УДК 004.42

Е.М. Кирьянов, К.Ю. Дрыгин

ПРИМЕНЕНИЕ ШАБЛОНОВ ПРОЕКТИРОВАНИЯ ДЛЯ УНИФИКАЦИИ ИНТЕРФЕЙСА ДОСТУПА К СИСТЕМЕ АВТОМАТИЧЕСКОГО РАСПОЗНАВАНИЯ РЕЧИ

Системы автоматического распознавания речи (ASR, Automatic Speech Recognition), поддерживающие русский язык, перестают быть экзотикой. Особенно хорошо этот факт заметен в телефонных приложениях: справочная МГТС [1], Российское представительство Microsoft, справочные ОАО «Уралсвязьинформ» и ОАО «Центртелеком» [2].

Однако используемые на российском рынке решения являются «полуфабрикатом». Например, программный интерфейс (API) наиболее популярной системы ScanSoft SpeechPearl включает в себя несколько десятков функций. Полное программное руководство [3] по ним занимает более сотни страниц, не считая примеров, среди которых нет ни одного, иллюстрирующего работу в режиме «клиент-сервер». На рисунке 1 в качестве примера приведена диаграмма состояний и переходов канала системы распознавания речи:

Упростить для пользователя процесс распознавания речи позволяет применение шаблона «Фасад», скрывающего сложное внутреннее устройство системы за простым интерфейсом. Реализация выглядит так: создается один или несколько классов, предоставляющих всю нужную конкретному пользователю или группе пользователей функциональность. Внутреннее устройство этих классов может быть сколь угодно сложным, однако для клиента их использование должно быть предельно простым.

В API SpeechPearl особую значимость имеют три функции, которые вызываются неоднократно и требуют довольно значительного времени для выполнения. Все они различаются по назначению и набору входных и выходных параметров; кроме того не исключена возможность добавления новых функций при расширении функциональности системы (например, для поддержки режима обучения). С

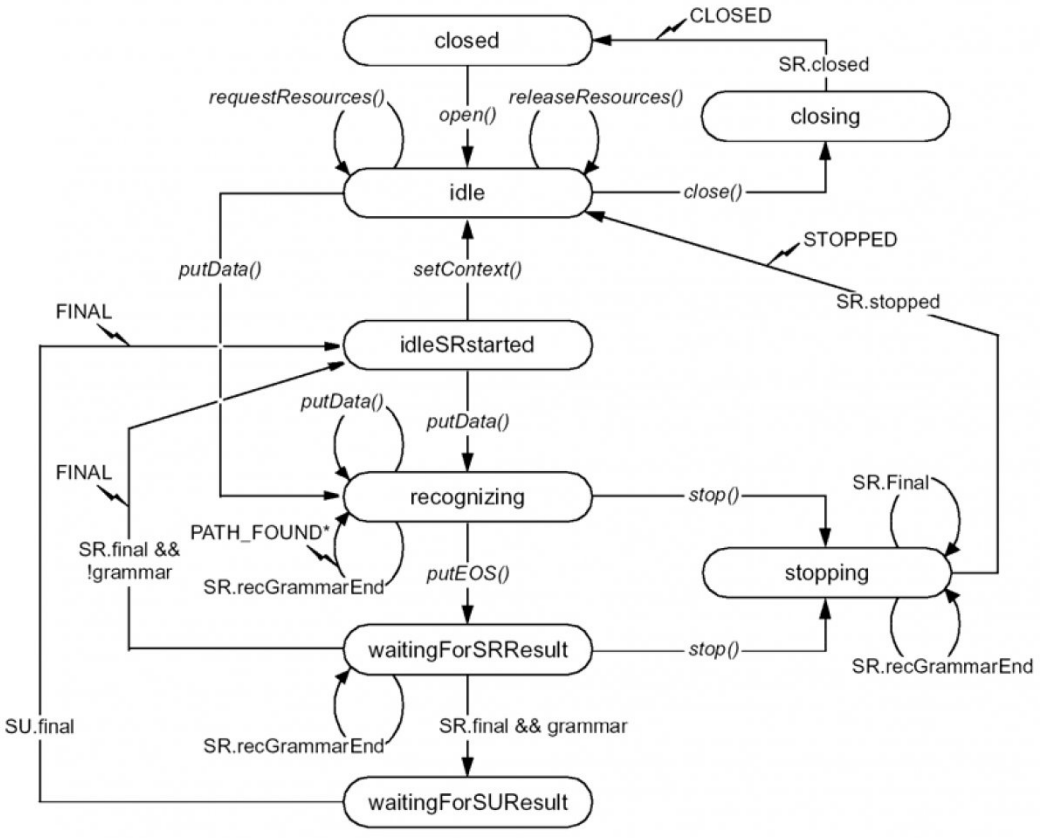

Рис 1. Диаграмма состояний ASR ScanSoft SpeechPearl

учетом всего вышеперечисленного разумным представляется помещение вызова каждой функции в отдельный класс и наделение этих классов одинаковым интерфейсом.

