой системе ему отведена незначительная роль.

2.1.5 Интерфейс базы данных

Этот модуль, по аналогии с телефонным сервером, служит для абстрагирования системы от конкретного типа сервера базы данных.

2.1.6 Менеджер обработчиков сценариев

Данный модуль, наряду с телефонным сервером, занимает важное место в системе. Он предоставляет сценарию простой и удобный синхронный интерфейс управления системой, скрывая

сложную асинхронную систему обмена сообщениями и распределения ресурсов, благодаря чему сценарий задается как последовательный набор операций в удобном для человека виде. Обработчик сценариев, в свою очередь, занимается интерпретацией текста сценария.

Поскольку одной из главных задач, поставленных при разработке системы, была универсальность, способ описания сценариев и его возможности подробно описаны в следующем разделе.

2.2 Внутреннее представление бизнес-логики

Чтобы не ограничивать возможности системы, по крайней мере на уровне внутреннего представления бизнес-логики, для ее представления было решено использовать алгоритмически полный язык. Таким образом, сценарий работы для конкретного сервиса задается как программа на этом языке. Так же можно создавать вспомогательные средства: наборы тестов, систему самодиагностики и т.п.

Какие требования предъявляются к кандидатам на роль языка описания бизнес-логики? В первую очередь, он должен гармонично сочетать относительно высокую производительность с удобством описания на нем достаточно сложных алгоритмов. Таким требованиям удовлетворяют, в первую очередь, различные скриптовые языки. Кроме того, интерпретатор языка должен легко встраиваться в систему, как с точки зрения интеграции самого интерпретатора в систему, так и с точки зрения реализации специфической среды исполнения для скриптов (имеется в виду необходимость создания встроенных в интерпретатор процедур для управления телефонным сервером из скрипта). Наконец, предпочтительно не создавать свой собственный скриптовый язык, а использовать уже готовый, известный потенциальным пользователям. Всем этим требованиям удовлетворяет язык Tcl [6]. Для него существует эффективный интерпретатор с открытыми исходными кодами на языке C, кроме того, возможна компиляция Tcl-скрипта в байт-код, что повышает производительность при многократном исполнении скрипта.

Однако использование интерпретатора Tcl заставило изменить реализацию многопоточности в центре обработки вызовов. Традиционно при создании ПО для АТС применяется невытес-

няющая многозадачность, когда существует один поток, который периодически передает управление разным модулям и ждет, когда они его вернут. Соответственно, на модули накладываются жесткие временные ограничения. Однако адаптировать интерпретатор Tcl под такие требования и реализовать в нем возможность одновременного исполнения нескольких скриптов в рамках одной нити трудно.

Многопоточная модель, когда для каждого скрипта запускается отдельный экземпляр интерпретатора, избавляет от этой проблемы, позволяя использовать стандартные библиотеки и интегрировать существующие интерпретаторы.

Но множество нитей требует больших накладных расходов. Однако наблюдения показали, что наибольшее время скрипт будет проводить в процессе исполнения команд управления телефонным сервером. Следовательно, на время исполнения этих команд нить интерпретатора можно усыплять, а исполнение всех команд проводить в рамках единственного потока управления телефонного сервера. Так достигается необходимый компромисс между простотой реализации и скоростью работы.

2.3 Внешние интерфейсы

Важнейшей задачей, вставшей при проектировании внешних интерфейсов, вновь оказалась задача представления сценариев. После выбора в качестве языка описания сценариев универсального языка Tcl возникла необходимость определить, каким именно образом сложный сценарий будет представлен для пользователя или администратора системы. Были рассмотрены три варианта интерфейса:

1. представление для пользователя совпадает с языком описания сценариев;

2. для каждого конкретного сервиса создается собственный сценарий и дружественный к пользователю мастер по его настройке;

3. промежуточный вариант, предоставляющий пользователю набор заранее заданных примитивов для создания из них произвольных комбинаций.

Задача технически осложняется тем, что интерфейс должен позволять работу через Web с использованием в качестве клиента обычного Web-броузера.

Далее рассмотрим эти варианты подробно.

2.3.1 Прозрачный интерфейс

Этот интерфейс просто предоставляет пользователю (администратору) прямой доступ к внутреннему представлению бизнес-логики, т.е. к скрипту на языке Tcl, ассоциированному с данным сервисом. Пользователь имеет возможность полностью изменять его. Настройка сервиса также осуществляется правкой скрипта.

Этот метод имеет следующие достоинства:

- отсутствие ограничений в использовании возможностей системы, полная универсальность;
- простота реализации

и недостатки:

- пользователю необходимо освоить язык скриптов и среду его исполнения в системе компьютерной телефонии;
- затруднительно гарантировать безопасность выполнения скрипта по отношению к базе данных, другим пользователям и системе в целом;
- семантика сценария скрывается в синтаксисе языка, что затрудняет его сопровождение.

