ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
ИНСТИТУТА КОМПЬЮТЕРНЫХ НАУК И КИБЕРБЕЗОПАСНОСТИ
ВЫСШАЯ ШКОЛА ПРОГРАММНОЙ ИНЖЕНЕРИИ

Отчет о прохождении преддипломной практики на тему: «Разработка плагина Jenkins для визуализации статистики работы сборок Jenkins»

Кухто Андрея Денисовича, гр. з5130903/90301

Направление подготовки: 09.03.03 Прикладная информатика.
Место прохождения практики: СПбПУ, ИКНК, ВШПИ г. Санкт-Петербург, ул. Обручевых, д. 1, лит. В.
Сроки практики: с 19.12.2023 по 09.01.2024.
Руководитель практической подготовки от ФГАОУ ВО «СПбПУ»: Пархоменко Владимир Андреевич, ст. преподователь ВШПИ, .
Оценка: _____

Руководитель практической подготовки
от ФГАОУ ВО «СПбПУ» В.А. Пархоменко

Обучающийся А.Д. Кухто

Дата: 09.01.2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня разработка информационных систем достаточно сложный процесс, который состоит из нескольких этапов: анализ требований заказчика, проектирование системы, разработка, тестирование и доставка приложения потенциальному заказчику.

Для упрощение процесса работы в настоящий момент широко применяются практики DevOps одной из которых является CI/CD - непрерывная итеграция, сборка и доставка. Существует множество средств CI, которые применяются в промышленной разработке: TeamCity, Jenkins, Gitlab CI и другие.

Одним из лучших средств CI, в котором доступно много функций ”из коробки” является TeamCity компании Jetbrains. TeamCity - мощный и сложный инструмент, который использовался крупными ИТ компаниями в промышленной разработке до 2022 года. Одним из главных недостатков TeamCity является то, что это платное решение, лицензия обходится ИТ компания достаточно дорого, также недостатком является то, что компания JetBrains покинула ИТ сектор РФ. Для того, чтобы преодолеть данные проблемы ИТ компании РФ начали поиск бесплатных средств с открытым исходным кодом. Одним из таких средств является Jenkins - средство CI, которое всегда пользовалось популярностью у разработчиков при локальной разработке решений с открытм исходным кодом.

Jenkins обладает меньшим функционалом в сравнении с TeamCity, но имеет много подключаемых плагинов, которые могут помочь заменить или даже улучшить те функции, которые требуется разработчикам в процессе тестирования, сборки и доставки приложений.

Актуальность исследования. На данный момент в Jenkins нет плагина, который полностью заменяет модуль визуализации статистики Build Configuration Statistics. Часть плагинов реализует частичный функционал модуля TeamCity, подробнее о недостатках таких средств будет описано в сравнительном анализе и обзоре аналогов. Этот плагин/модуль требуется для того чтобы отслеживать состояние отдельных конфигураций сборки с течением времени, плагин собирает статистические данные по всей истории сборки и отображает их в виде наглядных диаграмм.

В данной работе будет разработан плагин, который обеспечит визуализацию статистики работы сборок.

Объект исследования — сборки Jenkins с их метриками.

Предмет исследования — технологии разработки плагинов для Jenkins.

Цель - разработать плагин Jenkins для визуализации статистики сборок Jenkins.

Задачи:

A. Изучить систему CI/CD Jenkins и работу и характеристики сборок Jenkins.
B. Изучить методы разработки плагинов Jenkins.
C. Провести проектировани плагина и описать архитектуру.
D. Разработать код плагина.
E. Провести тестирование и аппробацию плагина.

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В главе проведен анализ предметной области:

A. Исследованы и описаны особенности CI/CD инструментов.
B. Проведен обзор и сравнительный анализ средств CI/CD.
C. Проведен обзор и сравнительный анализ плагинов Jenkins по визуализации статистики сборок.
D. Описаны требования к разработке.

1.1. Особенности CI/CD систем

CI/CD (Continuous Integration, Continuous Delivery — непрерывная интеграция и доставка) — это технология автоматизации тестирования и доставки/развертывания готового приложения заказчику [4]. Данная технология стала неотъемлемой состовляющей DevOps методологии и помогает сократить временные рескрсные затраты в процесе современного жизненного цикла приложения, когда до заказчика изначально доходит MVP (minimum viable product) продукт, а затем дорабатывается с учетом новых требований заказчика, т.е. идет непрерывная разработка новых версих продукта.

Преимущества CI/CD подхода [24]:

- упрощение разработки - позволяет разработчикам распределять приоритеты и сконцентрироваться на самых важных аспектах;
- улучшение качества кода - качество кодапроверяется до того, как он достигнет среды тестирования, проблемы в коде могут быть выявлены на ранних стадиях;
- более короткие циклы тестирования - меньший объем кода для проверки;
- более простой мониторинг изменений - меньший объем кода для проверки;
- более короткие циклы тестирования - становиться проще определить проблемы в процессе развертывания;
- более лекгий откат - меньшие усилия для отката приложения к предыдущей версии при возникновении проблем в новой версии.

Этапы разработки и принцип CI/CD подхода можно отразить с помощью рисунка 1.

