Новая концепция XML-ориентированного языка разметки диалогов

Зобнин Д.С.,

Старцева Н.С.

В данной статье рассматривается опыт создания новой концепции языка разметки диалогов, которая могла бы успешно использоваться для создания систем с голосовым диалоговым интерфейсом на основе систем синтеза и распознавания речи.

• распознавание речи • синтез речи • голосовой интерфейс • VoiceXML • сценарий диалога.

A new concept of dialog markup language is proposed which can be successfully used for creation of systems with voice dialog interfaces based on speech synthesis and recognition technologies.

• speech recognition • speech synthesis • voice interface • VoiceXML • dialog markup.

Введение

Основной проблемой при использовании систем распознавания речи в контексте взаимодействия посредством телефонной связи является отсутствие подробной характеристики акустического сигнала [1]. Одним из способов решения проблемы низкого качества сигнала является использование систем распознавания, основанных на грамматиках. В такие системы уже заложены основные акустические модели, характерные для конкретных языков. При этом набор слов, по которым происходит анализ, строго ограничен грамматикой и задается разработчиком. Повышение качества распознавания в данном случае будет достигаться уменьшением грамматики, но при этом встает вопрос гибкости сценария диалога [2].

Гибкий сценарий диалога подразумевает эмуляцию естественного диалога конечного пользователя с системой, что достигается путем заложения в грамматику и в сценарий диалога как можно большего числа различных вариантов высказываний, по которым будет выполняться распознавание. Естественно, ограничение множества вариантов возможно благодаря анализу предметной области, для которой создается сценарий. Поэтому задачу составления сценариев диалогов чаще всего делегируют лингвистам и специалистам в конкретных предметных областях.

Возможность привлечения экспертов и лингвистов при составлении сценариев диалогов позволяет создавать приложения с голосовым интерфейсом, которые будут обладать достаточной гибкостью для внедрения в различные сферы жизнедеятельности человека. Для этого необходимо наличие специ-

альных инструментальных средств построения сценариев, обеспечивающих абстрагирование от низкоуровневых деталей устройства систем синтеза и распознавания речи. Одно из таких средств и рассматривается в данной статье.

Стандарт VoiceXML

Структура и архитектура VoiceXML

С появлением систем синтеза и распознавания речи на рынке возможность их интеграции с телефонией стала первостепенной проблемой бизнес-специалистов. Возникло немало трудностей в связи с ограничениями стандартов телефонии. Но, так или иначе, проблемы решались, и возникала задача создания гибкого механизма составления сценариев диалогов. Сценарии должны были отражать бизнес-логику самого диалога конечного пользователя с различным кругом систем посредством телефонии.

Результатом совместной работы корпораций AT&T Corporation, IBM, Lucent и Motorola стал стандарт VoiceXML 0.9, представленный в марте 1999 г. Результат развития описательных возможностей стандарта представлен актуальным VoiceXML 2.1 и пока еще не выпущенным VoiceXML 3.0.

Стандарт VoiceXML опирается на стек других стандартов организации W3C:

- SRGS (Speech Recognition Grammar Specification) — стандарт описания грамматик, используемый для обозначения множества шаблонов и предложений, которые ожидаются к распознаванию [3].
- SISR (Semantic Interpretation for Speech Recognition) — стандарт, используемый совместно с SRGS для указания семантических результатов распознавания по грамматике [4].
- ECMAScript — стандарт Javascript, система событий которого заложена в основу ядра VoiceXML. Также ECMAScript используется повсеместно и в других стандартах для описания различных конструкций [5].
- SSML (Speech Synthesis Markup Language) стандарт, используемый для разметки синтезируемых фраз [6].
- PLS — стандарт, используемый для описания произношения слов. Применяется совместно с распознавателем и синтезатором речи [7].
- CCXML (Call Control eXtensible Markup Language) – стандарт описания сценария телефонных операций, таких как соединение, перенаправление, разрыв и прочие [8].
- MSML (Media Server Markup Language) [9] и MSCML (Media Server Control Markup Language) [10] — стандарты, описывающие работу с медиаданными.