Обеспечить одинаковый интерфейс несложно – достаточно вынести его в некоторый базовый класс. Однако следует учитывать режим работы функции (синхронный или асинхронный), и перед вызовом синхронной функции выполнить некоторые дополнительные действия: проанализировать полученный набор параметров, дополнить его собственными значениями, затем создать новый поток и уже из его контекста выполнить обращение к функции. Структура этого алгоритма является общей для всех потомков базового класса, но реализация отдельных его шагов зависит от типа вызываемой функции. Чтобы избежать дублирования выполняющего одинаковые действия кода во всех производных от базового классах и дать им возможность изменять отдельные шаги алгоритма, не меняя общей его структуры, обычно применяется шаблон проектирования «Шаблонный метод»(«Template method»).

В реализации данного шаблона структуру алгоритма определяет базовый класс, а точнее, один из его методов. Описанная в таком «шаблонном» методе последовательность содержит действия двух разных видов:

- общие для всех потомков действия, которые могут быть описаны и реализованы как угодно, хотя для создания четкой и понятной структуры их лучше оформить в виде вызова соответствующих методов;
- действия, которые могут быть изменены потомками, должны быть представлены вызовом виртуальных методов.

В результате производные классы могут переопределить методы, реализующие методы второго вида – они будут выполнены, и на структуре алгоритма это не скажется.

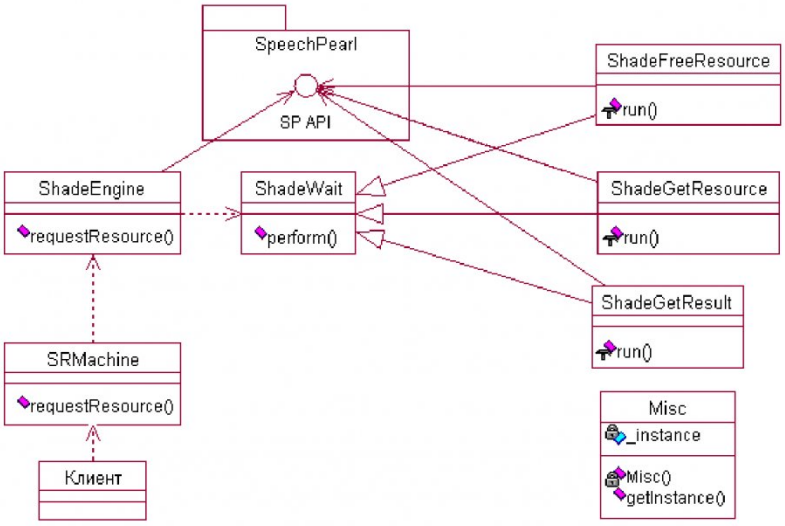

Рис. 2. Структура классов унифицированного интерфейса доступа к системе ASR

Еще один класс предназначен для хранения данных, к которым необходимо обеспечить глобальный доступ. Он также содержит набор вспомогательных методов, которые могут быть использованы как системой, так и пользователем. Глобальный доступ обеспечивает использование шаблона «Одиночка».

Структура решения в целом представлена на рис. 2.

На рисунке представлены следующие базовые классы.

- `SRMachine` – класс, с которым работает клиент. Представляет собой реализацию шаблона «Фасад».
- `ShadeEngine` – класс, позволяющий работать с отдельным ресурсом распознавания.
- `ShadeWait` – базовый класс преобразования синхронной функции в асинхронную, определяющий интерфейс потомков и алгоритм их работы. Реализует «шаблонный метод» (именно метод, а не весь шаблон).
- `ShadeGetResource`, `ShadeFreeResource` и `ShadeGetResult` – классы преобразования конкретных синхронных функций, переопределяющие отдельные шаги базового алгоритма.
- `Misc` – класс-«одиночка», содержащий данные и методы вспомогательного назначения.