Приведем пример создания простейшего сервиса с помощью этого интерфейса. Данный сервис называется “Справка” и предоставляет клиенту возможность прослушать звуковое сообщение, оставленное пользователем. Для определения сервиса пользователь пишет скрипт на языке Tcl примерно следующего содержания (рис. 3).

Однако такое решение кажется слишком слабым для современной системы компьютерной телефонии, поэтому оно использовалось только на начальной стадии разработки.

Рис. 3: Задание сервиса “Справка” с помощью прозрачного интерфейса

2.3.2 Мастера

Другой крайностью является почти полный отказ от универсальности в пользу простоты и надежности. Разработчик или специально подготовленый администратор реализует Tcl-скрипты для наиболее массовых сценариев, создает для каждого уникальный Web-интерфейс, а пользователю остается лишь настройка сценария под конкретные нужды в рамках, предоставляемых Web-интерфейсом.

Для примера из предыдущего раздела интерфейс выглядит следующим образом (рис. 4).

При этом созданный разработчиком скрипт может ничем не отличаться от приведенного в предыдущем разделе, однако создатель гарантирует его правильность и безопасность.

Поскольку такой подход привлекателен для большинства пользователей, а архитектура системы позволяет разработчикам эффективно создавать новые сценарии по мере возникновения новых потребностей у пользователей, на данный момент в прототипе используется именно он.

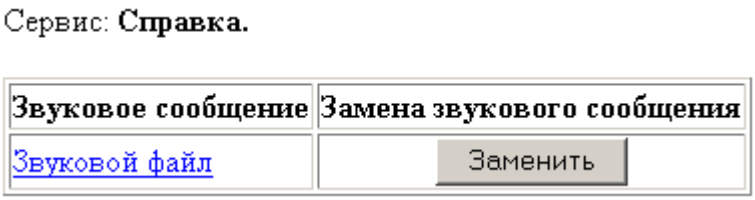

Рис. 4: Задание сервиса “Справка” с помощью мастера

2.3.3 Набор примитивов

Наконец, была разработана схема, в какой-то мере сочетающая универсальность прозрачного интерфейса и надежность и дружественность мастеров. Она позволяет пользователям самим создавать сценарии почти произвольной степени сложности, при этом им достаточно овладеть лишь небольшим количеством технической информации и не нужно заботиться о безопасности работы созданного сценария.

Базовым для этой модели является понятие *примитива* (action) — логически неделимого этапа взаимодействия системы с клиентом (позднее были добавлены вспомогательные примитивы, соответствующие некоторым конструкциям языков императивного программирования). Например, проигрывание клиенту звукового файла с последующей возможностью оставить ответ есть примитив “Автоответчик”(“Auto answer”). Набор этих взаимодействий (пока их 6) определен разработчиком заранее, но может быть относительно легко им расширен.

Примитив имеет параметры, которые пользователь может просматривать и менять. Они разделяются на входные и выходные. Например, для примитива “Автоответчик” к первым относятся файл приветствия и флаг, разрешающий принимать ответы, а ко вторым — эти ответы, оставленные позвонившими клиентами. Примитивы способны передавать управление друг другу, например, существует примитив, который обеспечивает ветвление в зависимости от того, какую цифру клиент донаби-

рает тональным набором. Один из примитивов является главным, т.е. именно с него начинается выполнение сценария.

Таким образом, набор примитивов представляет собой ориентированный граф, вершинами которого являются примитивы, а дуга обозначает возможность передачи управления от одного примитива к другому. Выбрав точку входа (главный примитив), пользователь определяет некоторый сценарий.

Конечно, построение этого графа для пользователя труднее, чем использование мастеров, но благодаря выделению типично "телефонных" операций оно существенно проще, чем написание скриптов на Tcl.

Реализация подобной схемы также требует создания достаточно сложного Web-интерфейса и разработки алгоритма генерации Tcl-скрипта по графу примитивов.

Задача генерации скриптов была решена следующим образом: для каждого типа примитивов был создан образец Tcl-кода (*проекция*), в которую при генерации подставляются параметры конкретного примитива. Каждая проекция представляет собой отдельную процедуру, что позволяет вызывать один примитив из другого. Также существует каркас скрипта, к которому при генерации добавляются эти процедуры. Очевидно, что этот алгоритм линеен относительно количества примитивов.

Важен вопрос, в какой момент осуществлять генерацию скрипта. Генерация по необходимости, т.е. в то время, когда приходит телефонный звонок, может негативно сказаться на производительности системы в этот ответственный момент. Явное указание от пользователя обновить скрипт неудобно. Наилучшим решением является установка триггеров в базе данных, которые обновляют скрипт после каждого изменения примитива, но при этом также увеличивается нагрузка на систему.