Структура VoiceXML опирается на модель MVC (Model View Controller), используемую в DFP (Data Flow Presentation) [11], рис. 1.

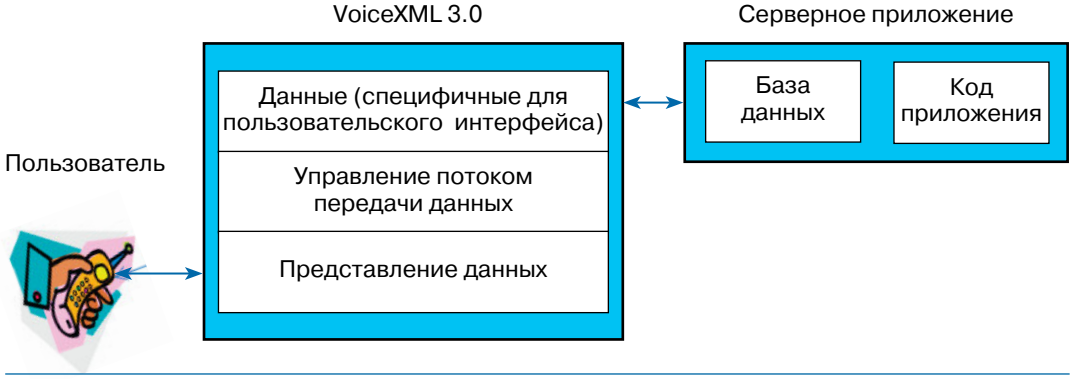

Рис. 1. Архитектура DFP

DFP является основой для построения систем, использующих VoiceXML. При этом доступ к данным во время сессии звонка осуществляется посредством стандарта SCXML (State Chart XML) [12], а сами данные должны быть представлены в виде XML. Обработка данных и манипулирование ими производится посредством ECMAScript и моделей DOM [13] или XPath [14].

Модельное представление сценария диалога на VoiceXML

Вдохновленные идеями голосового браузера и возможностями применения речевых технологий совместно с web, разработчики стандарта VoiceXML постарались максимально точно повторить схему взаимодействия конечного пользователя с web-страницей, заменив ручной ввод на голосовое управление.

Как и в HTML, в VoiceXML структурной единицей обмена данными является форма. Она является ключевым компонентом VoiceXML и служит для получения некоторой информации от пользователя и последующей обработки этой информации (перехода в другую форму, исполнения скрипта, и т.д.) Весь процесс распознавания речи происходит в рамках формы, поэтому действия, применяемые при отсутствии распознания или низкой точности результата, объявляются в локальном контексте.

За задачу передачи данных между формами отвечает контекст исполнения, содержащий массив локальных и глобальных переменных. Контроль области видимости переменных (на уровне всего документа или только в пределах формы) обеспечивает некоторую инкапсуляцию форм друг от друга.

Ветвление сценария на уровне VoiceXML заключается в переключении от одних форм к другим [15]. В рамках формы система взаимодействует с пользователем, обмениваясь информацией, но, к сожалению, стандарт не предполагает выполнения каких-либо сложных операций на уровне сценария (получения писем электронной почты, отправки смс-сообщений и т.д.). Множество доступных действий ограничивается возможностями ECMAScript. При этом предполагается, что все остальные действия могут быть выполнены за счет взаимодействия с сервером.

Недостатки VoiceXML

Несмотря на все плюсы, у стандарта VoiceXML есть и явные недостатки. Во времена создания стандарта в процессе непосредственного написания сценариев диалогов не предполагалось участие технически неподкованных специалистов, а степень точности систем синтеза и распознавания речи не позволяла создавать крупные коммерческих продукты, способные обеспечивать гибкий диалог с конечным пользователем. Именно поэтому в основу стандарта описания сценария работы диалога было положено множество исключительно «инженерных» технологий, например, доступ к данным посредством ECMAScript или описание запросов к данным через XPath.

