1) Обеспечить возможность приёма платежей пластиковыми картами в модуле розничной торговли ERP-системы.

2) Предоставить возможность сохранять информацию о совершённых платежах с помощью пластиковой карты в ERP-системе.

3) Обеспечить взаимодействие сервиса платёжной системы с устройством считывания информации с пластиковой карты.

4) Предоставить администраторам ERP-системы возможность настройки взаимодействия с платёжной системой.

Программное средство интеграции ERP-системы с системой проведения платежей предусматривает выполнение следующих основных функций:

1) Формирование запроса на проведение платёжной операции к системе проведения платежей PXP.

2) Обработка ответа от системы проведения платежей PXP о статусе проведённого платежа.

3) Формирование запроса на проверку актуальности программного обеспечения устройства считывания информации с пластиковой карты.

4) Обработка выполненного платежа в формате, понятном для ERP-системы Microsoft Dynamics AX 2012 R3.

Предлагаемое программное средство интеграции состоит из следующих компонентов:

1) Модуль обработки платёжных операций для Microsoft Dynamics AX 2012 R3 for Retail POS.

2) Модуль настроек интеграции с платёжной системой для Microsoft Dynamics AX 2012 R3.

3) Модуль взаимодействия с устройством считывания информации с пластиковой карты для Microsoft Dynamics AX 2012 R3 for Retail POS.

Таким образом, предлагаемое программное средство интеграции ERP-системы Microsoft Dynamics AX 2012 R3 и системы проведения платежей PXP позволяет осуществлять хранение и учёт проведенных платежей, что упрощает управление предприятием и контроль его финансов.

Список использованных источников:

1. ERP-системы: выбор, внедрение, эксплуатация. Современное планирование и управление ресурсами предприятия / Дэниел О'Лири — М.: Вершина, 2004

2. Платонов, Ю.Г. Методы обеспечения интеграции информационных систем / Ю.Г. Платонов. — Институт систем информатики им. А.П. Ершова, 2011.

СТРУКТУРИРОВАННОЕ ФОРМАТИРОВАНИЕ КОДА НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ В СИСТЕМАХ РАСПРЕДЕЛЕННОГО КОНТРОЛЯ ВЕРСИЙ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Подлозный В. А.

Куликов С.С. – к.т.н., доцент

Подход к форматированию исходного кода во многих языках высокого уровня является вопросом личных предпочтений разработчика. В коллективной разработке ПО нередки ситуации, когда правки в файлы с исходным кодом вносятся несколькими разработчиками, придерживающимися, зачастую, различных стилей форматирования. В работе представлена высокоуровневая архитектура системы, позволяющей использовать индивидуальный стиль форматирования исходного кода каждым из разработчиков, при поддержании единых стандартов форматирования в удалённых репозиториях систем распределённого контроля версий.

Форматирование кода для компиляторов многих высокоуровневых языков программирования является семантически несущественным. Тем не менее, исследования показывают, что систематическое следование определённым правилам форматирования улучшает читаемость кода и может способствовать повышению качества конечного программного продукта [1], [2].

Современные интегрированные среды разработки обладают необходимым функционалом для поддержания заданного стиля форматирования кода. Выбор конкретных стилевых опций, однако, определяется личными предпочтениями разработчика, что создаёт проблему при использовании систем контроля версий (в частности, распределённых, где на рабочей машине каждого из разработчиков размещается полнофункциональная локальная копия удалённого репозитория) в коллективной разработке ПО.

Несмотря на предпринятые попытки построения утилит семантического сопоставления ревизий файлов с исходным кодом [3], распространённым инструментом просмотра истории изменений всё ещё остаются diff-подобные утилиты посимвольного сравнения. Существенным ограничением данного класса утилит является неспособность разделить правки, вносящие изменения в логику работы исходного кода, от правок, вносящих изменения форматирования, для логики работы несущественных.

Распространённым подходом к решению данной проблемы при коллективной разработке ПО является

выработка единых для всех разработчиков стандартов форматирования. Несмотря на это, с высокой вероятностью различные проекты будут иметь отличные стандарты форматирования, что вынудит разработчика всякий раз перестраиваться при работе над очередным проектом.

Альтернативой вышеназванному подходу может служить подход, основанный на использовании ассоциированных с каждым репозиторием распределённой системы контроля версий (будь то локальным для каждого из разработчиков или удалённым) специфичных опций форматирования. В процессе копирования репозитория разработчик может задать предпочитаемый стиль форматирования, отличный от такового для исходного репозитория. Каждая ревизия полученных из исходного репозитория файлов форматируются в соответствии с заданными предпочтениями.

В процессе внесения правок в исходный код, в локальном репозитории разработчик руководствуется индивидуальным стилем форматирования. При публикации правок в удалённый репозиторий, выполняется форматирование файлов в соответствии с настройками удалённого репозитория, и сохранение в таком виде под версионным контролем. Таким образом, семантически эквивалентные файлы с исходным кодом имеют различное форматирование в локальном для каждого из разработчиков репозитории, но единое в централизованном, что позволяет эффективно применять diff-подобные утилиты для выявления изменений в логике работы исходного кода.

Ограничением предложенного подхода можно считать его несовместимость с системами контроля версий, основанными на хранении записей об изменении содержимого файлов (Mercurial), а не самих изменённых файлов (Git).

Список использованных источников:

1. Buse R. P. L., Weimer W. R. A metric for software readability // Proceedings of the 2008 international symposium on Software testing and analysis. – Seattle, WA, USA: ACM, 2008. – С. 121-130.

2. Miara R. J., Musselman J. A., Navarro J. A., Shneiderman B. Program indentation and comprehensibility // Commun. ACM. – 1983. – Т. 26, № 11. – С. 861-867.

3. Jackson D., Ladd D. A. Semantic Diff: a tool for summarizing the effects of modifications // Software Maintenance, 1994. Proceedings., International Conference on – 1994. – С. 243-252.

АВТОМАТИЧЕСКОЕ ПОСТРОЕНИЕ АРХИТЕКТУРЫ НЕЙРОННОЙ СЕТИ ПРИ ПОМОЩИ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Попитич А. Я.

Серебряная Л. В. – к-т. техн. наук, доцент

В ходе решения задач, где необходимо использование нейронной сети, всегда присутствует проблема выбора конкретной архитектуры сети. Эта проблема в большинстве случаев решается простым подбором и подгоном параметров по причине разнообразия анализируемых данных и уникальности решаемых задач. В данном докладе будет предложено применение генетического алгоритма, который позволяет автоматизировать процесс и сгенерировать сеть, которая будет соответствовать анализируемым данным и решаемой задачи.

Основная задача была поставлена следующим образом – обеспечить построение топологии нейронной сети автоматическим образом. В качестве исходных данных была взята обучающая выборка для нейронной сети, которая в дальнейшем будет применяться для обучения каждого, генерируемого генетическим алгоритмом, экземпляра нейронной сети. Все эксперименты для простоты были организованы для решения небольшой задачи XOR, результаты которой могут быть впоследствии применены и для решения более масштабных задач.

Для применения генетического алгоритма в ходе решения поставленной задачи было использовано прямое кодирование связей нейронной сети. На рисунке 1 представлен пример небольшой нейронной сети и кодирования межнейронных связей.

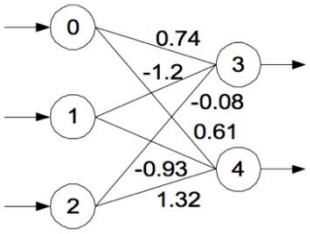

Рис. 1 – Пример топологии нейронной сети