ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

DOI: 10.25586/RNU.V9187.21.01.P.136 УДК 004.75

А.А. Башлыкова, Д.В. Растягаев

ИНТЕРОПЕРАБЕЛЬНОСТЬ РЕПОЗИТОРИЕВ ВЕРСИЙ ИНСТРУМЕНТАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСОВ

Анализируется возможность достижения интероперабельности репозиториев, что важно для работы программистов над конечным программным продуктом (ПП). Рассмотрены различные программные средства, предназначенные для использования в ходе проектирования, разработки и сопровождения программ. Описана специфика работы с репозиториями (программными пакетами) в контексте управления версиями инструментального программного обеспечения. Представлены инструменты контейнеризации программного обеспечения. Разработаны модели достижения интероперабельности репозиториев версий инструментального программного обеспечения вычислительных комплексов.
Ключевые слова: вычислительный комплекс, интероперабельность, инструментальное программное обеспечение, репозиторий.

A.A. Bashlykova, D.V. Rastyagaev

INTEROPERABILITY OF REPOSITORIES OF VERSIONS INSTRUMENTAL SOFTWARE COMPUTING COMPLEXES

The article analyzes the possibility of achieving interoperability of repositories, which is important for programmers to work on the final software product (SP). Various software tools intended for use in the design, development and maintenance of programs are considered. The specifics of working with repositories (software packages) in the context of version control of instrumental software are described. The software containerization tools are presented. Models of achieving interoperability of repositories of versions of instrumental software of computing systems have been developed.
Keywords: computing complex, interoperability, instrumental software, repository.

Вводные замечания

Специалисты в области информационных технологий постоянно используют инструментальное программное обеспечение (ИПО) в своем рабочем окружении и вычислительных комплексах. На этапах проектирования, разработки и сопровождения незаменимо применение уже созданного программного обеспечения (ПО). Доступ к ИПО должен быть непрерывным, особенно если сотрудники работают на удаленной основе; важно обеспечение единого и быстрого доступа к рабочему окружению, вычислительному комплексу, что возможно только при условии хранения ИПО в сети репозиториев.

Наряду с интероперабельностью и доступностью рабочего окружения необходимо предусмотреть возможность настройки ИПО, а процесс тестирования внесенных изменений и размещения в необходимом репозитории должен быть автоматизированным, с встроенным механизмом уведомления об успешном или неуспешном прохождении заранее написанных тестов.

Актуальность рассмотрения интероперабельности репозиториев ИПО обусловлена:

- развитием облачных технологий в российском IT-сегменте рынка;
- широким кругом областей, в которых необходимо применение инструментального программного обеспечения;
- возможностью непрерывного доступа к инструментальному программному обеспечению удаленно для его использования в рабочем окружении.

Инструментальное программное обеспечение

Инструментальное программное обеспечение (ИПО) предназначено для использования в ходе проектирования, разработки и сопровождения программ [5]. К данной категории относятся различные программные средства (ПС):

1) *компиляторы* – специальные трансляторы, выполняющие перевод написанных на каком-либо языке программирования команд программы в набор машинных кодов, понятных компьютеру, например, Min GW, Clang, Build Tools for VS;

2) *интерпретаторы* – программы – анализаторы команд, которые сразу же их и выполняют, например, Repl.It, Sym Py, I Python;

3) *текстовые редакторы* – программы для редактирования текстовых данных, такие как Emacs, Vim, Neovim, Sublime Text, Notepad++;

4) *интегрированные средства разработки* – комплексы программных средств, которые используются разработчиками для разработки ПО: Jet Brains IntelliJ, Visual Studio, Net Beans, Code::Blocks, Anaconda;

5) *средства непрерывной интеграции* – программные решения, используемые для менеджмента проектов, тестирования кода в реальном времени, развертывания приложения в различных окружениях с автоматическим подключением всех необходимых зависимостей, например, Jenkins, Team City, Travis CI, Git Lab CI;

6) *системы управления версиями* – программы, облегчающие работу с часто изменяющейся информацией за счет мониторинга ее версий: Git, Mercurial;

7) *средства автоматизированного тестирования* – программы, позволяющие выполнять тесты проверки работоспособности и функционала программного обеспечения: Git Lab Auto DevOps, Test Complete;

8) *проблемно ориентированные инструментальные системы* – системы, решающие определенную специфическую проблему: САПР, АСУ, АРМ, 1С-Битрикс.