На сегодняшний день крупные коммерческие продукты, использующие технологии синтеза и распознавания в телефонии, обладают незаурядными по сложности сценариями диалогов, а составляющие их специалисты (лингвисты и эксперты в предметных областях) в качестве способа формализации используют, в основном, блок-схемы. Изначальная направленность VoiceXML непосредственно на инженеров-телефонистов и отсутствие нативных средств перевода блок-схем бизнес-логики в сценарии диалогов создают немалые трудности при организации процесса разработки прило-

жений для телефонии. Часть обязанностей лингвистов и экспертов перекладывается на плечи подразделений по разработке ПО. Отсутствие прямых возможностей внесения изменений лингвистами в сценарий диалога на VoiceXML (в основном, по причине сложности освоения стека стандартов) приводит к возникновению временных затрат как при разработке продукта, так и при его дальнейшей поддержке.

Отсутствие встроенных возможностей автоматической генерации грамматик без применения сложной логики взаимодействия с базами данных и адаптерами посредством ECMAScript, значительно усложняет задачу создания гибкого диалога, вынуждает прибегать к обработке частей диалога на стороне сервера и частично выносить бизнес-логику за пределы описания сценария. Такой подход к разработке голосовых приложений рано или поздно приводит к путанице в схеме работы приложения и значительно увеличивает время, необходимое для изменения логики взаимодействия с пользователем. В связи с вышесказанным, очевидна необходимость создания новой концепции языка разметки диалогов, которая обладала бы более широкими возможностями, по сравнению с VoiceXML.

Новая концепция языка описания сценариев диалогов

Графовая модель сценария диалога

При моделировании сущностей реального мира находят применение два основных подхода. Один из них основан на идентификации сущностей посредством выделения присущего им характерного набора атрибутов (и дальнейшей классификации по типам, исходя из одинаковости наборов характеристических атрибутов). Другой метод заключается в том, что моделируемые сущности оказываются подходящей содержательной интерпретацией объектов некоторой формальной системы.

При моделировании некоторых проистекающих во времени процессов взаимодействия конечный пользователь-система наиболее оптимальным становится описание непосредственно реакций системы на определенные события, порожденные конечным пользователем. Наиболее интуитивно-понятной представляется модель сценария диалога в виде графа состояний, где вершины — действия системы в реальном времени, а переходы определяют следующее действие. При этом выбор перехода определяется в зависимости от реакции конечного пользователя.

Сценарием диалога в общем смысле будем называть связный ориентированный граф S:

$$S := (V,U). \quad (1)$$

Граф S — это упорядоченная пара из V — непустого множества вершин или узлов и U — множества упорядоченных пар различных вершин, называемых дугами или ориентированными ребрами. В нашем случае дуги будем называть связями между вершинами, и изначально будем предполагать, что определение не допускает наличия «висящих» связей (связей с одним или двумя пустыми концами). Под графом диалога будем понимать граф сценария диалога.

Спецификой используемой в системе модели сценария диалога является то, что непосредственно переходы между вершинами графа не задаются пользователем при моделировании. Модель узла графа предполагает наличие входных и выходных данных, необходимых для работы узла. С точки зрения специалиста, занимающегося составлением графа, переходы между узлами представляются связями типа предок-потомок, а выбор потомка, в который перейдет на следующем шаге диалог, определяется логикой работы самого узла.

Таким образом, некоторые манипуляции над стандартными представлениями о переходах между узлами графа диалога позволили обойтись при моделировании одной сущностью — узлом графа и отношениями между узлами типа предок-потомок. Сужение бази-

са, на котором основана модель, сокращает время освоения самой модели и в итоге позволяет более лаконично описывать граф.

Опыт работы с графом диалога показал, что целесообразно представление графа в виде модулей — это облегчает процесс анализа дублированных частей сценария. Этот факт обуславливает применение модели ветви сценария, описанной ниже.