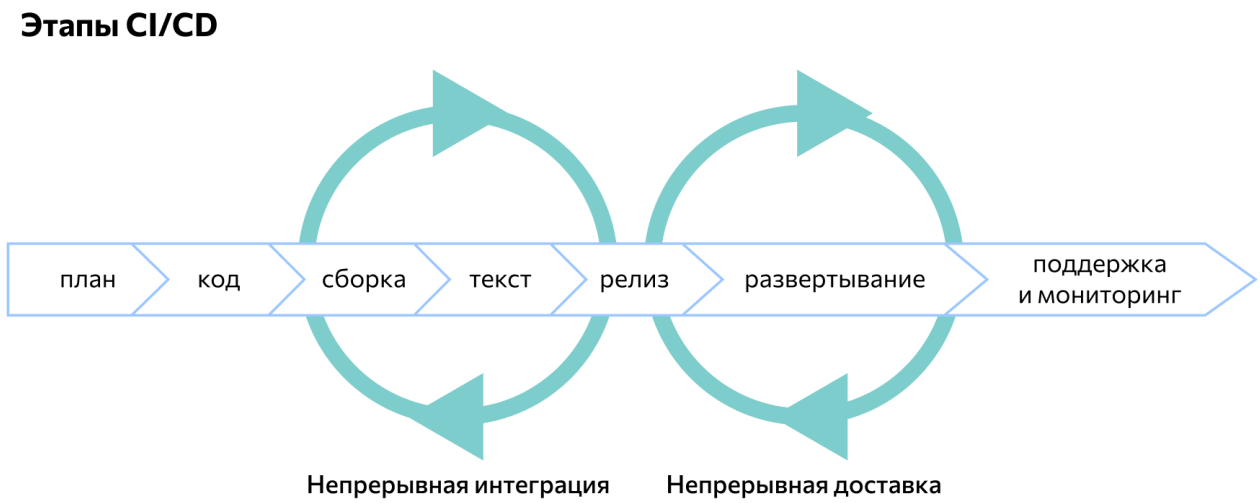

Рис.1.1. Цикл CI/CD

1.2. Обзор и сравнительный анализ средств CI/CD

На данный момент существует множеств инстркментов CI/CD, которые обладают своими преимуществами и недостатками, были выделены самые распространенные системы:

- TeamCity;
- Jenkins;
- GitLab CI;
- CircleCI;
- Bamboo.

Jenkins — это автономный сервер автоматизации с открытым исходным кодом, который можно использовать для автоматизации всех видов задач, связанных со сборкой, тестированием, доставкой или развертыванием программного обеспечения [12].

TeamCity - это сервер CI от компании Jetbrains [16], который позволяет запускать параллельные сборки одновременно на разных платформах и средах, а также настраивать статистику по продолжительности сборки, уровню успешности, качеству кода и пользовательским метрикам.

GitLab CI - сервер CI от компании Gitlab [7], которая также предоставляет одноименный репозиторий Git. GitLab CI/CD может обнаруживать ошибки на ранних этапах цикла разработки и гарантировать, что весь код, развернутый в рабочей среде, соответствует установленным стандартам кода.

Критерий	Jenkins	TeamCity	GitLab CI	CircleCI	Bamboo
Открытый исходный код	+	-	-	-	-
Цена	Бесплатно	от 45$ в месяц	от 5$ в месяц	от 15$ в месяц	от 1200$ в год
Встроенный функционал	2/5	5/5	4/5	4/5	4/5
Интеграции	5/5	4/5	4/5	3/5	3/5
Поддержка репозиториев Git	Любой репозиторий	GitHub, GitLab, Bitbucket	GitHub, GitLab, Bitbucket	GitHub, GitLab, Bitbucket	Любой репозиторий

CircleCI - сервер CI [5], который позволяет настроить для эффективной работы очень сложных конвейеров кэширование, кэширование уровня Docker и классы ресурсов для работы на более быстрых машинах.

Bamboo — это инструмент непрерывной интеграции и доставки [2], который связывает автоматизированные сборки, тесты и выпуски в единый рабочий процесс.

Особое внимание следует уделить критерию OpenSource, этот критерий является достаточно важным с учетом, того, что многие компании после 2022 года ушли из РФ, тем самым стали либо недоступны, либо прекратили лицензировании и стали менее безопасными, т.к. новые версии продуктов больше недоступны и проблемы с безопасностью и другими дефектами не будут исправлены/доступны на территории РФ. Также критерий важен тем, что даже при наличии действия продуктов компаний, они обходилось крупным ИТ-компаниям достаточно дорого.

Критерий интеграция показывает сколько можно подключить к системе плагинов и интеграций со сторонними сервисами, а встроенный функционал сколько функций из коробки поддерживает, то или иное средство, наличие инструментов, которые позволят облегчить работу.

Среди всех средств особо ярко выделяется Jenkins, поскольку он является бесплатным и с открытым исходным кодом, а также обладает большим количество интеграций и плагинов, которые постоянно пишутся, что позволяет устранить основной его недостаток по наличию встроенных функций. После 2022 года в РФ это стал самый востребованный инструмент для настройки CI конвейеров, и обосновывает важность разработки плагина для устранения недостатков функциональности, которые есть в других средствах.