ПО, используемое в реализации функционала подсистем вычислительных комплексов, должно быть с исходным кодом в соответствии с ГОСТ Р 54593–2011 «Свободное программное обеспечение. Общие положения».

Репозитории и система управления версиями инструментального программного обеспечения

На современном этапе развития информационных технологий работа с репозиториями становится все более распространенным явлением. Удобное хранение и удаленный доступ к данным стали доступны широкому кругу пользователей еще в 2002 г. благодаря компании Amazon, запустившей серию сервисов, основанных на применении облачных технологий. Спустя некоторое время заинтересованность в облачных решениях проявили такие крупные компании, как IBM, Microsoft, Oracle, Google. Россия также принимает

участие в развитии ПО, предоставляемого в виде услуг (SaaS), составляя конкуренцию западным решениям в российском сегменте IT-рынка.

Под репозиторием (программных пакетов) понимается замкнутая совокупность программных пакетов и метаинформации о них. Репозиторий называется замкнутым, если для каждого бинарного пакета возможно вычислить его замыкание, то есть можно установить пакет в систему с соблюдением всех его зависимостей. С ростом доступности облачных технологий удаленные репозитории становятся все более распространенными для хранения ИПО вычислительных комплексов. Вопрос автоматизации технических процессов с применением практик CI/CD рассмотрен в многочисленных иностранных публикациях [8–10]. Особую значимость в определении данных практик имели работы таких инженеров, как Эберхард Вольф и Мартин Вофлер.

Система управления версиями (Version Control System, VSC) облегчает работу с динамически изменяющимися данными. Помимо исходных кодов ПО системы управления версиями способны хранить и другие типы данных, в том числе мультимедийные файлы, векторную и растровую графику, что расширяет область применения подобных систем в иных сферах, например дизайне.

На современном рынке можно выделить конкурентоспособную Git-систему – Mercurial. Главным ее отличием от Git является то, что в Mercurial используется распределенная модель хранения данных вместо традиционной клиент-серверной модели. Данная модель хранения не подразумевает обязательного наличия централизованного репозитория, так как вся история изменения файлов хранится локально на рабочих станциях пользователей, и при необходимости отдельные файлы синхронизируются с файлами, которые хранятся уже в репозитории на другой рабочей станции.

Синхронизация между разными версиями ведется с помощью обмена так называемыми патчами, или наборами изменений. Также данная система обязывает пользователя иметь свой локальный репозиторий [7].

Инструменты контейнеризации программного обеспечения

Необходимо отличать контейнеризацию и виртуализацию с помощью виртуальных машин. Данные технологии схожи по своим функциям, обе предоставляют механизмы изоляции пользовательского окружения, однако отличаются по методам реализации. Принципиальное отличие заключается в том, что контейнеры запускают дискретный процесс в операционной системе рабочей станции хоста, как это работает на примере инструмента Docker; при этом количество данных процессов с контейнерами, которые могут иметь в себе отличающиеся рабочие окружения, ограничивается лишь памятью рабочей станции. Уровень изоляции ограничивается строго теми параметрами, которые задал пользователь.

Одним из ведущих решений в плане контейнеризации является Docker, который позволяет «упаковать» разработанное ПО в контейнер, в котором работает определенный образ операционный системы, что предоставляет настраиваемую среду по управлению контейнерами. В отличие от других контейнерных инструментов Docker предоставляет виртуальную сеть на основе хост-машины, которая обеспечивает безопасное и удобное межконтейнерное взаимодействие. Docker – один из самых современных механизмов контейнеризации. В Docker были решены вопросы, связанные с интероперабельностью,

кроссплатформенностью, возможностью работы сразу с несколькими копиями одновременно, удобством подключения различных зависимостей.

