УДК 004.416:004.896

РАЗРАБОТКА КОМПЛЕКСА АНАЛИЗА ОШИБОК В КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

© 2013 А.Ю. Крайнов, А. А. Смагин

Ульяновский государственный университет

Поступила в редакцию 21.06.2013

В данной статье описана концепция системы анализа ошибок, возникающих в процессе функционирования корпоративных программных систем. Приведён анализ современных средств мониторинга работы информационных систем. Разработанная структура системы обеспечивает эффективное выполнение всех основных процессов, связанных с идентификацией и анализом ошибок ПО.
Ключевые слова: предприятие, корпоративная информационная система, сопровождение программного обеспечения, анализ ошибок.

ВВЕДЕНИЕ

Сопровождение комплекса программных продуктов, функционирующих на отдельно взятом предприятии и составляющих его *корпоративную информационную систему* (КИС), — один из важнейших процессов управления его инфраструктурой. Неотъемлемой частью процесса сопровождения программного обеспечения (ПО) является сбор и консолидация данных об ошибках, возникающих в процессе его работы.

Источниками данных об ошибках могут быть как пользователи ПО, так и средства мониторинга вычислительных систем. В работе [1] для сопровождения ПО (в том числе для проведения мониторинга и организации обратной связи) был предложен подход на основе программных агентов. В работе [2] указанный подход был применён авторами при создании системы сбора и анализа протоколов, генерируемых ПО.

Целью настоящей работы является разработка концепции и структуры централизованной системы сбора и консолидации данных об ошибках в работе КИС.

1. СИСТЕМА АНАЛИЗА ПРОТОКОЛОВ

Разработанный макет системы анализа протоколов [2] включает программный агент, систему управления бизнес-правилами и базу прецедентов ошибок (рис. 1). В его функции входит:

- сбор программных протоколов;
- идентификация ошибок на основе последовательностей записей протокола;
- поиск аналогов ошибок в базе прецедентов;
- управление обратной связью с пользователем.

Вместе с тем для создания полноценной системы сбора и анализа ошибок как части комплекса сопровождения программного обеспечения необходимо расширить функционал системы за счёт:

- расширения средств сбора сведений об ошибках;
- расширения адаптационных возможностей системы;
- включения функций обеспечения безопасности передаваемых данных;
- интеграции с корпоративной базой знаний.

2. РАСШИРЕНИЕ СРЕДСТВ СБОРА ДАННЫХ ОБ ОШИБКАХ

Сбор протоколов (англ. *logcollection*) представляет собой процесс исследования накопления и записей, сформированных программным продуктом, с цельюих последующего анализа и может быть двух видов:

- активный: средства сбора интегрированы с подсистемой протоколирования и, как следствие, могут работать в режиме реального времени;
- пассивный: средства сбора производят сканирование уже записанных протоколов через определённые интервалы времени.

Несмотря на эффективность активных средств анализа, их применение на практике затруднено из-за различий в языках программирования, использованных для разработки ПО. Для решения этой проблемы можно использовать специализированные средства сбора протоколов, такие как Apache Flume, Lilith, log.io или Log Expert.

Зачастую анализ протоколов не может выявить всех проблем в работе ПО. В этом случае можно воспользоваться средствами мониторинга процесса выполнения приложений.

Предметная область, занимающаяся исследованием возможностей и средств наблюдения за

Крайнов Александр Юрьевич, аспирант кафедры телекоммуникационных технологий и сетей.
E-mail: kralyu@mail.ru
Смагин Алексей Аркадьевич, доктор технических наук, профессор, заведующий кафедрой телекоммуникационных технологий и сетей. E-mail: alsmagin@ulsu.ru

