Автоматизация диагностического тестирования приложений

Системы Числового Программного Управления

Комаров Александр Витальевич, Евстафьева Светлана Владимировна

МГТУ «СТАНКИН», кафедра «Компьютерные Системы Управления»

komarov_alexander@mail.ru

Проблема автоматизации диагностического тестирования становится все более актуальной. Это связано с тем, что реализуется много новых различных технологий. Однако перед тем как внедрить их на производство, каждую технологию нужно протестировать. Сейчас создаются программы, упрощающие этот процесс с применением средств автоматизации.

На данный момент, автоматизированное тестирование программного обеспечения представляет собой часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

Не смотря на то, что с помощью ручных тестов можно найти много недостатков в программном приложении, это довольно долгий и трудоемкий процесс. Автоматизация заключается в процессе написания компьютерной программы автономно обрабатывающую информацию, которую в противном случае пришлось бы исследовать вручную. Когда исследования автоматизировались, они уже могут быть многократно запущены и проведены снова. Это решение является наиболее экономически выгодным для программных продуктов, которые нуждаются в постоянном техническом обслуживании, так как даже незначительные исправления, в течение всего срока эксплуатации, могут привести к краху всей системы в целом.

Один из способов создания тестов заключен в тестирование на основе модели. Варианты тестирования частично или целиком получаются из модели описывающей некоторые аспекты (чаще функциональные) тестируемого программного обеспечения. Такой способ актуален для систем Числового Программного Управления (ЧПУ), поскольку к приложению предъявляются требования долгого времени выполнения и ограниченного использования ресурсов. Данная модель показана на Рисунке 1.

Рисунок 1. Архитектура тестирования на основе модели

Целью работы[1] является создание кроссплатформенного приложения для автоматизации проведения тестирования процессов *системы Числового Программного Управления.* Среда проектирования – Microsoft Visual Studio 2010 с использованием средств Windows API.

Для достижения намеченных целей поставлены задачи:

1. Реализация кроссплатформенных функций для диагностики процессов приложений системы ЧПУ;
2. Разработка алгоритмов получения данных о состоянии и параметрах процесса;
3. Вывод данных в удобной форме.

Важнейшей частью операционной системы, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами. Процесс (или по-другому, задача) – абстракция, описывающая выполняющуюся программу. Для операционной системы процесс представляет собой единицу работы, заявку на потребление системных ресурсов. Наше приложение разрабатывается с целью считывать значения процессорных ресурсов системы ЧПУ и обрабатывать их в соответствие с поставленной задачей.

Разрабатываемое приложение пишется на современных высокоуровневых кроссплатформенных языках программирования C, C++. Для них есть компиляторы под различные платформы. В данном случае, оно предназначено для работы в операционных системах Linux и Windows одновременно. Архитектура операционной системы не влияет на работу приложения.

Кроссплатформенным (переносимой) называется программное обеспечение, функционирующее более чем на одной аппаратной платформе и/или операционной системе. Это довольно трудоемкая задача, учитывая, что различные операционные системы имеют разные интерфейсы прикладного программирования и API-интерфейсы (например, UNIX-системы используют различные API функции для прикладного программного обеспечения, в отличие от Windows систем). Другими словами, программа, написанная на популярном языке программирования, не всегда будет работать на операционных системах, поддерживающих его.
Ниже приведен код, в котором достигается самый простой способ получения кроссплатформенности:

```
void CCrossFunc::Func_bzero(void* s, size_t n)
{
#ifdef __linux__
  bzero(s, n);
#elif defined _WIN32
  ::ZeroMemory(s, n);
#endif
}
```

В коде, написанном на языке C++, создается кроссплатформенная функция Func_bzero, наследуемая от функции CCrossFunc и получающая необходимые параметры. При компиляции, обрабатывается та часть кода, в зависимости под какой платформой он компилируется. В среде Windows(98/ME/NT/2000/XP/Seven) компилятор обработает нам часть кода под _WIN32 этой функции, а другую часть – отбросит. Это очень удобное свойство также используется и в системах ЧПУ, которые реализованы, как и под семейство операционных систем Windows NT, так и под семейство UNIX-систем.