Отдельно изображена система SpeechPearl с ее собственным программным интерфейсом.

Работа с системой начинается с ее инициализации: необходимо создать объект класса `SRMachine` и вызвать его метод `init`. Третий ее параметр (пользовательский обработчик сообщений) формально не является обязательным, но без него работа с системой практически не имеет смысла. Осуществить повторную инициализацию системы, задав произвольные параметры, можно в любой момент.

После получения нужного результата (все каналы были созданы) для начала распознавания следует запросить ресурс распознавания. Такой запрос выполняет метод `getResource`. В передаваемом ему наборе параметров следует указать имя запрашиваемого ресурса (группа `SHADE_REQUESTED_RESOURCE`). В случае успешного завершения в тот же набор параметров в группу `SHADE_ENGINE_INDEX` будет

записан идентификатор канала, с которым будет связан новый ресурс. Позднее, по окончании асинхронного выполнения, пользовательская функция-обработчик получит уведомление о его результате.

Как только подтверждение запроса ресурса будет получено, можно приступать к распознаванию. Передача системе аудиоданных осуществляется вызовом методов `putFile` и `putData`. Первый достаточно вызвать один раз. При использовании второго нужно после передачи всех аудиоданных вызвать его с параметрами (0,0) для начала распознавания.

Сообщение об окончании процесса распознавания речи передается пользовательской функции (тип сообщения `DS_SP_FINAL`). После его получения можно запросить результат в необработанном виде – как его возвращает SpeechPearl. Преобразование ответа к более удобной форме (`RecognitionResult`) занимает некоторое время. По окончании преобразования также вызывается функция-обработчик.

Распознавание с использованием запрошенного ресурса может осуществляться неоднократно. Тем не менее, не стоит оставлять его без работы – на сервере обычно устанавливается лимит времени, которое ресурс может оставаться занятым. Если по истечении этого лимита не было произведено ни одной операции распознавания, ресурс принудительно освобождается, а пользователю отправляется уведомление (тип сообщения `DS_SP_RM_SHUTDOWN`).

Освободить ресурс, который более не нужен, позволяет метод `releaseResource`. В передаваемом ему наборе параметров нужно установить идентификатор канала, с которым освобождаемый ресурс связан (группа `SHADE_ENGINE_INDEX`). Указанный метод является асинхронным. О результате завершения освобождения система уведомляет пользователя вызовом функции – обработчика.

Если ресурс нужно закрыть (после чего он не будет доступен до перезапуска сервера), следует воспользоваться методом `closeResource`.

Освобождение всех ресурсов происходит автоматически с уничтожением объекта. Если же пользователю необходимо сделать это самостоятельно, следует вызвать метод `init` с числом каналов, равным нулю.

Получить имя ресурса, связанного с конкретным каналом, или всех запрошенных ресурсов позволяют методы `getResourceName` и `getCurrentResources` соответственно.

Отдельное замечание по работе пользовательской функции: она вызывается по завершении любой асинхронной операции, но тип этой операции определяется разными путями. Стандартные асинхронные функции SpeechPearl могут быть идентифицированы по типу сообщения (см. `SP_ResultMessage::getEngineId`). Преобразованные в асинхронные функции записывают в набор параметров (`SP_ResultMessage::getParams`), в группу `SHADE_WAITER_INSCRIPTION` значение, соответствующее их назначению.

Таким образом, рассматриваемый интерфейс работы с системой распознавания речи ASR ScanSoft SpeechPearl упрощает доступ к ресурсам распознавания речи этой системы и преобразующего некоторые синхронные функции ее прикладного программного интерфейса в асинхронные. Более того, предложенное решение «отвязывает» пользовательскую программу, использующую ASR, от особенностей решения каждого конкретного производителя, и позволяет, в случае необходимости мигрировать на другую платформу.

СПИСОК ЛИТЕРАТУРЫ

1. SpeechPearl 8.0 Grammar Guide / Philips Speech Processing. Aachen, 2002. − 55 с.
2. http://www.mgts.ru/menu.html?ID_DOC=639&idm=706.
3. SpeechPearl 8.0 Reference Manual / Philips Speech Processing. Aachen, 2002. − 131 с.

Авторы статьи:

Кирьянов
Егор Михайлович
– аспирант кафедры вычислительной техники и информационных технологий

Дрыгин
Константин Юрьевич
– старший преподаватель кафедры вычислительной техники и информационных технологий