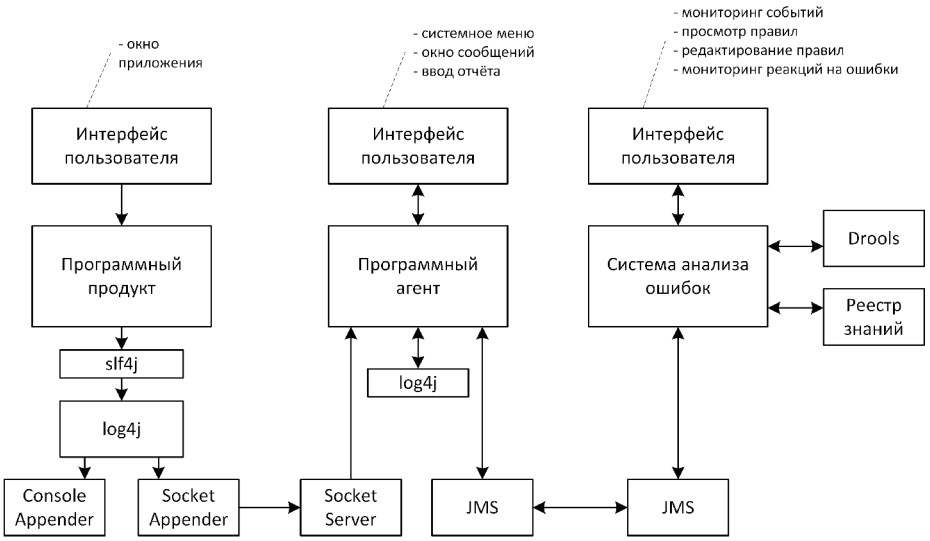

Рис. 1. Структура системы анализа протоколов

выполнением приложений, называется *управлением производительностью приложений* (англ. *Application performance management*) [3]. Для сбора дополнительных данных о производительности программ в программной инженерии разработаны следующие методы:

1. Профилирование (англ. *profiling*). Это процесс сбора низкоуровневых характеристик выполняемой программы. Обычно применяется на стадии разработки ПО для выявления проблем в производительности. Существует два основных вида профилирования: (а) событийное — основанное на механизмах перехвата событий, встроенных в язык программирования; (б) статистическое — основанное на сканировании контекста выполнения программы через определённые интервалы времени.

2. Инструментирование (англ. *instrumentation*). Это процесс анализа производительности программы, происходящий благодаря внедрению в программный код специальных блоков, собирающих данные о процессе её выполнения.

3. Анализ процесса выполнения (англ. *runtime intelligence*). Это комплексный процесс исследования статистики использования пользователями одного или нескольких программных продуктов. Связан с областью бизнес-аналитики и, как правило, применяется в корпоративных системах.

Не все перечисленные методы могут быть применены при разработке агента системы сопровождения. При выборе подходящих средств должны быть выполнены следующие условия:

1. оригинальный программный продукт не должен быть изменён указанными средствами ни на этапе разработки, ни на этапе выполнения;

2. применение указанных средств не должно приводить к нестабильной работе продукта.

Применение указанных средств не должно приводить к существенному падению производительности рабочей станции.

Полезную информацию о причинах возникшей ошибки может дать также наблюдение за окружением нестабильно работающей программы, например, аппаратными ресурсами, сервисами операционной системы, загрузкой сети. Подобная информация может быть получена при использовании средств *мониторинга сетевых ресурсов* (англ. *network monitorin gsystems*).

Данный класс систем предназначен, в первую очередь, для отслеживания статуса серверов и сетевого оборудования. Большинство систем данного класса поддерживают также мониторинг рабочих станций, включая наблюдение за использованием жёсткого диска, памяти, процессорного времени и т. д. Примерами подобным систем являются Cacti, Nagios, Pandora FMS, Zabbix, Zenoss.

Традиционными средствами сбора описаний ошибок и заявок на модификацию ПО от пользователей являются *системы отслеживания ошибок* (англ. *Bug tracking systems*) [4]. Большинство подобных системы не выполняют каких-либо сложных бизнес-процессов, требующих высокой производительности и надёжности, а являются, фактически расширенными веб-интерфейсами к базам данных, хранящим информацию о запросах.

Как следствие, их функциональность практически одинакова; различия могут наблюдаться в следующем:

1. удобство пользовательского интерфейса;

2. перечень состояний (этапов обработки) запросов на модификацию;

3. перечень полей, описывающих запрос на модификацию;

4. перечень способов добавления запросов на модификацию.

Примерами подобных систем являются Bugzilla, Fossil, Mantis, Redmine, Trac.

3. ПОВЫШЕНИЕ АДАПТАЦИОННОЙ СПОСОБНОСТИ СИСТЕМЫ

На эффективность системы анализа ошибок, выражающуюся в сокращении трудозатрат её пользователей и снижении потребляемых аппаратных ресурсов (в первую очередь — пропускной способности сети передачи данных), влияет также возможность системы к адаптации под изменяющиеся условия рабочей среды.