Достижение интероперабельности репозиториев версий инструментального программного обеспечения вычислительных комплексов

С ростом конкурентности на рынке информационных технологий организации начали уделять большое внимание ПО и выделять ресурсы на разработку ПО в ускоренном темпе. Такие практики, как непрерывная интеграция (Continuous Integration, CI), непрерывная доставка (Continuous Delivery, CDE), непрерывное развертывание (Continuous Deployment, CD), нацелены помочь в ускорении процессов разработки ПО без какого-либо ущерба для итогового продукта или вычислительного комплекса.

Для демонстрации отношений между всеми данными практиками можно использовать схему на рисунке 1. Подобное графическое изображение поможет выстроить полное понимание того, в какие моменты применяется та или иная практика. Каждая из практик играет свою роль в достижении интероперабельности.

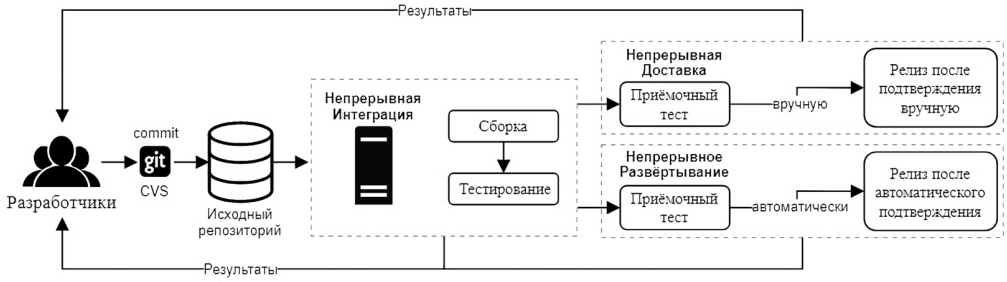

Рис. 1. Схема интероперабельности между непрерывной интеграцией, доставкой и развертыванием

Основой всей системы является автоматизированный конвейер, который на входе принимает весь проект, синхронизирует все копии изменений, которые были внесены разработчиком, автоматически собирает его и применяет написанные заранее тесты.

Помимо данного функционала конвейер выполняет роль сервиса уведомлений, предварительно настроенный администратором с помощью механизма Webhook. Благодаря такому распределению вычислительных ресурсов с применением автоматизации и практик CI/CD внедрение данной системы на производстве позволит не только сэкономить количество человеко-часов на выполнение рутинных задач, но и ускорить цикл разработки инструментального программного обеспечения.

Модель информационно-вычислительной системы является той необходимой основой, на которую стоит ориентироваться при построении прототипа, так как в модели наиболее точно описаны все связи между объектами и выбраны технические решения, поддерживающие требуемую степень интероперабельности.

Модель (рис. 2) была построена с учетом влияющих критериев, которые отразились на выборе вспомогательного программного обеспечения и сервисов:

1) *контроль версий* – информационная система вычислительных комплексов, поддерживающая управление версиями ИПО;

2) *мониторинг проблем* – сервисы, включенные в архитектуру системы и имеющие средства мониторинга каких-либо проблем с возможностью уведомления администратора и разработчиков о возможных неполадках;

3) *непрерывная доставка, интеграция и развертывание* – характеристики, влияющие на уровень автоматизации системы.