1.3. Обзор и сравнительный анализ плагинов Jenkins по визуализации статистики сборок

В данной работе производится разработка плагина для визуализации метрик сборки Jenkins, основание для разработки является отсутсвие плагина, который полностью визуализирует метрики сборок в Jenkins, аналогично встроенному модулю в Teamcity. Многие российские ИТ-компании широко использовали TeamCity, этот модуль позволял отслеживать состояние отдельных конфигураций сборки с течением времени,собирать статистические данные по всей истории сборки и отображать их в виде наглядных диаграмм. В данной работе будет разработан плагин для воссоздания это модуля в Jenkins, с дополнительным функционалом, которого не хватало в TeamCity.

Сначала рассмотрим уже разработанные плагины визуализации и их недостатки и преимущества в сравнении с разрабатываемым решением.

Build Monitor Plugin - плагин, который обеспечивает наглядное представление статуса выбранных заданий Jenkins. Отображает состояние и ход выполнения выбранных заданий.

Global Build Stats Plugin - плагин, который позволит собирать и отображать глобальную статистику результатов сборки, а также позволяющий отображать глобальную тенденцию сборки Jenkins/Hudson с течением времени.

Build Time Blame - плагин, который сканирует вывод консоли на наличие успешных сборок и генерирует отчет, показывающий, как эти шаги повлияли на общее время сборки. Это предназначено для того, чтобы помочь проанализировать, какие этапы процесса сборки являются подходящими кандидатами на оптимизацию.

После проведения сравнения аналогичных решений, были выявлены преимущества разрабатываемого плагина, которые обосновывают его разработку, это отсутсвие у данных плагинов функционала по визуализации Artifacts Size, Time Spent in queue, Success Rate истории сборок, а также наличие отслеживания аномальных результатов метрик, которое позволит определить проблемные сборки, у которых возникают временные проблемы с процессом CI/CD. Также данные плагины не предлагают динамическое изменение графиков по мере изменения временного интервала или установления фильтров.

Критерий	Build Monitor Plugin	Global Build Stats Plugin	Build Time Blame	Плагин разрабатываемый
Наличие отслеживания аномальных результатов метрик	-	-	-	+
Открытый исходный код	+	+	+	+
Визуализация времени выполнения и статуса последней сборки	+	+	- (только время)	+
Визуализация Success Rate истории сборок	-	+/- (в Teamcity гистаграммы, которые показываею процентное соотношение нагляднее)	-	+
Визуализация Build Duration истории сборок (в числе average)	-	+	+	+
Визуализация Time Spent in queue	-	-	-	+
Визуализация Test Count	-	-	+	+
Визуализация Artifacts Size	-	-	-	+
Отображение всех графиков на одной странице по одному диапазону времени для наглядного отображения всех метрик в один момент и во времени	-	+	-	+

1.4. Требования к разработке

Поскольку разрабатываемый плагин является аналогом модуля статистики сборок в TeamCity, то функционал должен быть соотествующим. В первую очередь должна производиться визуализация метрик сборок с помощью графиков и диаграм, а именно:

- визуализация метрики success rate - процент успешности сборок, который будет показывать сколько сборок завершилось успешно;
- визуализациия метрики Build Duration - время выполнения сборок, в том числе должен быть доступен фильтр на добавления в график упавших сборок, а также возможность вычислять не только суммарно время сборок, а также среднее время всех сборок за определенный интервал времени;
- визуализациия метрики Time Spent in queue - время проведенное в очереди сборок, в том числе среднее время, вычисляемое аналогично Build Duration;
- визуализациия метрики Test Count - количество выполненных тестов в сборке, в том числе количество выполненых тестов в упавших сборках, если таковые успели выполниться;
- визуализациия метрики Artifacts Size - размер созданных во время сборок артефактов, в том числе средний размер за определенный интервал времени, а также учет артефактов, которые умпели создаться в сборках до падения.

На всех графиках и диаграммах должна быть возможность выбора значения из выпадающего списка интервала времени, за который будет производиться сбор статистики за день, месяц, квартал, неделю, год и за весь промежуток времени, т.е. если, например, был выбран промежуток времени месяц, то должен выполняться следующий набор действий:

A. Должна собираться информация о требуемой метрики у всех сборок.
B. Производиться фильтрация сборок - т.е. отбираться только сборки за последний месяц (в том числе упавшие, если был выбран данный чекбокс).
C. Полученные сборки должны группппироваться по дням т.е. на итоговом графике должно быть 30/31 точка или столбца.
D. Если необходимо производиться вычисление среднего среди всех сгруппированных за день метрик сборок.
E. Отображение всей информации о матриках сборки на одном графике или диаграмме.

Также все графики должны располагаться друг под другом на однйо странице, что может наглядно показать (если на каждом графике был выбран один период), все вычесленные метрики за один период, например при выборе месяца все перечисленные метрики будут отображены на странице и можно будет увидеть, что происходило, например вчера по результатам запуска всех сборок.

Помимо реализации перечисленных функция, которые полностью аналогичны функциям TeamCity, плагин будет вычислять аномальные значения за определенный период - т.е. можно будет наглядно увидеть, например, в какие дни произошли сбои в работе сборок, это может быть например слишком большой размер артифактов у одной сборки за какой-то промежуток времени.

