УДК 004.4’232
DOI 10.17513/snt.39950

# АРХИТЕКТУРА РАСПРЕДЕЛЕННОЙ ИНТЕГРИРОВАННОЙ СРЕДЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**[1]Сазонов А.С., [2]Виденин С.А.**

*[1]ФГАОУ ВО «Сибирский федеральный университет», Красноярск, e-mail: bennettst@yandex.ru;*
*[2]ФГАОУ ВО «Национальный исследовательский университет «Высшая школа экономики», Москва, e-mail: svidenin@hse.ru*

**Аннотация.** В статье рассматривается концепция архитектуры интегрированной среды разработки, построенной на идее распределенных компонентов, и является развитием облачных технологий, что предполагает, в свою очередь, выполнение вычислительных операций над исходным кодом в облаке, а не на персональном компьютере. Модули анализа кода, отладчик, навигация и другие компоненты рассматриваемой концепции среды разработки будут работать на облачных устройствах, отправляя на персональный компьютер уже готовый результат. Таким образом, среда разработки с распределенной архитектурой позволит программистам удаленно работать над проектами, будучи нетребовательной к характеристикам персонального компьютера. Это решение позволит не только снизить себестоимость разработки программного обеспечения, но и уменьшит возможность утечки исходного кода программы, так как весь исходный код будет располагаться в облачной инфраструктуре правообладателя программного кода, а не на машине разработчика. Представленная архитектура распределенной интегрированной среды разработки программного обеспечения позволит повысить безопасность исходного кода, уменьшить требования к рабочим станциям программистов и повысить удобство разработки в распределенных командах. Также распределенная архитектура позволяет использовать компоненты среды разработки в парном программировании, предоставляя полный или частичный доступ к функциям распределенной среды разработки.

**Ключевые слова: операциональные преобразования, интегрированная среда разработки, распределенная интегрированная среда разработки, распределенные вычисления, облачные вычисления**

# DISTRIBUTED INTEGRATED DEVELOPMENT ENVIRONMENT ARCHITECTURE FOR SOFTWARE DEVELOPMENT

**[1]Sazonov A.S., [2]Videnin S.A.**

*[1]Siberian Federal University, Krasnoyarsk, e-mail: bennettst@yandex.ru;*
*[2]National Research University Higher School of Economics, Moscow, e-mail: svidenin@hse.ru*

**Annotation.** The paper considers the concept of an integrated development environment architecture based on the idea of distributed components and is a continuation of cloud technologies, which in turn involves performing computational operations on source code in the cloud rather than on a personal computer. Code analysis modules, debugger, navigation, and other components of the development environment concept will operate on cloud devices, sending a finished result to a personal computer. Thus, a distributed development environment allows programmers to work remotely on projects without being demanding on personal computer specifications. This solution will not only reduce the cost of software development but also reduce the possibility of a program’s source code leakage due to the fact that the entire source code will be located in the cloud infrastructure of the software code holder rather than on the developer’s machine. The presented distributed architecture of the integrated development environment for software development will increase the security of the source code, reduce requirements for programmers’ workstations, and increase the convenience of development in distributed teams. Also, the distributed architecture allows using development environment components in pair programming, providing full or partial access to distributed development environment functions.

**Keywords: operational transformations, integrated development environment, distributed integrated development environment, distributed computing, cloud computing**

В современном мире информационные технологии играют важную роль во всех сферах жизни. Но нельзя упускать из внимания то, что параллельно с развитием ИТ, развивается и киберпреступность, вследствие чего ИТ-компаниям приходится тратить ресурсы на обеспечение безопасности данных сотрудников и их клиентов. Учитывая то, что последние несколько лет, особенно в сфере ИТ, увеличивается доля сотрудников, работающих удаленно, можно говорить о том, что безопасности данных сотрудников и внутренних данных компании необходимо уделять больше внимания. Отметим, что все эти вызовы стоят не только перед ИТ-компаниями, которые занимаются разработкой прикладного программного обеспечения, но и перед создателями интегрированных сред разработки (Integrated Development Environment, IDE), так как именно они работают с исходным кодом разрабатываемого компаниями программного обеспечения.