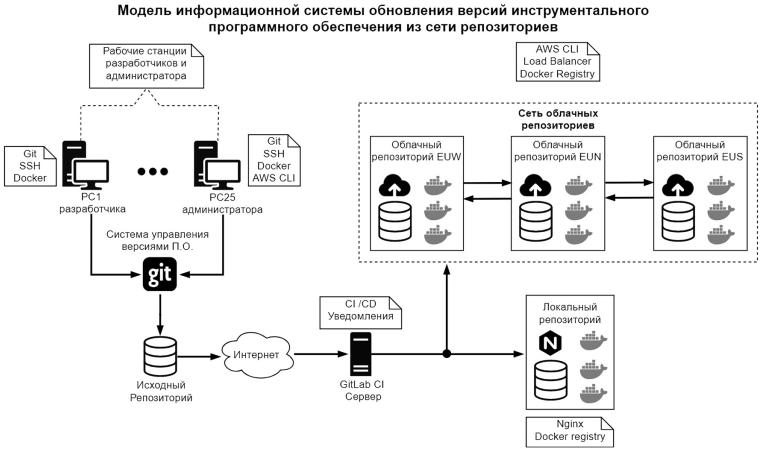

Рис. 2. Схема модели информационной системы репозиториев обновления версий инструментального программного обеспечения вычислительных комплексов

На рисунке 3 представлена эталонная модель интероперабельности.

Рис. 3. Эталонная модель интероперабельности систем вычислительных комплексов

Степень достижения интероперабельности репозиториев следует рассматривать в тесной связи со следующими характеристиками:

- функциональность;
- производительность;
- надежность;
- масштабируемость;
- безопасность (чем выше требования безопасности, тем, как правило, значительно тяжелее обеспечить интероперабельность, и наоборот);

• адаптируемость (повышение способности к адаптации, реакция на все возможные изменения у систем с высоким уровнем интероперабельности станет гораздо выше, а обновления и доработки будут требовать значительно меньшего объема стоимостных и временных ресурсов).

Заключение

Достижение интероперабельности репозиториев версий инструментального ПО вычислительных комплексов – сложная задача, решение которой должно проходить в соответствии с описанием в ГОСТ Р 55062–2012 «Системы промышленной автоматизации и их интеграция. Интероперабельность. Основные положения».

Для достижения интероперабельности репозиториев версий инструментального ПО вычислительных комплексов необходимо выполнить:

1) ранжирование набора показателей для определения таких из них, которые будут оказывать наибольшее влияние на показатель интероперабельности;

2) построение иерархии показателей интероперабельности по принятой эталонной модели OSI/RM в рамках построения профиля интероперабельности;

3) построение профиля интероперабельности репозиториев версий инструментального программного обеспечения вычислительных комплексов.

Стандарты профиля интероперабельности репозиториев версий инструментального ПО вычислительных комплексов должны обеспечивать следующие слои:

- технический (независимо от программно-аппаратной платформы);
- семантический (понимание информации из всех репозиториев как источников);
- организационный.

Анализ показал, что достижение интероперабальности репозиториев версий инструментального ПО вычислительных комплексов – достижимая задача. В статье представлена схема модели информационной системы репозиториев обновления версий ИПО механизма контейнеризации Docker.

Литература

1. *Башлыкова А.А., Олейников А.Я.* Интероперабельность и информационное противоборство в военной сфере // Журнал радиоэлектроники. 2016. № 11. С. 1–21 [Электронный ресурс]. – URL: http://jre.cplire.ru/jre/dec16/14/text.pdf (дата обращения: 20.01.2021).

2. *Башлыкова А.А., Каменщиков А.А., Олейников А.Я., Широбокова Т.Д.* Подход к обеспечению интероперабельности в высокопроизводительной среде на примере e-science // Журнал радиоэлектроники. 2019. № 11. С. 1–14 [Электронный ресурс]. – URL: http://jre.cplire.ru/jre/nov19/5/text.pdf (дата обращения: 20.01.2021).

3. ГОСТ Р ИСО/МЭК 12207–2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. М.: Стандартинформ, 2011. 188 с.

4. ГОСТ Р 55062–2012. Системы промышленной автоматизации и их интеграция. Интероперабельность. Основные положения. Введ. 2012-11-13. М.: Стандартинформ, 2012. 12 с.