Модель ветви сценария диалога

Применение метода декомпозиции к графовой модели сценария привело к созданию модели ветви сценария диалога. Под ветвью сценария диалога (далее ветвь диалога) будем понимать совокупность множества внутренних узлов ветви V_I, множества внешних узлов ветви сценария V_E, множества внутренних связей U_I ветви диалога (множество связей, соединяющих внутренние узлы ветви), множества внешних связей U_E ветви (множество связей, соединяющих внешний узел ветви с внутренним узлом ветви):

$$B := (V_I, V_E, U_I, U_E). \quad (2)$$

В рамках данной модели сценарий диалога может быть представлен как множество ветвей, т.е.:

$$S := B_r = \{b \mid b \in B_r\}. \quad (3)$$

Ветвь представляется как:

$$b = (v_I, v_E, u_I, u_E), \quad (4)$$

где:

- v_I — множество внутренних узлов ветви:

$$v_I = \{v \mid \forall \notin b_1 : b_1 \in B_r \,\&\, b_1 \neq b \rightarrow v \in V(b) \,\&\, v \notin V(b_1)\}; \quad (5)$$

- v_E — множество внешних узлов ветви:

$$v_E = \{v \mid v \notin v_I\}; \quad (6)$$

- u_I — множество внутренних связей ветви:

$$u_I = \{(v_1, v_2) \mid v_1 \in v_I \,\&\, v_2 \in v_I\}; \quad (7)$$

- u_E — множество внешних связей ветви:

$$u_E = \{(v_1, v_2) \mid v_1 \in v_1 \,\&\, v_2 \in v_E \vee v_1 \in v_E \,\&\, v_2 \in v_1\}. \quad (8)$$

- Под оператором $U_I(x)$ будем понимать оператор, применимый к ветвям диалога и возвращающий множество внутренних связей ветви.
- Под оператором $U_E(x)$ будем понимать оператор, применимый к ветвям диалога и возвращающий множество внешних связей ветви.
- Под оператором $V(x)$ будем понимать оператор, применимый к ветвям диалога и возвращающий множество внутренних узлов ветви.

При подобном подходе полноценный граф сценария будет представляться как:

$$S := (V, U). \quad (9)$$

Под множеством вершин графа будем понимать объединение по всем ветвям множеств внутренних вершин ветвей:

$$V = U_{b \in B_r} V(b), \quad (10)$$

а под множеством связей будем понимать объединение по всем ветвям множеств внутренних и внешних связей графа:

$$U = U_{b \in B_r} U_I(b) \cup U_E(b). \quad (11)$$

Изначальным условием модели была связность графа сценария. При введении нового понятия ветви необходимо определить некоторые ограничения:

• Для любого внутреннего узла ветви сценария существует связь, соединяющая этот узел с внешним или внутренним узлом ветви:

$$\forall v \in V(b) \rightarrow \exists(v_1, v_2)\ \ U_I(b) \cup U_E(B)\colon v = v_1 \vee v = v_2, \tag{12}$$

введение такого ограничения позволяет устранить висячие узлы.

• Для любого внутреннего узла ветви найдется путь внутри ветви, включающий в себя рассматриваемый узел, который бы заканчивался или начинался связью с внешним узлом:

$$\forall v \in V(b) \rightarrow \exists s\colon v \in s\ \&\ (\text{tail}(s) \in U_E(s) \vee \text{head(s)} \in U_E), \tag{13}$$

где tail(*s*) — связь, являющаяся конечной для пути s в графе, head(s) — связь, являющаяся начальной в пути *s*.

Это ограничение не допускает составления ветвей, не связанных с другими ветвями.

Кроме того, так как определение связи между узлами не позволяет создание висящих связей, то модель представления сценария как множества ветвей будет эквивалентна модели полноценного связного графа сценария, и в дальнейшем любые манипуляции над одной моделью могут быть также применены и к другой модели.

Таким образом, определение новой модели графа сценария через введение понятия ветви и дополнительных ограничений не сужает список возможностей манипулирования сценарием и одновременно позволяет декомпозировать его на отдельные логически связанные части — ветви. Возможности декомпозиции позволят создавать ветви, отвечающие определенной цели, состоящие из логически связанных узлов, что на уровне редактирования и дальнейшей модификации уменьшит количество дублированных частей и позволит упростить понимание сложных сценариев.

Понятие переменной сценария диалога

В целях обмена данными между узлами диалога и манипулированием этими данными было введено понятие переменной сценария диалога. Переменная представляется как тройка (имя, тип, значение):

$$v := (n, t, \text{val}), \tag{14}$$

где:

- *n* — строка с именем переменной;
- *t* — тип переменной;
- val — значение переменной определенного типа.