В настоящее время, для написания исходного кода разрабатываемой системы, IDE требует от программиста скачивания

исходного кода разрабатываемой системы на рабочую станцию программиста. Эта необходимость повышает вероятность утечки кодовой базы разрабатываемой системы, особенно если в команде разработки есть удаленные сотрудники. Также при разработке коммерческих систем компании необходимо обеспечить команду программистов мощными рабочими станциями, чтобы IDE могла работать с большими объемами исходного кода.

Все эти проблемы присущи современным IDE, основанным на монолитной архитектуре. Монолитная архитектура [1, 2] представляет собой систему с взаимозависимыми подсистемами. Подсистема – это некоторый реализованный функционал внутри системы, решающий определенную задачу.

Примером таких подсистем в архитектуре IDE могут служить:

– редактор кода;
– проектная модель;
– отладчик;
– поддержка конкретного языка программирования.

Все эти подсистемы объединены в одну большую кодовую базу. В архитектуре монолитных IDE присутствует понятие модульности, которое предполагает динамическое подключение подсистем при их необходимости.

Несмотря на механизм модульности, в монолитных IDE остается проблема с зависимостью подсистем друг от друга [3], что приводит к необходимости скачивания исходного кода и обеспечения программистов дорогостоящими рабочими станциями. Для решения этих проблем, а также для повышения удобства разработки предлагается подход организации архитектуры IDE, в основе которого распределенность.

Целью данного исследования является описание концептуальной модели архитектуры распределенной интегрированной среды разработки. На данный момент основной архитектурой интегрированных сред разработки является монолитная архитектура, выполняющая всю вычислительную работу на компьютере программиста. В последние годы создатели интегрированных сред разработки уделяют большое внимание распределенности. Возможность выполнять сложную вычислительную работу в облаке, не нагружая компьютер программиста, открывает много новых возможностей. Представленная концепция архитектуры позволит добиться увеличения скорости выполнения сложных вычислительных операций и даст новые возможности в коллаборативной разработке.

### *Концепция распределенной интегрированной среды разработки*

Главная идея распределенных IDE заключается в декомпозиции всего исходного кода на подсистемы, слабо связанные друг с другом. Каждая такая подсистема должна решать конкретные задачи и быть самодостаточной. Подсистемы могут быть запущены на разных рабочих станциях и могут взаимодействовать друг с другом по защищенной сети. Для обеспечения удобства разработки в распределенных командах, распределенная IDE может поддерживать совместную разработку.