На рисунке 5 изображен общий интерфейс, в котором реализован пример сервиса "Справка".

Скрипт, сгенерированный по данному сценарию, будет иметь следующий вид:

```
# Automatically generated script for scenario 'Справка'

# Projection for action 'Проиграть сообщение'
proc Action123 { } {
    uplevel {
```

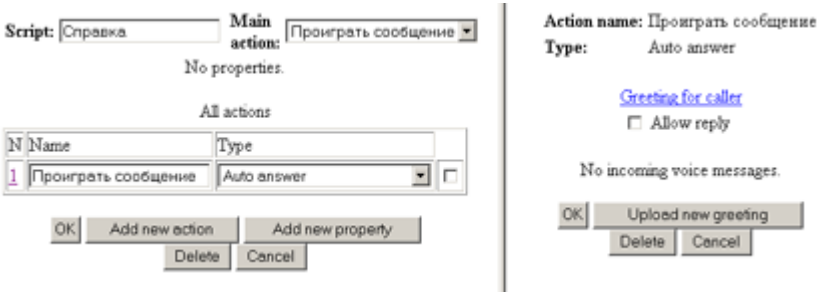

Рис. 5: Задание сервиса “Справка” с помощью примитивов

```
        # Load file from database into temporary memory file
        set afile [ DBLoadFile "67657" "$ScriptID" ]
        # Play file on incoming line
        PlayFile "$MainLineDevice" "$afile"
        # Delete memory file
        DeleteAudioFile "$afile"
    }
}

# Start section
Action123
OnHook "$MainLineDevice"
```

Наиболее целесообразным кажется применение примитивов совместно с мастерами, хотя в настоящее время эта возможность еще не реализована.

3 Перспективы развития

Системы компьютерной телефонии представляют собой молодое явление, поэтому еще не до конца ясны пределы их применения. Однако уже сейчас можно выделить несколько направлений, в которых возможно дальнейшее развитие описанной системы.

Возможно экстенсивное развитие, которое заключается в добавлении новых сценариев и мастеров, в создании новых примитивов и расширении функциональности старых.

Более интересна задача увеличения масштабируемости пользовательских интерфейсов. Когда сервис становится очень гро-

моздким, например, предоставляя расписание движения сотен маршрутов автобусов, анализ большого количества примитивов становится затруднительным. Использование мастеров облегчает работу, но все равно целесообразно разработать новые подходы к управлению сервисами.

Также стоит предоставить возможность интеграции сервисов компьютерной телефонии с другими службами. Например, при проведении телефонного опроса в какой-либо телепередаче необходим доступ к результатам опроса из системы нелинейного видеомонтажа для вывода на телеэкран хода опроса в режиме реального времени.

Не менее важным является поддержка международных стандартов для разработки звуковых приложений, таких, как VoiceXML [5], что позволяет системе исполнять функции интеллектуального шлюза между телефонной сетью и VoiceXML-сервером. Для этих целей можно использовать OpenVXI [4]. Также интерес представляют системы генерации и распознавания речи.

На фоне постоянного увеличения функциональности не менее важно тщательно следить за запасом производительности системы, перекладывая по возможности нагрузку с центра обработки звонков, который по существу является системой реального времени, на другие части системы.

Заключение

Основная проблема, с которой сталкивается исследователь в области систем компьютерной телефонии — отсутствие эффективных способов создания нестандартных сценариев работы сервиса. В данной работе предложена архитектура системы компьютерной телефонии, благодаря которой можно преодолеть этот недостаток. На данный момент система на основе предложенной архитектуры реализована не полностью, однако имеющиеся наработки свидетельствуют о правильности выбранного подхода.

Список литературы

[1] Гольдштейн Б.С., Фрейнкман В.А. Call-центры и компьютерная телефония. — СПб.: BHV, 2002. — 370 с.

[2] Дыдычкин Д.А. Разработка и реализация системы автоматической обработки телефонных звонков: Дипломная работа. — СПб.: 2002 — 43 с.

[3] Тихонов А.А. Разработка методов реализации сервисов с использованием оператора для центров обработки вызовов: Дипломная работа. — СПб.: 2002 — 25 с.

[4] Eberman B., Carter J., Meyer D., Goddeau D. Building VoiceXML Browsers with OpenVXI // Proc. of the 11-th International World Wide Web Conference. — 2002. — P. 713-717.

[5] McGlashan S., Burnett D., Danielsen P. e.a. Voice Extensible Markup Language (VoiceXML) Version 2.0. — 2002. — http://www.w3.org/TR/2001/WD-voicexml20-20011023.

[6] TCL/TK Reference Manual. — Ajuba Solutions, 2000.