Последовательности событий, приводящие к ошибкам, могут автоматически выявляться на основе методов ассоциации, широко применяющихся в настоящее время для анализа рыночной корзины (англ. *market basket analysis*) [5, с. 281]. Автоматически созданные таким образом правила далее могут предоставляться пользователю для утверждения.

Полнота и детализированность данных, принимаемых системой анализа ошибок, сильно влияет на качество принимаемых ей решений. Очевидно, отслеживание записей только наивысшего уровня критичности (“ERROR”) не позволит в полной мере применить методы анализа цепочек событий. Включение наблюдения за наименее критичными записями (“TRACE”) сильно увеличит нагрузку на сеть. Решение этой проблемы заключается в создании “толстого клиента”, когда программным агентом будет осуществляться предварительная фильтрация явно бесполезных записей протоколов.

Очевидно также, что это потребует внедрения средств сетевой синхронизации для поддержании в актуальном состоянии локальных баз знаний агентов. В качестве таких средств могут выступить *системы управления конфигурацией* (англ. *configuration management software*). Данный класс систем предназначен для автоматизированного управления настройками рабочих станций, серверов и прочих узлов локальной (как правило, корпоративной) сети. Подобные системы могут быть выполнены в 2-х вариантах:

1. клиент-серверном, где клиентская часть представляет собой агента, периодически обращающегося к серверу за информацией об обновлениях конфигурации;

2. децентрализованном, где каждый узел может выступать в виде источника конфигурации.

Большинство популярных систем управления конфигурацией являются кроссплатформенными. Примерами систем управления конфигурацией являются Chef, Opsi, Puppet, Smart Frog, Spacewalk.

Текстовые сообщения, поступившие через систему обратной связи программного агента или систему отслеживания ошибок, часто содержат важную информацию для идентификации и анализа ошибки. Вместе с тем автоматический анализ сообщений от пользователей, как правило, затруднён по следующим причинам:

1. пользователь может дать неправильное или неполное описание ошибки;

2. отчёты об ошибках, выраженные в текстовом виде, сложно классифицировать.

Первая проблема может частично решаться путём фиксации набора полей, которые должен заполнить пользователь, после чего проверять данные на достаточность и корректность.

Для решения второй проблемы могут применяться различные методы интеллектуального анализа текстов (англ. *Text mining*)[6], в частности, алгоритмы *извлечения информации* (англ. *Information extraction*). Применение подобных технологий, однако, требует проведения специальных исследований по предметной области. В то же время для анализа текстовых пояснений к событиям, автоматически сгенерированных программой-источником ошибки, можно использовать более простые подходы, например, извлечение фрагментов с помощью регулярных выражений.

4. ПОВЫШЕНИЕ БЕЗОПАСНОСТИ

Распределённый характер информационной системы предъявляет усиленные требования к обеспечению безопасности как системы в целом, так и отдельных её компонентов.

Данные, пересылаемые как внутри рабочей станции (при взаимодействии сокетов), так и по сетевым каналам (например, при отправке сообщений JMS), могут содержать конфиденциальные данные. Необходимо предусмотреть средства криптографической защиты этих данных, а также защиты компонентов системы (в первую очередь, программного агента) от перехвата управления.

Не менее важной задачей следует считать повышение стабильности работы программного агента и его взаимодействия с пользователем. Испытания макета показали, что неполнота описания предметной области приводит повышенной “чувствительностью” к новым ошибкам. Неосторожное создание пользовательских правил может существенно повредить эргономичности системы. Поэтому при создании действующего комплекса необходимо обеспечить предотвращение неадекватной реакции агента из-за неправильной или неполной конфигурации базы знаний путём разработки набора ограничений для создания пользовательских правил, а также предусмотреть возможности работы агента в автономном режиме, не требующим ответа пользователя.

5. РАЗРАБОТКА КОРПОРАТИВНОЙ БАЗЫ ЗНАНИЙ

Часто ошибка в программном обеспечении связана с некоторыми особенностями предметной области или специфичностью взаимодействия участников процесса сопровождения. Разработка онтологии предметной области [7], создание *централизованной корпоративной базы знаний* и интеграция её, в частности, с системой анализа ошибок позволит повысить качество сопровождения ПО [8]. Общий интерфейс для базы знаний может быть реализован на базе Drools Guvnor или аналогичной системы.

