Выбор инструментальной среды разработки ПО СУ. Сравнительный анализ библиотек классов для разработки графических интерфейсов MFC и QT

Городко А.С.; Снисаренко С.В.
Кафедра систем управления
Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
e-mail: snisarenko@bsuir.by

***Аннотация*— Проведен сравнительный анализ библиотек классов для разработки графических интерфейсов MFC и QT.**

Ключевые слова: библиотеки классов, графический интерфейс, объектно – ориентированная архитектура.

Библиотеки для создания пользовательского интерфейса применяются в большом количестве операционных систем.

Целью статьи является проведение сравнительного анализа библиотек классов для разработки графических интерфейсов MFC и QT.

Microsoft Foundation Class (MFC) – это библиотека классов, графический инструментарий для операционных систем Windows [1].Это надстройка над интерфейсом прикладного программирования (API) win32. Предоставляемый инструментарием MFC API имеет смешанный C/C++ интерфейс.

QT – это кросс-платформенная библиотека, графический C++ инструментарий, который разрабатывается с 1994 года Trolltech. Он доступен для Windows (любой версии), UNIX (любой версии), MacOS X[1].

Концепция «документ-представление» (Document/View). Библиотека MFC требует от разработчика использования модели «документ-представление» и шаблонов. Но шаблоны имеют фиксированную структуру, и поэтому очень сложно расширить их возможности. Например, невозможно разделить область и отобразить два вида двух различных документов. Вторая проблема заключается в том, что шаблон создает вид, но не имеет к нему доступа. Все должно быть сделано документом, и это в некоторых случаях затрудняет разработку.

QT в свою очередь не ограничивает разработчика какой-либо моделью дизайна. Без каких – либо ограничений разработчик может использовать модель «документ-представление» если сочтет это нужным. Или не использовать ее.

Сравнение псевдообъектной и объектной архитектур. Главное отличие между Qt и MFC- в их дизайне. MFC – это библиотека, имеющая псевдообъектный дизайн.

Qt – это библиотека, построенная на объектно-ориентированной архитектуре. Её инструментарий имеет согласованное именование, наследование, организацию классов и методов. Используя однажды один из классов, разработчик легко сможет использовать и другие, так как они работают сходным образом[1].

Цикл сообщений. Работа MFC основана на механизме сообщений. К сожалению, непросто узнать, как их использовать, какую информацию они передают, когда они посылаются. Механизм сообщений усложняет отладку и чтение листингов. Работа QT базируется на механизме обратного выхода, основанном на передаче сигналов и их приёме слотами (Slots). Эта система является основным механизмом связи между объектами. Сигнал может передавать любое число аргументов. Необходимые сигналы подключаются к соответствующим слотам, так что в итоге программист всегда знает, что происходит. Число сигналов, передаваемых классом, обычно невелико (4-5), и все они очень хорошо документированы. Этот процесс находится под полным контролем. Механизм использования сигналов и слотов приближенно напоминает Java listener, но он более универсален [1-2].

Создание интерфейса. MFC не обеспечивает компоновку элементов управления внутри окна: это создает проблемы при желании сделать окно измеряемого размера.

Данная проблема приобретает большее значение для программ, интерфейс которых должен быть переведен на язык с более длинными словами или предложениями. Разработчик должен будет перекомпоновать программу для каждого такого языка.

Некоторые свойства могут быть откорректированы, а мастер класса позволяет легко создавать переменные и методы. Однако стоит заметить, что создать вручную цикл сообщений, функцию DDX (Dialog Data Exchange) или макрос IMPLAMENT_DYNCREATE будет достаточно сложно.

QT имеет гибкий механизм компоновки, который является очень простым. Для создания пользовательского графического интерфейса QT предлагает инструмент QT Designer. С помощью него можно откорректировать любые свойства используемых элементов управления.

Qt Designer позволяет делать то, что невозможно в MFC, например, создать список с предзаполненными полями или использовать различного вида вкладки (tab controls).

Уникод. Чтобы отобразить уникод в MFC, необходимо скомпилировать и собрать приложение со специальными опциями. Разработчик должен добавить макрос (_T) к каждой строке, используемой в программе, и сменить тип char на TCHAR. Большим минусом является то, что программа, откомпилированная с поддержкой уникода, не будет работать с библиотеками DDL, откомпилированными без его поддержки. Если при разработке использовались сторонние библиотеки DDL, это может принести серьезные неудобства.