На рисунке 2 представлен общий принцип работы разрабатываемого приложения. Тестировщик запускает диагностическое приложение “Process Information”, основанное на кроссплатформенных функциях, и задает необходимые параметры тестирования. Оно, в свою очередь, обращается к набору программных интерфейсов Windows API (application programming interface). Используя базовые функции Windows API, приложение получает доступ к системным процессам, считывая необходимую информацию. Затем приложение выводит полученные данные в командную строку (или консоль) и, параллельно с этим, сохраняет информацию в XML-файле.

[1] Работа выполнена по Госконтракту №П978 на проведение НИР в рамках ФЦП "Научные и научно-педагогические кадры инновационной России" на 2009-2013 гг.

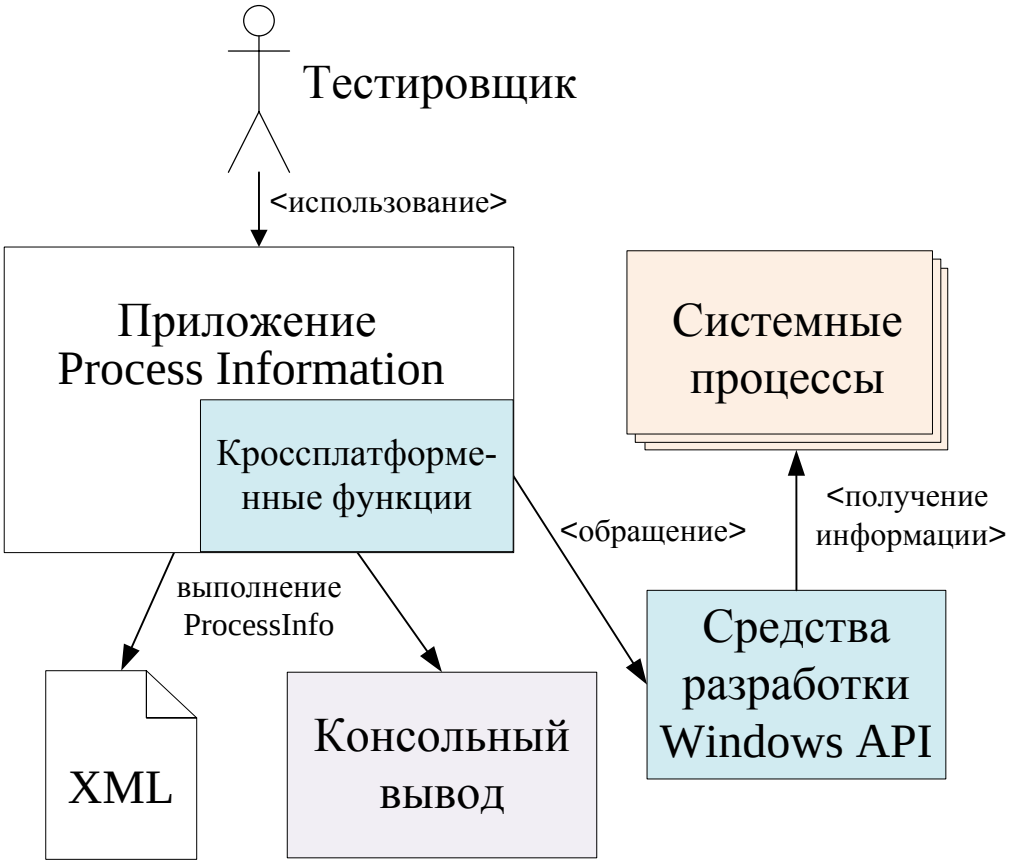

Рисунок 2. Общая блок-схема работы разрабатываемого приложения

На рисунке 3 представлена диаграмма прецедентов взаимодействия компонентов в работе приложения.