Создание комплексной системы сопровождения (особенно основанной на знаниях) потребует проектирования эффективной системы пользовательского интерфейса и *визуализации данных*, что является одним из критериев качества современных систем бизнес-аналитики [5, с. 173].

Пользовательский интерфейс может также предоставлять инструменты для поддержки процесса *поиска первичных ошибок* (анализа первопричин — англ. *root cause analysis*), в том отображать результаты в виде специализированных диаграмм, таких как деревья ошибок (англ. *faulttrees*) и др.[9].

6. ОБЩАЯ СХЕМА КОМПЛЕКСА

Перечисленные выше средства можно представить в виде комплекса (рис. 2).

Ключевым компонентом системы является программный агент, устанавливаемый на каждой рабочей станции и взаимодействующий с пользователем. В его функции входят:

- обеспечение обратной связи пользователя с администратором (получение отчётов);
- выдача уведомлений об возникших ошибках и методах их устранения;
- мониторинг ПО (сбора протоколов, отслеживание процесса выполнения);
- мониторинг операционной системы и ресурсов рабочей станции;
- первичный (локальный) анализ ошибок.

Консолидация поступающих от программных агентов данных выполняется централизованной системой анализа ошибок, управляемой администратором. Помимо сбора и анализа данных о выполнении ПО и текстовых отчётов пользователей, в её функции входит удалённое управление программными агентами.

Важным компонентом комплекса является подсистема интеграции с корпоративной базой знаний, которая позволяет накапливать информацию об ошибках и производить более тщательный их анализ и поиск первопричин. Накопленная информация может также передаваться разработчику ПО для устранения ошибок.

Обобщённый алгоритм анализа ошибки с помощью данного комплекса выглядит следующим образом:

1. Программный агент собирает данные о работе ПО, состоянии операционной системы и потреблении аппаратных ресурсов.

2. Через определённые промежутки времени собранные данные передаются подсистеме локального анализа ошибок. Результатом анализа является заключение о том, соответствуют ли данные потенциальной ошибке.

3. При положительном решении программный агент формирует заявку на обслуживание, в которую включает информацию о рабочей станции, вызвавшем ошибку ПО и перечень программных протоколов, соответствующих ошибке.

4. Центральная подсистема анализа ошибок производит поиск аналогов ошибки в базе прецедентов. При обнаружении прецедента агенту передаётся указание на выполнение действия, ассоциированного с прецедентом. Если прецедент не

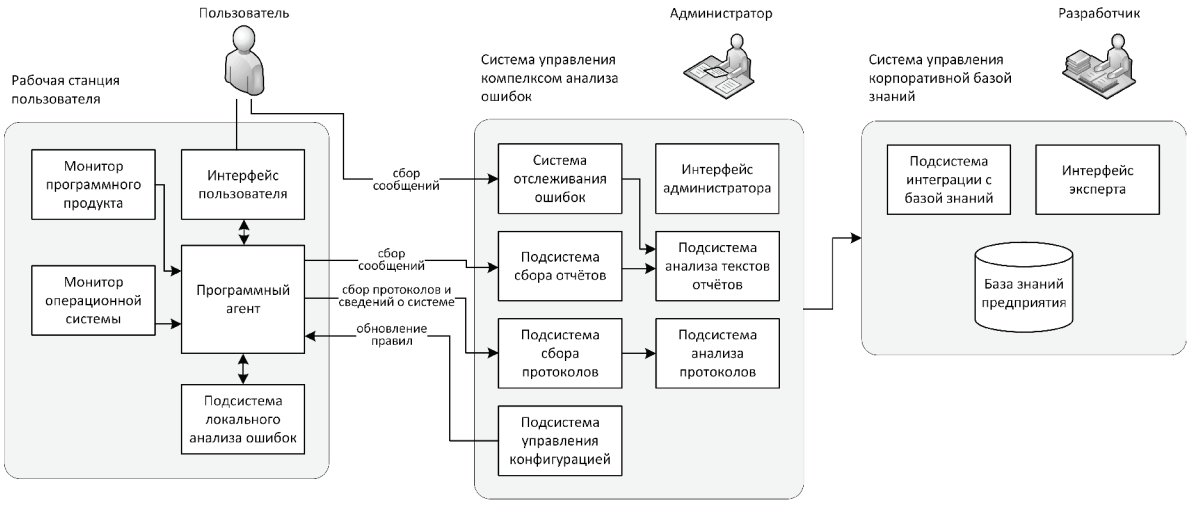

Рис. 2. Структурная схема комплекса анализа ошибок на предприятии

был найден, агенту передаётся указание на запрос от пользователя дополнительной информации.