При анализе требований к данным, манипуляция которыми будет осуществляться в сценарии диалога, было предложено ограничить множество возможных типов четырьмя категориями. Т.е. множество типов T представимо как:

$$T := \{\text{string, dict, list, dictlist}\}, \tag{15}$$

где:

- string — строковый тип;
- dict := "[{key, val}]" — тип словаря, определяется как упорядоченное множество пар строк ключ-значение;
- lict := "[val]" — тип списка, определяется как упорядоченное множество строковых значений;
- dictlist := "[{key, val}]" — тип списка словарей, определяется как упорядоченное множество упорядоченных множеств строковых пар ключ-значение.

Наличие такой системы типизации позволяет описывать элементарные объекты типа строк и списков строк, а простые объекты – как словари. При этом множество словарей, устроенных одинаковым образом, организует список простых объектов.

Модель контекста исполнения диалога

В процессе моделирования сценария диалога было решено использовать отдельную сущность, посредством которой будет осуществляться обмен данными между узлами диалога. В качестве такой сущности было решено использовать понятие контекста диалога.

Контекст диалога представляет собой хранилище общей информации, доступной в ходе диалога конечного пользователя с системой. Контекст представляется как объединение следующих сущностей:

- Граф диалога. Доступность к графу из узлов через контекст иногда необходима для самоанализа графа диалога. Например, при валидации узла диалога.
- Хранилище переменных — особый контейнер, осуществляющий хранение, изменение переменных, предоставляющий доступ к значениям переменных.
- Множество обработчиков событий, вызов которых осуществляется при возникновении особых ситуаций по ходу диалога.

За счет использования контекста как места объединения данных и структуры сценария становится возможным процесс отладки сценария при помощи анализа состояния контекста при переключении от одного узла к другому.

Модель узла графа как функции

Ключевым элементом сценария диалога является модель узла. Узел диалога осуществляет манипуляцию данными в сценарии, определяет выполняемые действия, контролирует поток дальнейшего исполнения сценария.

В общем виде узел сценария можно условно представить в виде функции, принимающей в качестве входного параметра пару из двух сущностей (<команда узлу>, <контекст сценария>), а возвращающей тройку (<следующий узел>, <следующая команда>, <новый контекст диалога>). Такая система организации аргументов и возвращаемого значения необходима для предоставления следующих возможностей:

- обмен данными между узлами за счет использования контекста;
- осуществление прерывания стандартного хода сценария;
- обмен данными с использующей движок системой.

Обмен данными между узлами осуществляется за счет доступа к общему хранилищу переменных. При работе узла возможно как считывание узлом переменных, так и запись или изменение значений новых переменных.

Наличие общего хранилища переменных обусловлено необходимостью записи сразу нескольких значений во время работы узла и считывания множества переменных. При этом парность функции может быть неопределенной.

При подобной схеме работы с внутренними данными сценария появляется возможность статического анализа диалога на предмет наличия ошибок использования переменных как в рамках ветви, так и в рамках диалога в целом.

Иерархия типов узлов сценария диалога

Узлы классифицируются в соответствии с выполняемыми ими функциями. Множество типов узлов графа представляется иерархией типов, для которой

определены понятия наследования. В предлагаемой модели предполагается, что основными будут следующие типы:

- корневой узел;
- узел двоичного ветвления;
- узел операции;
- узел условного ветвления.

Тип корневого узла

Данный тип узла определяет точку входа в сценарий диалога во множестве ветвей сценария. Узел обладает тремя потомками:

- первый потомок — узел по умолчанию, т.е. узел в который будет совершен переход при инициализации диалога;
- второй потомок — узел, в который будет совершен переход в случае возникновения ошибки в процессе работы сценария;
- третий потомок — узел обработки стадии завершения сценария диалога.

Тип узла двоичного ветвления

Данный тип узла графа диалога предназначен для представления узлов, ветвление потомков которых осуществляется в двух направлениях, т.е. узлов, разветвляющих ход диалога на два подграфа. Такой тип ветвления часто используется в ситуациях ожидания некоторого события, которое возможно и не произойдет. В таких случаях ход диалога будет происходить по какому-то заранее заданному направлению в графе, а в случае возникновения определенного события управление должно быть отдано в другой узел графа.