Также было принято решения добавить анализ данных, чтобы делать предположение, о том какими метриками будет обладать следующая запущенная сборка, при вычислении данного значения должно быть расчитаны веса каждой сборки/сборок по графику за определенный период, и если сборка была собрана, например, месяц назад - она должна иметь меньший вес, чем сборка, собранная вчера.

Также к разработке будет предъявлено требование об удобстве интерфейса: все графики должны быть удобными, не перегруженными информацией, а также интерфейс должен быть интуитивно понятно, чтобы данный плагин не усложнял восприятие собранной статистики сборок и не вызывал желание воспользоваться другим плагином или разработать другой более удобной, или отказаться от идеи смотреть статистику по сборкам.

1.5. Выводы

По всем описанным выше разделам можно прийти к выводу, что данный плагин актуален для ИТ-компаний, которые ранее отдавали предпочтению многофункциональному инструменту TeamCity, в котором уже были все необходимые для работы функции, особенно это актуально для компания в РФ, но также может понадабиться и другим компания, которые приняли решение отказаться от TeamCity в пользу Jenkins из-за больших денежных затрат на лицензию. Также будет реализованы дополнительный функционал по сравнению с модулем TeamCity, что даст преимущества не только в цене. После проведенного обзора аналогичных решений становится понятно, что сейчас в Jenkins нет полнофункциональной замены модуля статистики TeamCit, также необходимо учесть и визуальную со-

ставляющую, чтобы при установке данного плагина разработчики выбирали его не только из-за отсутсвия другого решения.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПЛАГИНА

В данной главе будет проведено проектирование разрабатываемого плагина: будет описана архитектура построения плагинов в Jenkins, а также архитектура разработки, будут выбраны инстркменты разработки, а также расмотрена функциональная модель системы. Поскольку плагин разрабатывается для системы Jenkins, то отладку и тестирование будем проводить в этой системе.

2.1. Языки программирования

Для программирования плагина будет использоваться язык Java. Поскольку Jenkis написан на Java, то все плагины необходимо писать на том же языке. Это является главным минусом, а возможно и сложностью при разработке плагинов на Jenkins, поскольку ограничивает свободу разработчика.

Есть возможность разработки плагина с использованием языка программирования Groovy. Groov это динамический язык с возможностями статической типизации и статической компиляции для платформы Java[8], нацеленный на повышение производительности разработчиков, который плавно интегрируется с любой программой Java.

Недостатком такого выбора является то, что абсолютное большинство плагинов написано на чисто Java, а значит сообщества и поддержка при разработке на Java будет значительно большей. Также в сравнии с Groovy, Java обладает большей производительностью[15], статической типизацией и подходит для разработки приложений в парадигме ООП.

Java — это язык высокого уровня, который можно охарактеризовать следующими словами: объектно-ориентированный, многопоточный, динамический, высокпроизводительный и безопасный [9]. Java используется для разработки высоконагруженный информационных систем, мобильных приложений, плагинов, десктопного ПО и др. К преимуществам Java также можно отнести компилируемость, что обеспечивает высокое быстродействие.

Java будет использоваться для программиования ядра плагина и бизнес–логики. Также для программирования графических компонентов, графиков и диаграмм будет использоваться язык программирования JavaScript. JS - это легковесный, интерпретируемый или JIT-компилируемый, объектно-ориентированный язык [10], основное предназначение которого выполнять сценарии на веб-страницах, что необходимо при разработке плагина, результаты которого отображаются на веб-страницах.

Помимо прочего, для стилизации компонентов веб-интерфейса будет использоваться язык каскадных таблиц стилей CSS [6], который позволит настроить удобное отображение и позиционирование элементов на странице плагина Jenkins.

Верстка страниц будет осуществляться с помощью инструмента Jelly - все разрабатываемые плагины используют данный инструмент в Jenkins, поскольку с ним можно легко интегрировать Java, XML и JS. Jelly — это средство для преобразования XML в исполняемый код, это механизм сценариев и обработки на основе Java и XML [11]. В Jelly можно вызывать функции Java, использовать такие синтакситеские конструкции, как циклы, условия и переменные, также он позволяет легко обратиться к объектам в Java.

2.2. Инструменты сборки

В качетве инструмента сборки проекта был выбран Maven, который можно использовать для создания и управления любым проектом на основе Java. Maven решает несколько проблем [13]:

- упрощение процесса сборки;
- обеспечение единой системы сборки;
- предоставление информации о проекте;
- упрощение работы с зависимостями, включая их автоматоическое обновление.

Абсолютное большинство разработанных плагинов для Jenkins использует Maven, поскольку Maven предоставляет удобные архетипы для начала разработки плагинов, что делает использование того же Gradle не рациональным.

2.3. Библиотеки

Поскольку проект предполагает использовани графиков и диагармм, то необходимо было выбрать инструмент для работы с графиками в Jenkins и Java, который позволит отображать графики прямо на странице задания Jenkins. В качестве этого иструмента была выбрана библиотека Chart.js, которая на данный момент является самой популярной JavaScript библеотекой по оценкам GitHub и загрузок npm [3]. К преимуществам данной библиотеки можно отнести:

- у Chart.js очень подробная документация;
- отрисовка canvas делает Chart.js очень производительным, особенно для больших наборов данных и сложных визуализаций;
- строит отзывчивый интерфейс - перерисовывает диаграммы при изменении размера окна для идеальной детализации масштаба.