Строки в Qt являются объектами класса QString, который изначально поддерживает уникод. Поэтому не придется изменять код или использовать какие-либо опции при компиляции и сборке. Класс QString достаточно функционален, даже если не заботится об уникоде.

Поэтому поддержка уникода достигается весьма просто.

Большое различие между строковым классом MFC CString и QString заключается в их дизайне. CString основан на типе char* с несколькими методами. Преимущество такого подхода в том, что везде, где требуется переменная типа char*, можно использовать член класса CString. Однако недостатком такого подхода является то, что можно изменить член char* класса CString без обновления самого объекта класса. Также возникают трудности при преобразовании в уникод.

QString, напротив, содержит уникод-версию строки и обеспечивает тип char* только когда требуется. API Qt в качестве строчных аргументов всегда требует QString, поэтому разработчику редко придется использовать тип char*. Класс QString также обладает некоторыми дополнительными возможностями, вроде совместного использования содержимого строки.

Локализация. MFC- программу можно локализировать. Для этого необходимо поместить каждую строку в строковую таблицу и везде в коде использовать функцию LoafString (IDENTIFIER). Затем необходимо поместить строковую таблицу в библиотеку DLL. Используя Visual Studio перевести строки в желаемый язык, преобразовать графические ресурсы (поэтому, что их текст не может быть помещен в строковую таблицу) и использовать эту библиотеку в программе достаточно сложно, переводчик не сможет сделать эту работу самостоятельно, ему потребуется помощь разработчика. Также могут возникнуть проблемы из-за фиксированного позиционирования элементов управления в MFC. Так как их расположение определялось длиной не переведенных строк, более длинные переведенные фразы будут накладываться. Если в последствии разработчик изменяет некоторые строки или добавит новые, он должен будет убедиться, что перевод был обновлен.

В Qt разработчик всего лишь передает строки функции tr (). Это очень удобно при разработке – разработчик может изменять строки непосредственно в его коде. Специальная программа Qt Linguist извлекает все требующие перевода строки и в удобном виде их отобразит. Ее интерфейс обеспечивает удобный перевод, возможность использования словаря, просмотр контекста строки, обнаружение конфликта клавиатурных сокращений обнаружение новых не переведенных или измененных строк. Эта программа может использоваться переводчиком без познаний в области разработки программ. Редактор Qt Linguist доступен по лицензии GPL, поэтому его можно модифицировать. Перевод сохраняется в формате XML, поэтому он может быть легко использован в различных целях. Задача добавления к программе нового перевода заключается в создании нового файла с помощью Qt Linguist [2].

Распространение. При распространении приложения, созданного с помощью MFC, разработчик должен использовать MFC-библиотеку, поставляемую вместе с Windows. При этом необходимо учитывать, что под одним и тем же названием, к примеру, MFC42.DLL, могут скрываться несколько различных версий этой библиотеки и в противном случае, ее обновить. Обновление библиотеки MFC может повлиять на работу многих приложений.

Названия библиотек Qt недвусмысленны (qt-mt303.dll), поэтому нет никакого риска, что установка библиотеки qt-mt303.dll. Также отсутствует проблема обновления системы в целом.

Вывод. Анализируя все вышеперечисленные возможности и недостатки двух библиотек можно сделать вывод, что библиотека классов MFC во многом уступает Qt и является менее гибкой. В то же время, следует сказать о высокой стоимости корпоративной версии Qt. Тем не менее, разработчики библиотеки Qt выпускают GPL версию библиотеки, что делает ее доступной учебным заведениям и программистам, разработчикам, которые создают программы с открытым исходным кодом.

[1] Э. Таненбаум. Современные операционные системы 3-е издание: Спб.: БХВ - Питер , 2011. – 1120 с.

[2] В.Давыдов. Visual C++. Разработка Windows-приложений с помощью MFC и API-функций. Спб.: БХВ - Питер , 2008. – 576 с.

[3] Ж. Бланшет, М. Саммерфилд. Qt 4. Программирование GUI на C++, 2-ое издание: М.: КУДИЦ-Пресс, 2008. – 718с