Тип узла операции

Тип узла операции предназначен для определения узлов, выполняющих какое-либо действие. Такие узлы обладают ровно одним потомком и предназначены для изменения контекста или изменения состояния диалога (состояние простоя, отмена ожидания результатов распознавания и т.д.).

Тип узла условного ветвления

Данный тип предназначен для обеспечения возможности определения множества реакций на действия конечного пользователя, сравнения различных числовых, строковых значений и календарных дат со значениями переменных контекста.

XML-подобный язык разметки графовой модели сценария диалога

Для поддержки возможности описания графовой модели сценариев диалога был разработан язык разметки DAML (Dialogue Application Markup Language), основанный на XAML (eXtensible Application Markup Language). DAML имеет ряд отличий от стандартного XML, используемого в VoiceXML, что дает дополнительные возможности при описании сложных объектов.

Особенности языка (дополнительные детали XAML):

1. Сложные свойства: возможность установки атрибутов элемента непосредственным их объявлением в объемлющем элементе (такая возможность необходима для возможности установки в атрибут сложного объекта). Например, в узле tts потомок узла задается ссылкой как сложным объектом, но при этом по сути лишь устанавливается значение свойства объекта:

<tts id='555'>

<tts.*children*>
<noderef>556</noderef>
</tts.*children*>
<const>\speed=60 \volume=60 Повторите ваш СНИЛС, по цифрам.</const>
</tts>.

2. Возможность установки сложных свойств как атрибутов внутристрочно. Предыдущий пример можно переписать следующим образом:

<tts children='556' id='555' phrase='\speed=60 \volume=60 Повторите ваш СНИЛС, по цифрам.' />.

Достоинство такого подхода заключается в том, что часто используемые XML-конструкции проще представить в виде строк и задавать как атрибуты. А при необходимости установки более сложных типов объектов удобно пользоваться возможностями XAML.

Способы объявления и методы манипулирования переменными и константами

Для работы с различными константными значениями было решено использовать текстовый формат JSON. Такое решение было принято в связи с малой долей корреляции специальных символов, используемых в форматах XAML и JSON при описании данных. Таким образом, для имеющихся четырех типов переменных (строковый, список, словарь, список словарей) в формате JSON уже существуют все необходимые конструкции:

1. Значение строкового типа задается как обычная строка:

<const type='string'>string typed value</const>

При установке в качестве значения свойства объекта может использоваться напрямую в атрибуте XML элемента:

<definedgr name='snils2' />.

1. Значение типа список задается как разделенные запятыми строки, заключенные в квадратные кавычки:

<const type='list'>["val1", "val2"]</const>.

2. Значение типа словарь задается как множество пар строк ключ-значение, где пара разделена символом ':', и ключ стоит слева от символа-разделителя. Каждая пара отделена от другой символом запятой, а множество пар ограничено символами фигурных кавычек:

<const type='dict'>{ "key1":"val1", "key2":"val2" }</const>.

3. Значение типа список словарей задается аналогично спискам, только вместо строк используются представления словарей:

<const type='dict'>[{ "key1":"val1", "key2":"val2" }, { "key1":"val3", "key2":"val4" }]
</const>.

4. Ссылки на переменные отличаются от обычных значений свойств префиксом «@» и могут задаваться через значения атрибутов:

<stringt source='@address' />.

5. Для узлов, имеющих одного потомка, в рамках внутренних связей ветви поддерживается упрощенный синтаксис:

<sleep time='15000' id='10' children='11' />

Пример приложения на новом языке разметки

Простейшее приложение, состоящее из одной ветви, в новом языке представления сценария диалога может быть задано следующим образом:

```
<root id='1'>
<root.children>
<noderef>8</noderef>
<noderef>5</noderef>
<noderef>4</noderef>
</root.children>
</root>
<asr result='service' id='8'>
<asr.children>
<noderef id='9' />
<noderef id='11' />
</asr.children>
<dynamicgr>
<const type='dict' value='{ "operator":"Оператор", "manager":"Менеджер" }' />
</dynamicgr>
</asr>
<sleep time='5000' children='10' id='9' />
<asrstop children='12' id='10' />
<switch source='service' id='11'>
<case value='operator' node='12' />
<case value='manager' node='2' />
</switch>
<tts id='2' children='13' phrase='Соединяем с менеджером' />
<connect num='' id='13'>
<connect.children>
<noderef id='4' />
<noderef id='14' />
</connect.children>
</connect>
<tts children='6' phrase='Соединяем с оператором' id='12' />
<connect id='6' num='+7499*******'>
<connect.children>
<noderef>4</noderef>
<noderef>7</noderef>
</connect.children>
</connect>
<tts id='7' children='4' phrase='К сожалению сейчас все операторы заняты, перезвоните позже' />
<tts id='14' children='4' phrase='К сожалению сейчас менеджер недоступен, перезвоните позже' />
<tts id='5' phrase='Произошла ошибка' children='4' />
<end id='4' />
```

Визуализация представленного графа сценария представлена на рис. 2, с. 40.

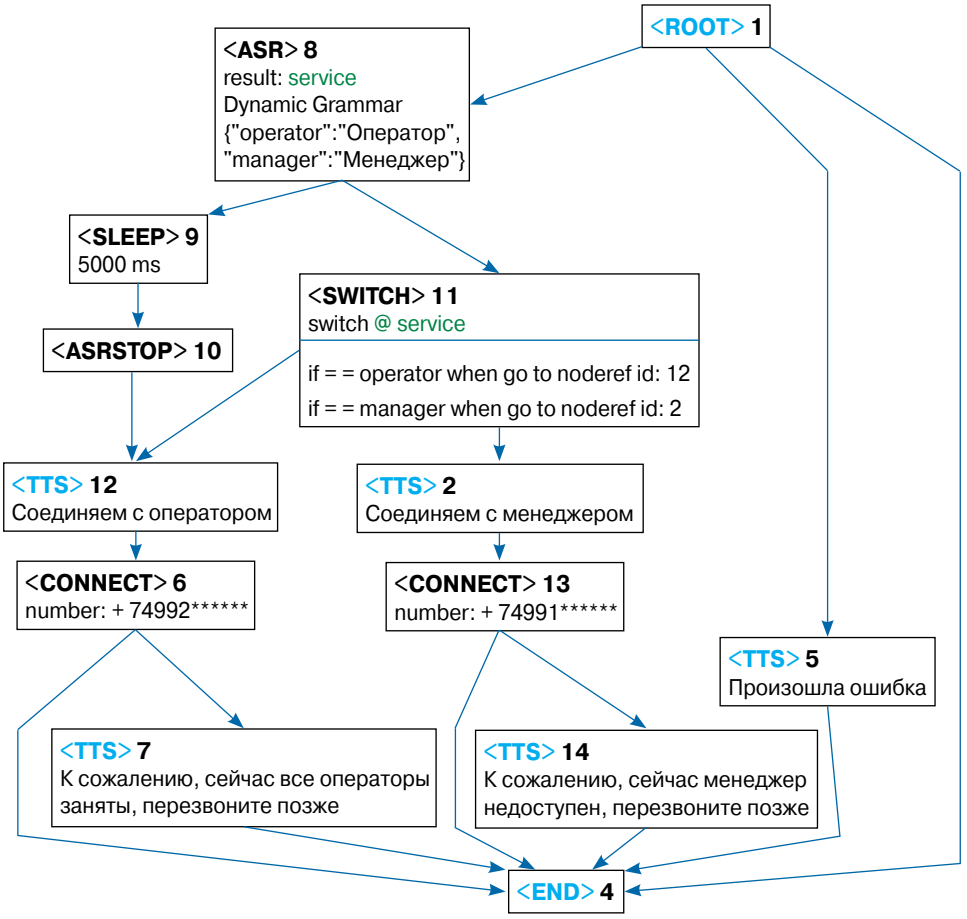

Рис. 2. Граф сценария диалога

Заключение

При создании систем, использующих технологию распознавании речи в качестве языка описания модели сценария диалога, чаще всего используется VoiceXML. Однако он обладает явными недостатками как на уровне концепции описания хода диалога, так и на уровне создания решений, основанных на нем.