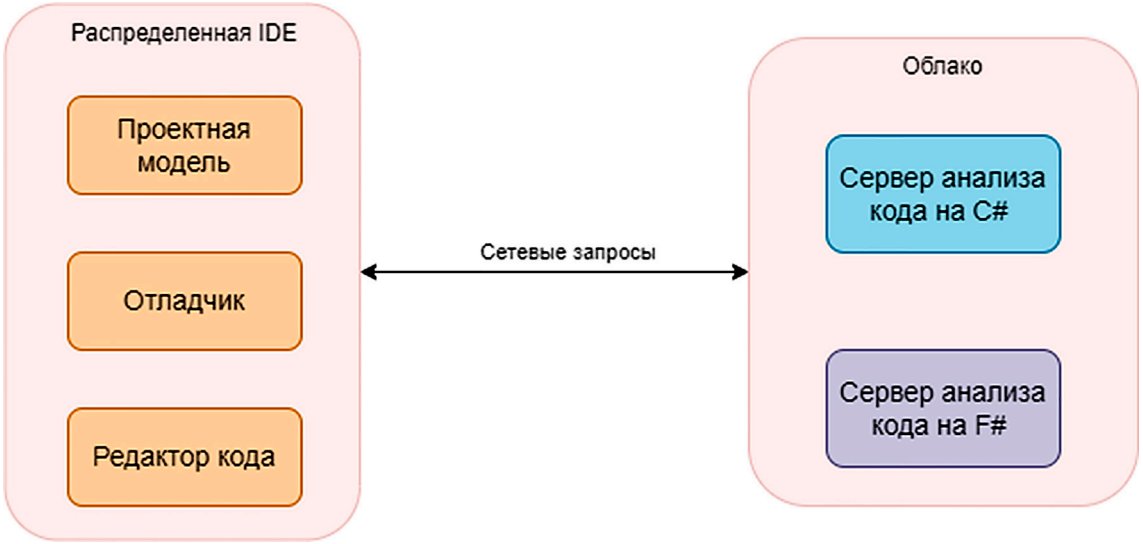


*Рис. 1. Взаимодействие распределенной IDE и удаленных подсистем*

Сильная сторона концепции распределенной IDE – возможность заранее выполнить большее количество работы по анализу кода разрабатываемой системы. Например, некоторые подсистемы могут располагаться на облачных серверах. Эти подсистемы имеют возможность заранее получить сведения об исходном коде разрабатываемой системы и сделать предварительные расчеты и анализы. Поэтому после начала работы в распределенной IDE уйдет меньше времени на открытие и анализ исходного кода, так как часть этой работы уже была завершена. Такие удаленные подсистемы могут быть как публичными, так и приватными и должны быть хорошо защищены. Взаимодействие распределенной IDE и удаленных подсистем изображено на рис. 1.

Для реализации подобной концепции недостаточно просто декомпозировать весь исходный код на подсистемы. Необходимо продумать область ответственности каждой подсистемы и реализовать корректное и согласованное взаимодействие между всеми участниками системы. Далее мы рассмотрим предлагаемую архитектуру, которая решает эти проблемы.

*Архитектура распределенных интегрированных сред разработки*

В распределенной архитектуре предполагаются три основных элемента:

1. Пользовательский интерфейс. В нем реализован весь интерфейс распределенной IDE. Содержит редактор кода и логику взаимодействия с ядром.

2. Компоненты. Подсистемы, отвечающие за реализацию функций в среде разработки. Это может быть анализ кода, сборка исходного кода или отладка.

3. Ядро. Управляет состоянием приложения и связывает компоненты с пользовательским интерфейсом. Выполняет задачи по обеспечению рабочего сеанса.

Взаимодействие пользовательского интерфейса и ядра осуществляется с помощью отправки команд, которые определяют конкретное действие, изменяющее состояние среды разработки. Например, это может быть команда с добавлением текста в конец документа или удалением файла. Ядру необходимо выполнить эту команду и передать результат всем компонентам, заинтересованным в этих изменениях. Например, после обновления текста в документе ядро может уведомить компонент анализа кода об изменениях. Компоненты также могут отправить команды на выполнение ядру.

Важно понимать, что ядро обеспечивает взаимодействие пользовательского интерфейса и компонентов, а также самих компонентов друг с другом. На рис. 2 изображена диаграмма архитектуры распределенной среды разработки.

В зависимости от конфигурации, компоненты могут запускаться как локально, так и удаленно. Конфигурация может зависеть от того, над каким проектом ведется работа. В случае коммерческого проекта компания может сконфигурировать среду разработки так, чтобы все компоненты, взаимодействующие с исходным кодом, выполнялись на специальных защищенных облачных серверах компании, без возможности прямого доступа к исходному коду. В остальных случаях все компоненты могут выполняться на персональном компьютере программиста. Плюсы такого подхода:

1. Уменьшение нагрузки на персональные компьютеры. Ресурсозатратные операции выполняются на удаленных машинах.