2.4. Архтектура Jenkins

Перед объяснением построения архитектуры плагинов Jenkins, необходимо привести схему архитектуры Jenkins, где будет отображено место разрабатываемых плагинов в CI системе (рисунок 2.1). Установленные плагины Jenkins-CI, а также локальные сценарии и приложения выполняются на сервере Jenkins-CI и предоставляют расширяемый набор функций управления и обработки данных [1].

Архитектура плагинов использует точки расширения, которые, предоставляют разработчикам плагинов возможности реализации для расширения функциональности системы Jenkins [17]. Точки расширения автоматически обнаруживаются Jenkins во время загрузки системы.

В разрабатываемомом плагине реализация будет происходить через класс Action. Actions являются основным строительным блоком расширяемости в Jenkins: их можно прикреплять ко многим объектам модели, хранить вместе с ними и при необходимости добавлять в их пользовательский интерфейс.

Помимо класса Action для того чтобы создать временные действия, которые будут прикреплены к заданию Jenkins будет использован класс TransientActionFactory, который позволяет создавать действия, которые будут отображаться на страницах Jenkins только при наличии соответствующего объекта - задания.

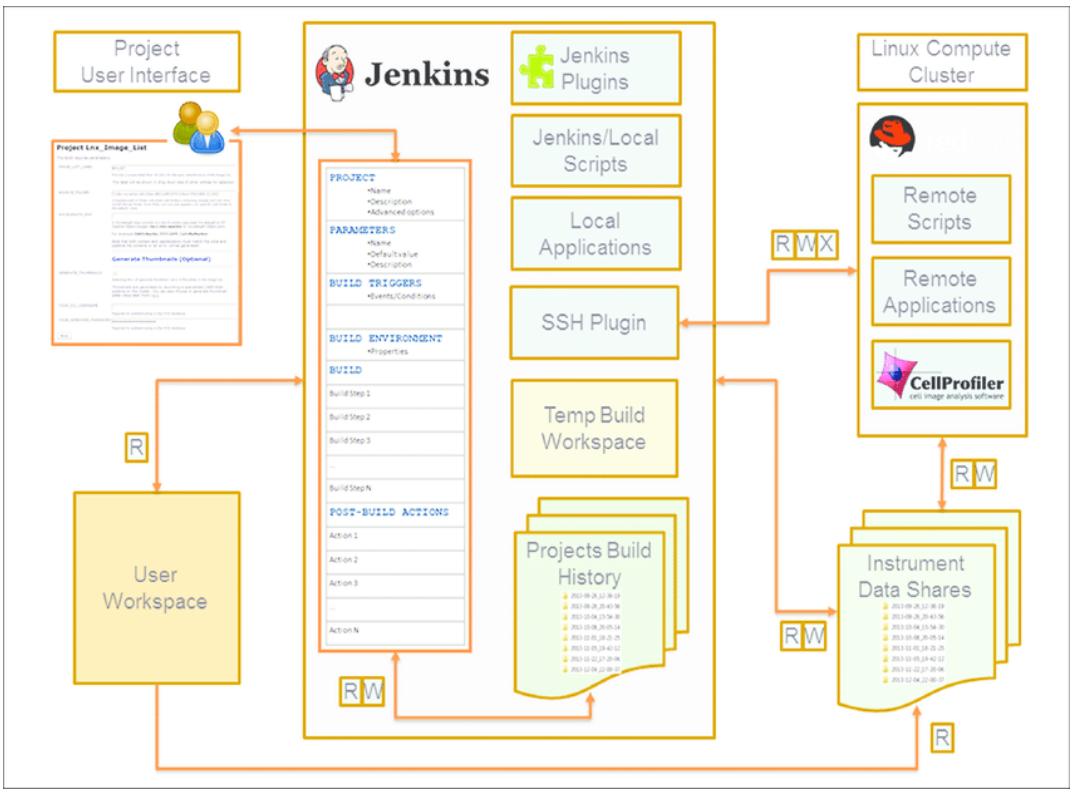

Рис.2.1. Архитектура Jenkins

Разработка будет выполняться в объектно-ориентированной парадигме, т.е. приложение будет разбито на классы, будет применяться наследование, полиморфизм и инкапсуляция. Все классы, которые будут разработаны для плагина описаны в приложении П1.1. При рассмотрении диаграммы необходимо отметить, что два класса являются встроенными в Jenkins, это TransientActionFactoryб который позволяет добавлять действия к любому типу объекта, а также интерфейс Action - добавленный к объекту модели, создает дополнительное подпространство URL-адресов под родительским объектом модели, через которое он может взаимодействовать с пользователями. Actions также способны открывать доступ к левому меню в интерфейсы Jenkins, по которому обычно производится навигация при конфигурировании сборки. Для удобства использования плагина, предполагается добавить дополнительную ссылку в меню слева, для перехода на страницу визуализации метрик, а также динамически обновлять страницу при изменении параметров и фильтров, что и обосновывает использование данных встроенных классов.

Основная часть остальных классов требуется для работы с определенной метрикой статистики выполнения сборок Jenkins, что следует из их названия. Также будет разработан дополнительный класс DateTimeHandler, который позволит создать методы для удобной работы с датой и временем, что необходимо поскольку

будет производиться преобразования одних типов дат к другим, сравнение дат между собой, а также получение определенных частей дат.