В частности, VoiceXML не обладает нативными средствами перевода блок-схем бизнес-логики в сценарии диалогов, возможностями внесения изменений лингвистами в сценарий диалога, встроенных возможностей автоматической генерации грамматик без применения сложной логики взаимодействия с базами данных и адаптерами и т.д.

Совершенно очевидно, что необходимо создание нового языка описания сценариев диалога. В качестве новой концепции была предложена графовая модель сценария на основе ветвей, составленных из узлов, связанных связями предок-потомок. Для описания предложенной модели был создан язык представления диалога DXML, лаконично описывающий гибкие сценарии и обладающий расширенными, по сравнению с VoiceXML, возможностями. DXML успешно используется специалистами компании S2S Next при создании собственных голосовых решений.

Литература

1. *Пономарь М.О.* О допустимых пределах искажений электроакустических речевых сигналов, *Вестник Московского Государственного университета*. Т. 580. С. 265–271, 2010.

2. *Голунов В.И.* Современные проблемы в области распознавания речи., Auditech.Ltd, 15 марта 2013. [Электронный ресурс]. Режим доступа: http://auditech.ru/page/darkness.html, свободный. — Дата обращения: 20 июня 2013.

3. W3C, Speech Recognition Grammar Specification 1.0, 16 марта 2004. [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/speech-grammar/, свободный. — Дата обращения: 20 июня 2013.

4. W3C, Semantic Interpretation for Speech Recognition (SISR) Version 1.0, 5 апреля 2007. [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/semantic-interpretation/, свободный. — Дата обращения: 20 июня 2013.

5. E. International, International ECMA Script Language Specification 5.1 Edition, Ecma International, июнь 2011. [Электронный ресурс]. Режим доступа: http://www.ecma-international.org/ecma-262/5.1/, свободный. — Дата обращения: 20 июня 2013.

6. W3C, Speech Synthesis Markup Language (SSML) Version 1.0, 7 сентября 2004. [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/speech-synthesis/, свободный. — Дата обращения: 20 июня 2013.

7. W3C, Pronunciation Lexicon Specification (PLS) Version 1.0, W3C, 14 октября 2008. [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/pronunciation-lexicon/, свободный. — Дата обращения: 20 июня 2013.

8. W3C, Voice Browser Call Control: CCXML Version 1.0, W3C, 5 июля 2011. [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/ccxml/, свободный. — Дата обращения: 20 июня 2013.

9. W3C, Voice Extensible Markup Language (VoiceXML) 3.0, 10 декабря 2010. [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/voicexml30/, свободный. — Дата обращения: 20 июня 2013.

10. *Saleem A., Xin Y.* Media Server Markup Language (MSML), IETF, 2010.

11. *Van Dyke J., Burger E.* Media Server Control Markup Language (MSCML) and Protocol, IETF, 2006.

12. W3C, State Chart XML (SCXML): State Machine Notation for Control Abstraction, 26 апреля 2011. [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/2011/WD-scxml-20110426/, свободный. — Дата обращения: 20 июня 2013.

13. W3C, DOM4, W3C, 6 декабря 2012. [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/dom/, свободный. — Дата обращения: 20 июня 2013.

14. Microsoft, XPath Reference, Microsoft Corporation, 2 августа 2012. [Электронный ресурс]. Режим доступа: http://msdn.microsoft.com/en-us/library/ms256115.aspx, свободный. — Дата обращения: 20 июня 2013.

15. Microsoft, VoiceXML Scripting Elements and Subdialogs, Microsoft, 2012. [Электронный ресурс]. Режим доступа: http://msdn.microsoft.com/en-us/library/ff769496.aspx, свободный. — Дата обращения: 20 июня 2013.

Сведения об авторах

Зобнин Дмитрий Сергеевич —
разработчик, ООО «Голосовая Платформа», info@speechplatform.ru

Старцева Наталья Сергеевна —
ведущий лингвист, ООО «Голосовая Платформа», info@speechplatform.ru