2. Улучшение безопасности. Снижаются риски утечки исходного кода разрабатываемых программ из-за отсутствия необходимости скачивания больших репозиториев на компьютер программиста

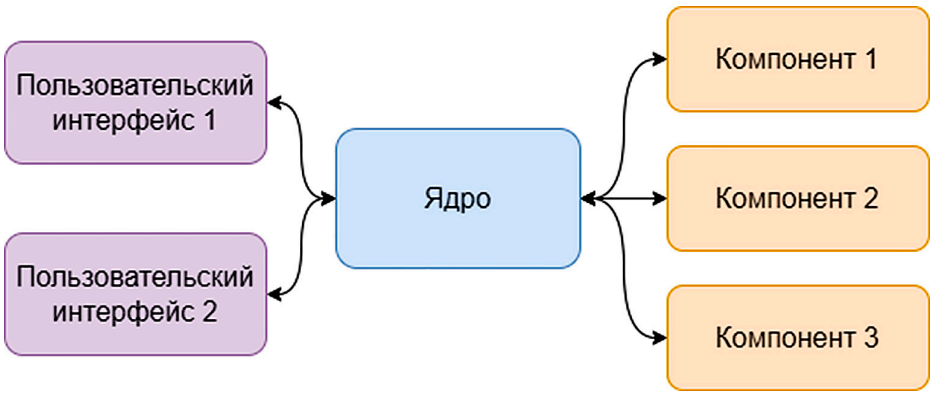


*Рис. 2. Диаграмма архитектуры распределенной среды разработки*

3. Моментальный старт. Компоненты можно заранее запустить на кодовой базе для предварительного анализа кода и выполнения других функций среды разработки. После открытия среды разработки она сразу готова к работе, без необходимости выполнения дополнительных работ.

4. Изоморфный опыт. В качестве редактора кода может выступать как настольное приложение, так и сайт в браузере.

*Пользовательский интерфейс*

Пользовательский интерфейс выполняет роль текстового редактора с дополнительными функциями. Помимо стандартных функций редактора [4], в нем также стоит реализовать лексический и синтаксический анализаторы [5] для подсветки синтаксиса и структурой проекта [6]. Данный функционал позволит использовать распределенную IDE не только как мощный инструмент для разработки приложений, но и в качестве легковесного редактора кода. При отключении от сети пользователи смогут продолжить работать, используя функционал редактора и последние данные, которые предоставили компоненты. После появления сети пользовательскому интерфейсу нужно синхронизировать обновленный исходный код с удаленными компонентами и вызвать необходимые процедуры анализа кода. У пользовательского интерфейса нет привязки к языку программирования или платформе. Поэтому он может быть реализован как в качестве настольного приложения, так и в качестве веб-сайта или мобильного приложения. Эти варианты могут комбинироваться.

*Ядро*

В распределенной среде разработки ядро выполняет задачи по реализации рабочего сеанса. Архитектура ядра является важнейшим аспектом в проектировании распределенной интегрированной среды разработки, так как является связующим звеном между всеми подсистемами и определяет их взаимодействие друг с другом. Самый важный аспект в архитектуре ядра – управление состоянием. Состояние – это данные, описывающие весь исходный код, создающийся в среде разработки. От того, как организована информация о редактируемом коде, будут зависеть все дальнейшие решения в проектировании архитектуры. В задачи входит управление состоянием приложения, синхронизация компонентов и пользовательского интерфейса. К одному может быть подключено несколько пользовательских интерфейсов для совместной разработки.

Состояние приложения может быть как изменяемым, так и неизменяемым. В распределенных интегрированных средах разработки чаще используют неизменяемое состояние. Это связано с тем, что мутабельная модель организации состояния приложения предполагает использовать примитивы синхронизации [7], что неизбежно приводит к большому количеству ошибок, связанных с многопоточностью [8]. В разработке настольных приложений существуют различные варианты управления состоянием:

1. Состояние приложения хранится в одной структуре данных, доступ к которой осуществляется с помощью примитивов синхронизации. Примитив синхронизации может быть любой, например блокировка чтения-записи.

2. Состояние приложения хранится в одной неизменяемой структуре данных. При изменении состояния поток не меняет его, а создает новое состояние на основе предыдущего и применяет к нему сделанные изменения. Ссылку на новое состояние имеет только поток, сделавший эти изменения. После внесения изменений в новое состояние, его необходимо объединить с главным. Операция объединения называется слиянием. Слияние может пройти удачно, а может завершиться ошибкой из-за конфликтов в данных. В случае конфликтов процедуру изменения состояния необходимо перезапустить.