5. Дополнительная информация от пользователя собирается программным агентом и передаётся подсистеме анализа текстов отчётов.

6. Консолидированная информация, полученная в результате анализа данных о рабочей станции, ПО, ошибке и отчёте пользователя, передаётся на рассмотрение администратора.

7. Администратор либо утверждает, либо отклоняет полученную информацию. В первом случае администратор также принимает решение по возникшей ошибке.На основе полученной информации и принятого решения формируется прецедент, который сохраняется в базе знаний предприятия.

8. При необходимости копия прецедента передаётся разработчику для возможности исправления возникшей ошибки.

ЗАКЛЮЧЕНИЕ

Разработанная концепция системы анализа ошибок позволяет создать программный комплекс, обеспечивающий эффективное выполнение всех основных процессов, связанных с идентификацией и анализом ошибок, возникающих в процессе функционирования корпоративных программных систем.

Приоритетными направлениями при разработке комплекса анализа ошибок должны стать: использование специализированных средств сбора протоколов, повышение стабильности работы и повышение безопасности. Перспективными направлениями, требующими проведения дополнительных научных исследований, являются: автоматическая генерация правил, анализ текстов пользовательских протоколов и разработка корпоративной базы знаний.

Работа выполнена в рамках государственного задания Министерства образования и науки Российской Федерации.

СПИСОК ЛИТЕРАТУРЫ

1. *Захаров В. Г., Крайнов А. Ю., Липатова С. В., Смагин А. А.* Построение системы доставки обновлений программных продуктов // Учёные записки Ульяновского государственного университета. 2012. № 1 (4). С. 161–174.
2. *Крайнов А. Ю., Смагин А. А.* Автоматизация сбора протоколов в системе сопровождения программного обеспечения на основе обработки сложных событий // Автоматизация процессов управления. 2013. № 2 (32). [В печати].
3. *Уайтхед Н.* Мониторинг работы Java-приложений [Электронный ресурс] // IBM DeveloperWorks. 2009. URL: http://www.ibm.com/developerworks/ru/library/j-rtm1/index.html (дата обращения 12.05.2013).
4. *КолинА.* Обзор систем отслеживания ошибок [Электронный ресурс] // Центр Компетенций Atlassian. 2010. URL: http://www.teamlead.ru/x/ZwDx (дата обращения 12.05.2013).
5. *Паклин Н., Орешков В.* Бизнес-аналитика. От данных к знаниям. 2-е изд. Питер, 2013. 704 с.
6. *Пескова О.В.* Алгоритмы классификации полнотекстовых документов // Автоматическая обработка текстов на естественном языке и компьютерная Лингвистика. М.: МИЭМ, 2011. С. 170–212.
7. *Куприянов А.А., Мельниченко А.С.* Модели структуризации и формализации онтологии предметной области на стадиях проектирования автоматизированных систем // Автоматизация процессов управления. 2010. № 2 (20). С. 70–75.
8. *Rodríguez O.M. et al.* Using a Multi-agent Architecture to Manage Knowledge in the Software Maintenance Process // Knowledge-Based Intelligent Information Engineering Systems / Ed. by Negoita M.G., Howlett R.J., Jain L.C. Springer Berlin Heidelberg, 2004. P. 1181–1188.
9. *Hari A.Yu.* The Shortcomings of Existing Root Cause Analysis Tools // Proc. World Congr. Eng. 2012 / Ed. by S. I. Ao et al. UK: London: Newswood Limited, 2012. Vol. 3.

DEVELOPMENT OF THE BUG PROCESSING SYSTEM FOR ENTERPRISE INFORMATION SYSTEMS

© 2013 A.Yu. Krainov, A.A. Smagin

Ulyanovsk State University

The article presents the conception of the bug processing system for enterprise software. We have analyzed a range of modern tools for information systems operation monitoring. Developed system architecture provides an effective execution of all main processes of software bug detection and processing.
Keywords: enterprise information system, software maintenance, bug processing.

Alexander Krainov, Graduate Student at the Telecommunication Systems and Networks Department. E-mail: kralyu@mail.ru
Alexey Smagin, Doctor of Technics, Professor, Head at the Telecommunication Systems and Networks Department. E-mail: alsmagin@ulsu.ru