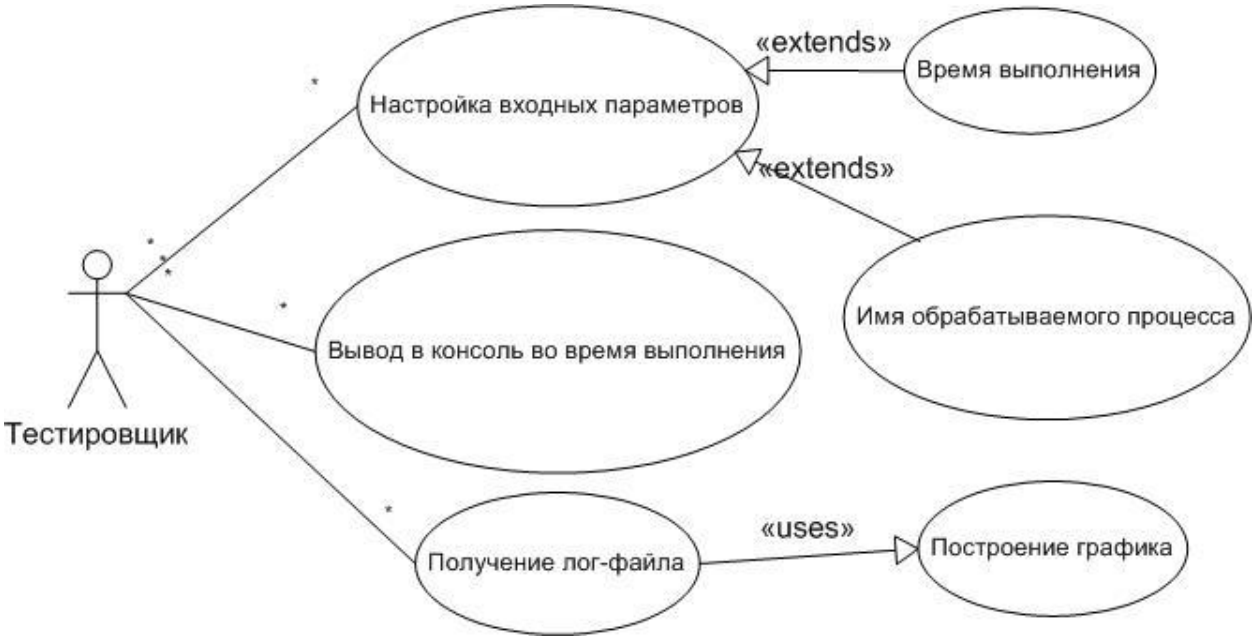

Рисунок 3. Диаграмма прецедентов разрабатываемого приложения

Приложение-тестер обеспечивает пользователя основными функциями доступа. Во время выполнения, программа постоянно отправляет данные в командную строку. Лог-файлы используются для построения графиков. Настройки можно конфигурировать с помощью С-файлов или вручную.