Функциональная модель в нотации Idef0 отображена в приложении П2.1.-3. Основное внимание на диаграмме уделяется визуализации статистики сборок, поскольку это изначально является целью разработки. Также там будут отражены дополнительные функции такие как фильтрация.

Диаграмма вариантов использования, показывающая функционал плагина отображена в приложении П3.1. На данной диаграмме основное внимание также уделяется процессу визуализации статистики метрик сборок. Основное действующее лицо одно - это пользователь системы, который запускает сборки и работает в CI системе, это может быть любой участник команды, который задействован в разработке, тестировании, доставке и внедрению приложения. В данном случае все эти роли представлены на диаграмме как разработчик.

ГЛАВА 3. РАЗРАБОТКА ПРОТОТИПА ПЛАГИНА

При разработке плагина надо было учитывать, что требуется отображать все графики на одной странице задания друг под другом, поскольку при выборе одно периода, например, месяца, будет получена сводная информация по каждой сборке или нескольких сборок запущенных в один день. Графики отображаются посредством переходна на соотвествующую ссылку, оставляя при этом пользователя в том же задании (странице с результатами последних сборок).

В интерфейсе у каждого графика были реализованы те дополнительные функции отображения, которые могут быть применины к визуализируемой метрике: отобразить статистику по упавшим сборкам/тестам, усреднить метрику.

Интерфейс страницы плагина с графиками в системе Jenkins на странице задания показан на рисунке 3.1.

Основное взаимодействие с графиками будет производить через меню, которое есть напротив каждого графика со своими параметрами, отображенном на рисунке 3.2:

При взаимодействии с раскрывающимся списком должен вызываться Java метод, который пересчитает и отфильтрует необходимые сборки Jenkins и динамически отобразит результаты по выбранными периоду, также динамически должна производить обработка метрик сборок, при выборе основной метрики ни

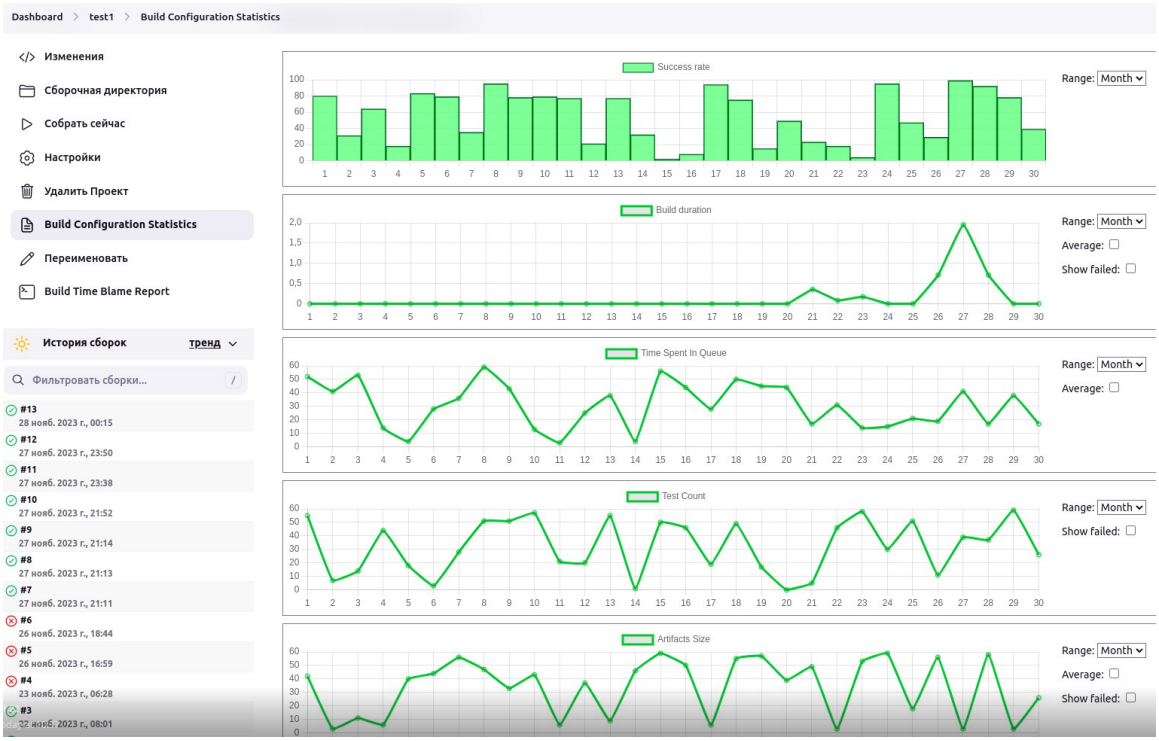

Рис.3.1. Интерфейс плагина Jenkins

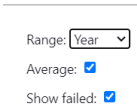

Рис.3.2. Интерфейс элементов управления

как суммы всех значений, а как среднего, а также включение в графики данных об упавших сборках, при выборе соотвествующих чекбоксов.

Интерфейс изменения навигационной панели отображен на рисунке 3.3. В данном случае видно что изменения видны при открытии конфигурации конкретной сборки, т.е. не надо будет переключаться между окном плагина и сборкой для визуализации метрик, при открытии данного пункта меню, также происходит изменения URL, с которым в дальнейшем и происходит взаимодействие.