3. Состояние приложения распределено между большим количеством классов в системе, не связанных между собой. Каждое такое место самостоятельно определяет механизмы доступа к собственной информации.

Перечислим часть задач, которые выполняет ядро:

1. Взаимодействие компонентов с пользовательским интерфейсом.

2. Взаимодействие компонентов друг с другом.

3. Оркестрация удаленных компонентов, имеющих дело с анализом кода.

Ядро поддерживает коллаборативную разработку, позволяя подключаться сразу нескольким пользовательским интерфейсам. Каждому пользовательскому интерфейсу может быть предоставлен как полный доступ ко всем функциям среды разработки, так и к ее части.

*Компоненты*

Компоненты в распределенной среде разработки отвечают за реализацию ее функционала. Ниже перечислен некоторый функционал, необходимый в интегрированной среде разработки:

– анализ и преобразование кода;

– навигация и поиск;
– сборка исходного кода;
– отладка исходного кода;
– контроль версий;
– профилировщики.

Каждый компонент может быть запущен как на машине пользователя, так и на удаленном компьютере. Взаимодействие компонентов напрямую нежелательно, такие действия лучше выполнять с использованием ядра в качестве посредника. Данная реализация позволит ядру всегда иметь актуальную информацию о приложении и информировать все заинтересованные компоненты об изменениях в состоянии.

Отдельно стоит отметить компоненты, отвечающие за реализацию анализа исходного кода. Это важные компоненты для любой IDE, отвечающие не только за анализ исходного кода, но и за его разбор и реализацию исправлений. Для этих целей используют языковые серверы и протокол языкового сервера (Language Server Protocol – LSP) в качестве взаимодействия с ними. Языковые серверы – это компоненты, реализующие поддержку конкретного языка программирования. Использование LSP позволяет унифицировать интегрирование языковых серверов в среды разработки. Это позволяет подключать языковые серверы на лету в качестве расширений среды разработки. Заметим, что реализация каждого из компонентов напрямую зависит от целевой платформы. Отладчики для C# и python будут сильно отличаться, поэтому эти компоненты не получится унифицировать.

В статье предложена архитектура интегрированной среды разработки, построенной на идее распределенных компонентов. Реализация подобной архитектуры позволяет повысить безопасность исходного кода, уменьшить требования к рабочим станциям программистов и повысить удобство разработки в распределенных командах. Также распределенная архитектура позволяет использовать компоненты среды разработки в парном программировании, предоставляя полный или частичный доступ к функциям распределенной среды разработки.

**Список литературы**

1. Кугушева Д.С. Проектирование сложного программного обеспечения с использованием микросервисной архитектуры // Инновации и инвестиции. 2020. № 5. С. 188–190.

2. Онокой Л.С., Морев Е.А. Современные подходы к проектированию архитектуры приложений // Качество. Инновации. Образование. 2022. № 5 (181). С. 90–95.

3. Харазян А.А. Особенности микросервисной архитектуры для современных приложений // Электронные информационные системы. 2023. № 2 (37). С. 40–45.

4. Саватеев М.В., Калашников В.А., Мартышкин А.И., Гурин Е.И. Создание корпоративного текстового редактора при помощи реплицируемых не конфликтных типов данных // XXI век: итоги прошлого и проблемы настоящего плюс. 2020. № 4 (52). С. 87–92.

5. Пырнова О.А., Никоноров Д.П., Шарифуллина А.Ю. Разработка статического анализатора программного кода // Научно-технический вестник Поволжья. 2023. № 12. С. 522–525.

6. Ванясин Н.В., Сидоркина И.Г., Поляков В.И. Архитектура интегрированной среды разработки программного обеспечения с поддержкой структурного редактирования // Научно-технический вестник информационных технологий, механики и оптики. 2019. № 6. С. 1079–1085.

7. Косяков М.С., Тараканов Д.С. Сравнительный анализ реализаций спин-блокировок // Программные продукты и системы. 2018. № 4. С. 763–767.

8. Егоров В.Б. Особенности многоядерности и многопоточности в сетевых процессорах // Системы и средства информатики. 2020. № 1. С. 82–92.