5. *Журавлев Е.Е., Иванов С.В., Каменщиков А.А., Корниенко В.Н. и др.* Особенности методики обеспечения интероперабельности в грид-среде и облачных вычислениях // Компьютерные исследования и моделирование. 2015. Т. 7, № 3. С. 675–682. DOI: 10.20537/2076-7633-2015-7-3-675-682

6. *Evstigneev K.* Continuous Delivery Blueprint: Software Change Management for Enterprises in the Era of Cloud, Microservices, DevOps, and Automation // Grid Dynamics. 2018. Pp. 63–89.
7. *Kim G., Debois P., Willis J., Humble J.* The DevOps HandBook: How To Create World-Class Agility, Reliability, and Security in Technology Organizations // IT Revolution Press. 2016. Pp. 48–72.
8. *Lu J., Yang Z.* Implementation of Continuous Integration and Automated Testing in Software Development of Smart Grid Scheduling Support System // International Conference on Power System Technology (POWERCON), 2014. Pp. 2441–2446.
9. *Mahdavi-Hezaveh R., Dremann J., Williams L.* Software Development with Feature Toggles: Practices Used by Practitioners // Empirical Software Engineering. 2021. Vol. 1. Pp. 1–36. DOI: 10.1007/s10664-020-09901-z
10. *Mojtaba S., Muhammad A., Liming Z.* Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices // IEEE Access, 2017. Vol. 5. Pp. 3909–3943. DOI: 10.1109/ACCESS.2017.2685629

Literatura

1. *Bashlykova A.A., Olejnikov A.YA.* Interoperabel'nost' i informacionnoe protivoborstvo v voennoj sfere // ZHurnal radioelektroniki. 2016. № 11. S. 1–21 [Elektronnyj resurs]. – URL: http://jre.cplire.ru/jre/dec16/14/text.pdf (data obrashcheniya: 20.01.2021).
2. *Bashlykova A.A., Kamenshchikov A.A., Olejnikov A.Ya., Shirobokova T.D.* Podhod k obespecheniyu interoperabel'nosti v vysokoproizvoditel'noj srede na primere e-science // Zhurnal radioelektroniki. 2019. № 11. S. 1–14 [Elektronnyj resurs]. – URL: http://jre.cplire.ru/jre/nov19/5/text.pdf (data obrashcheniya: 20.01.2021).
3. GOST R ISO/MEK 12207–2010. Informacionnaya tekhnologiya. Sistemnaya i programmnaya inzheneriya. Processy zhiznennogo cikla programmnyh sredstv. M.: Standartinform, 2011. 188 s.
4. GOST R 55062–2012. Sistemy promyshlennoj avtomatizacii i ih integraciya. Interoperabel'nost'. Osnovnye polozheniya. Vved. 2012-11-13. M.: Standartinform, 2012. 12 s.
5. *Zhuravlev E.E., Ivanov S.V., Kamenshchikov A.A., Kornienko V.N. i dr.* Osobennosti metodiki obespecheniya interoperabel'nosti v grid-srede i oblachnyh vychisleniyah // Komp'yuternye issledovaniya i modelirovanie. 2015. T. 7, № 3. C. 675–682. DOI: 10.20537/2076-7633-2015-7-3-675-682.
6. *Evstigneev K.* Continuous Delivery Blueprint: Software Change Management for Enterprises in the Era of Cloud, Microservices, DevOps, and Automation // Grid Dynamics. 2018. Pp. 63–89.
7. *Kim G., Debois P., Willis J., Humble J.* The DevOps HandBook: How To Create World-Class Agility, Reliability, and Security in Technology Organizations // IT Revolution Press. 2016. Pp. 48–72.
8. *Lu J., Yang Z.* Implementation of Continuous Integration and Automated Testing in Software Development of Smart Grid Scheduling Support System // International Conference on Power System Technology (POWERCON), 2014. Pp. 2441–2446.
9. *Mahdavi-Hezaveh R., Dremann J., Williams L.* Software Development with Feature Toggles: Practices Used by Practitioners // Empirical Software Engineering. 2021. Vol. 1. Pp. 1–36. DOI: 10.1007/s10664-020-09901-z
10. *Mojtaba S., Muhammad A., Liming Z.* Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices // IEEE Access, 2017. Vol. 5. Pp. 3909–3943. DOI: 10.1109/ACCESS.2017.2685629