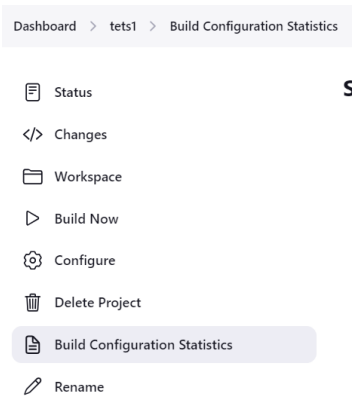

Рис.3.3. Интерфейс элементов управления

Код плагина представлен в приложении 4.

ГЛАВА 4. ТЕСТИРОВАНИЕ И АППРОБАЦИЯ ПЛАГИНА В JENKINS

В главе описано проведенное тестирование и аппробация плагина:

A. Описаны методы тестирования.
B. Проведена аппробация плагина в системе Jenkins.
C. Разработан код тестирования плагина.

4.1. Методы тестирования

Тестирование программного обеспечения — обширное понятие, которое включает планирование, проектирование и, собственно, выполнение тестов [25]. В процессе CI/CD производится непрерывное тестирование разработанного кода, а также тестирование разработанного приложения. Сам плагин является частью CI/CD процессе, но также требует тестирования корректной работы своей функциональности, тестирование разработанного кода, а также тестирования на сооветсвие исходным требования, которые были предъявлены к разработке в главе 1.

Существует множество методов тестирования и техник тест дизайна, в процесе анализ функциональных требования были отобраны те, которые наиболее релевантны для разработанного плагина Jenkins.

Будет проведено как ручное, так и автоматизированное тестирование плагина. Ручное тестирование поможет выявить нетипичные тест-кейсы, которые не покрываются автоматизированными тестами.

Ручные тесты будут проведены методом черного ящика. Данный метод это процедура получения и выбора тестовых случаев на основе анализа функциональности и технического задания, без применения знаний о внутреннем устройстве системы [22].

Были составлены следующие тест-кейсы:

В данном случае тест-кейсы были описаны с помощью техники тест-дизайна Матрица трассировки. Если обратиться к определению, то матрица трассировки — двумерная таблица, содержащая соответствие функциональных требований и подготовленных тестовых сценариев [21], а на пересечении столюбцов и строк ставится метка, о том, что данное требование покрывается данным тест-кей-

№	Описание	Ожидание
1	Проверка выбранного периода на графике Success Rate (месяц)	Количество дней соотвествует прошлому месяцу, на каждый день отображены корректные значения процента успешности сборок
2	Проверка выбранного периода на графике Build Duration (неделя) с учетом выбора настройки среднего значения и упавших сборок	Количество дней 7 в соотвествии со всеми днями недели, на каждый день отображены корректные значения среднего времени продолжительности сборок, учтены упавшие сборки
3	Проверка выбранного периода на графике Time Spent in queue (квартал) с учетом выбора настройки среднего значения	Количество кварталов 4 в соотвествии с кварталами года, на каждый квартал отображены среднии значения проведенного в очереди времени сборок
4	Проверка выбранного периода на графике Test Count (год) с учетом выбора настройки упавших сборок	Количество месяцев 12 соотвествует прошедшему году, на каждый месяц отображены корректные значения количества тсетов, с учетом тестов выполненыных на упавших сборках
5	Проверка выбранного периода на графике Artifacts Size (день)	Количество часов 24 соотвествует всем часам прошедшего дня, на каждый час отображены корректные значения размера итоговых артефактов полученного по результатм задания
6	Проверка корректного отображения при выборе периода ALL	Отображены сборки со всего периода прошедшео, информация поделена на равное части
7	Проверка корректного отображения аномальных значений	Отображены номера сборок, возде каждого графика, у которых аномальное значений метрики соотвествующей графику
8	Проверка корректного расчета предугаданной ’следущей’ сборки	Метрики предугаданной сборки расчитываются корректно для каждого графика

сом. В случае данной работы из соображений удобства и оптимизации тестовой документации, было принято решение модифицировать матрицу трассировки и совместить с подробным описание в формате чек листа: для оптимизации идет проверка, что каждый тип графика-метрики (Success Rate) корректно себя ведет на определенном периоде (неделя), таким образом не придется проверять, каждый график на каждом пероде при каждой доработке кода продукта.

Данные тест-кейсы оптимизированы, поскольку в соотвествии с пирамидой тестирования [23] ручные UI кейсы, находятся в самой верхней ее части и не должны занимать достаточно большое место в системе тестирования. С дрйгой стороны для улучшения покрытия требования, предъявляемых к продукты должны использоваться гораздо в большем объеме unit-тесты, т.е. тесты в которых самые маленькие компоненты системы - модули (модули, классы, методы), индивидуально проверяются на предмет правильной работы [18].

При написание юнит тестов используется метод белого ящика. Данный метод предоставляет тестировщику полное знание тестируемого приложения, включая доступ к исходному коду и проектной документации [20], т.е. тестирование происходит на основе знания исходного кода, таким образом, юнит тестирование методом белого ящика поможет нам достаточно широко покрыть все модули плагина.

Код юнит тестов приведен в приложении 5.