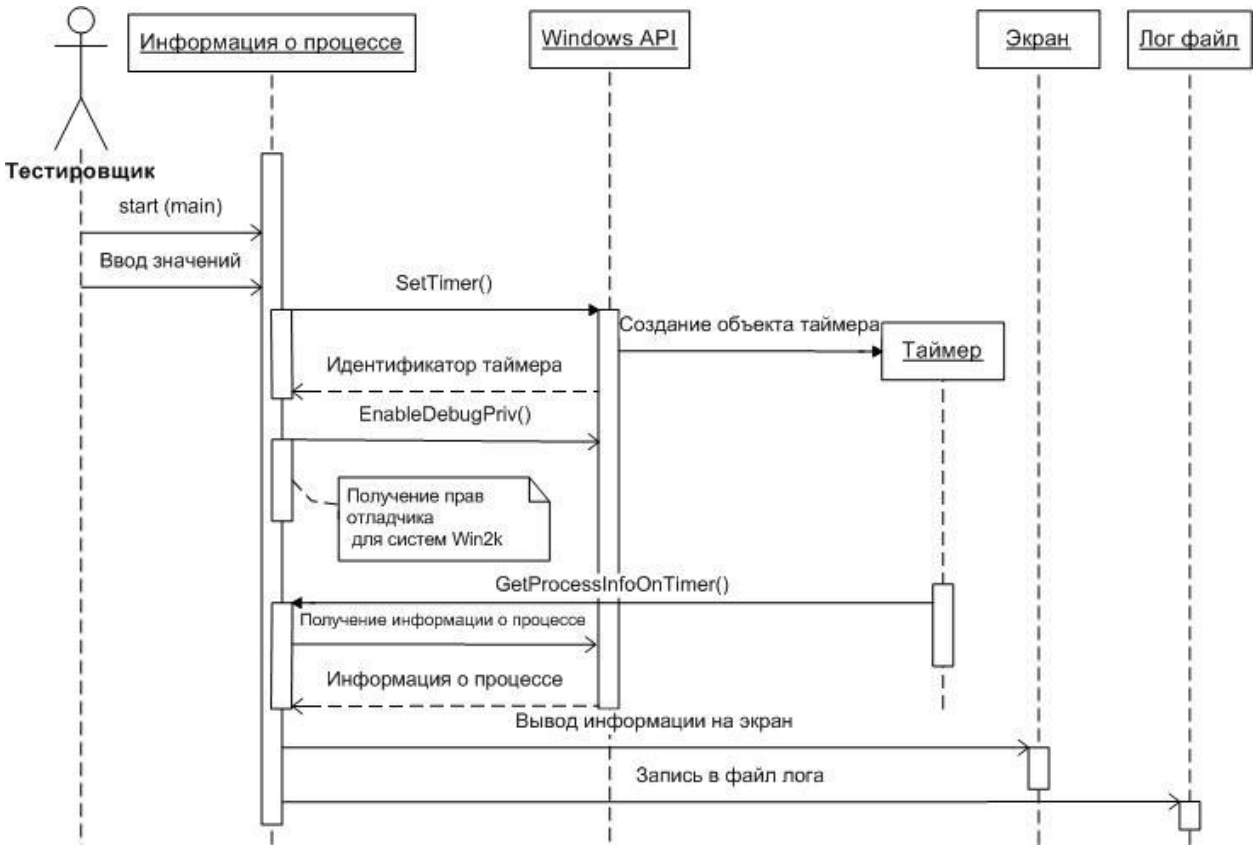

Рисунок 4. Последовательность работы приложения

На рисунке 4 изображена диаграмма последовательностей. Данная диаграмма показывает нам работу программы. В ней отражена последовательность вызовов функций в хронологическом порядке.

На рисунке 5 изображен пример работы программы.

```
Идентификатор процесса: 3680
Идентификатор таймера: 21489

                        -=Информация о процессе=-
------------------------------------------------------------------------

[Ошибки страниц] 43427
[Пиковый рабочий набор] 173508
[Текущие потребления памяти] 173508
[Мин/макс размер физической памяти] 204800 / 1413120
[Выгружаемый пиковый пул] 165
[Выгружаемый пул] 157
[Невыгружаемый пиковый пул] 30
[Невыгружаемый пул] 30960
[Выделенная память] 167532
[Выделенная пиковая память] 167536
[Количество потоков] 35
[Дескрипторы] 689
[Объекты GDI] 36
[Объекты USER] 6
[Базовый приоритет] 13 - Высокий.

------------------------------------------------------------------------

Время и Дата ОС: 15:02:47 03/13/12
```

Рисунок 5. Вид работы окна программы

Разработанные функции позволяют обращаться к процессу для получения информации о процессах приложений системы ЧПУ. Предложены алгоритмы и реализации в приложении сбора данных. Данные выводятся в консоль, а также сохраняются в файл в XML формате для последующего использования в графическом приложении.

Список литературы

1. Сосонкин В.Л, Мартинов Г.М. Системы числового программного управления: Учеб. Пособие. – М.: Логос, 2005.-296с.
2. Денисенко В.В. Компьютерное управление технологическим процессом, экспериментом, оборудованием.– М: Горячая линия – Телеком, 2009. – 608 с., ил.
3. Григорьев С.Н., Андреев А.Г., Мартинов Г.М. Перспективы развития кроссплатворменных компьютерных систем числового программного управления высокотехнологичного оборудования // Автоматизация в промышленности. 2011 №5 С.3-8.