Также для автоматизации тесстиования UI части плагина, был применен Selenium web driver и язык программирования python. WebDriver управляет браузером, как это делает пользователь, с использованием сервера Selenium [19]. Данные тест-кейсы будут в автоматическом режиме проверять реакцию элементов веб интерфейса на действия пользователя.

Код UI тестов приведен в приложении 6.

4.2. Аппробация плагина

Аппробация плагина будет проводится в системе CI Jenkins для которой и был разработан плагин визуализации. Для того чтобы провести аппробацию плагина на локальном сервере дженкинс, запущенном на локальном или удаленном ПК, потребкется произвести несколько операций. Для начала, нужно будет склонировать репозитоорий с кодом плагина на GitHub, перейти в папку проекта и выполнить [14] Maven mvn install и скопировать .hpi в папку /plugins/.

Затем потребуется на запущенном серере дженкинс перейти в Управление Jenkins и среди доступных плагинов выбрать Build Configuration Statistics, установить, после чего напротив каждой сборки дженкинс в боком меню, откуда можно запустить и отредактировать сборку, появится пункт меню Build Configuration Statistics, при нажатии на которой должны отобразиться все графики с собранной статистикой по метрикам каждого задания в Jenkins.

Для того чтобы графики отображали какие-то данные, необходимо сначала сгенерировать сборки разной длительности, статусов, с разным количеством тестов и размером артефактов.

ЗАКЛЮЧЕНИЕ

В результате проведенной работы был разработан прототип плагина для визуализации статистики сборок Jenkins. Были выбраны средства и инструменты разработки, спроектирована архитектура плагина, описаны функциональные возможности, а также разработан программный код и интерфейс плагина.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Jenkins-CI, an Open-Source Continuous Integration System, as a Scientific Data and Image-Processing Platform / I. Moutsatsos [и др.] // Journal of Biomolecular Screening. — 2016. — Т. 22. — С. 1087057116679993. — DOI 10.1177/1087057116679993.

2. Bamboo Docs. — URL: https://confluence.atlassian.com/bamboo/bamboo-documentation-289276551.html (visited on 20.11.2023).

3. Chart.js Documentation. — URL: https://www.chartjs.org/docs/latest/ (visited on 20.11.2023).

4. CI/CD. — URL: https://blog.skillfactory.ru/glossary/ci-cd (visited on 20.11.2023).

5. Circle CI Docs. — URL: https://circleci.com/docs/about-circleci/ (visited on 20.11.2023).

6. CSS Documentation. — URL: https://developer.mozilla.org/en-US/docs/Web/CSS (visited on 20.11.2023).

7. GitLab CI Docs. — URL: https://docs.gitlab.com/ee/ci/ (visited on 20.11.2023).

8. Groovy Docs. — URL: https://groovy-lang.org/documentation.html (visited on 20.11.2023).

9. Java Docs. — URL: https://docs.oracle.com/en/java/ (visited on 20.11.2023).

10. JavaScript Documentation. — URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (visited on 20.11.2023).

11. Jelly Documentation. — URL: https://commons.apache.org/proper/commons-jelly/ (visited on 20.11.2023).

12. Jenkins Documentation. — URL: https://www.jenkins.io/doc/ (visited on 20.11.2023).

13. Maven Documentation. — URL: https://maven.apache.org/what-is-maven.html (visited on 20.11.2023).

14. *Pathare A.* Tutorial: Developing Complex Plugins for Jenkins. — 2022. — URL: https://www.velotio.com/engineering-blog/jenkins-plugin-development (visited on 22.12.2023).

15. *Puzhevich V.* Groovy vs Java: Detailed Comparison and Tips on the Language Choice. — 2020. — URL: https://scand.com/company/blog/groovy-vs-java/ (visited on 20.11.2023).

16. TeamCity Docs. — URL: https://www.jetbrains.com/help/teamcity/teamcity-documentation.html (visited on 20.11.2023).

17. The architecture of Jenkins plugins. — URL: https://subscription.packtpub.com/book/programming/9781784390891/10/ch10lvl1sec62/the-architecture-of-jenkins-plugins (visited on 20.11.2023).

18. Unit testing. — URL: https://www.techtarget.com/searchsoftwarequality/definition/unit-testing (visited on 22.12.2023).

19. WebDriver Docs. — URL: https://www.selenium.dev/documentation/webdriver/ (visited on 20.12.2023).

20. What is White Box Testing? — URL: https://www.checkpoint.com/cyber-hub/cyber-security/what-is-white-box-testing (visited on 22.12.2023).

21. . — URL: https://habr.com/ru/companies/simbirsoft/articles/412677/ (visited on 22.12.2023).

22. . « ». — 2017. — URL: https://quality-lab.ru/blog/key-principles-of-black-box-testing/ (visited on 22.12.2023).

23. . — URL: https://habr.com/ru/articles/672484/ (visited on 22.12.2023).

24. . CI/CD : // Cloud Networks. — 2021. — URL: https://cloudnetworks.ru/analitika/podhodit-ci-cd-vashemu-biznesu-plyusy-i-minusy-konvejerov/ (visited on 20.11.2023).

25. : , — URL: https://habr.com/ru/companies/habr_career/articles/517812/ (visited on 22.12.2023).

Приложение 1

UML диаграмма классов плагина

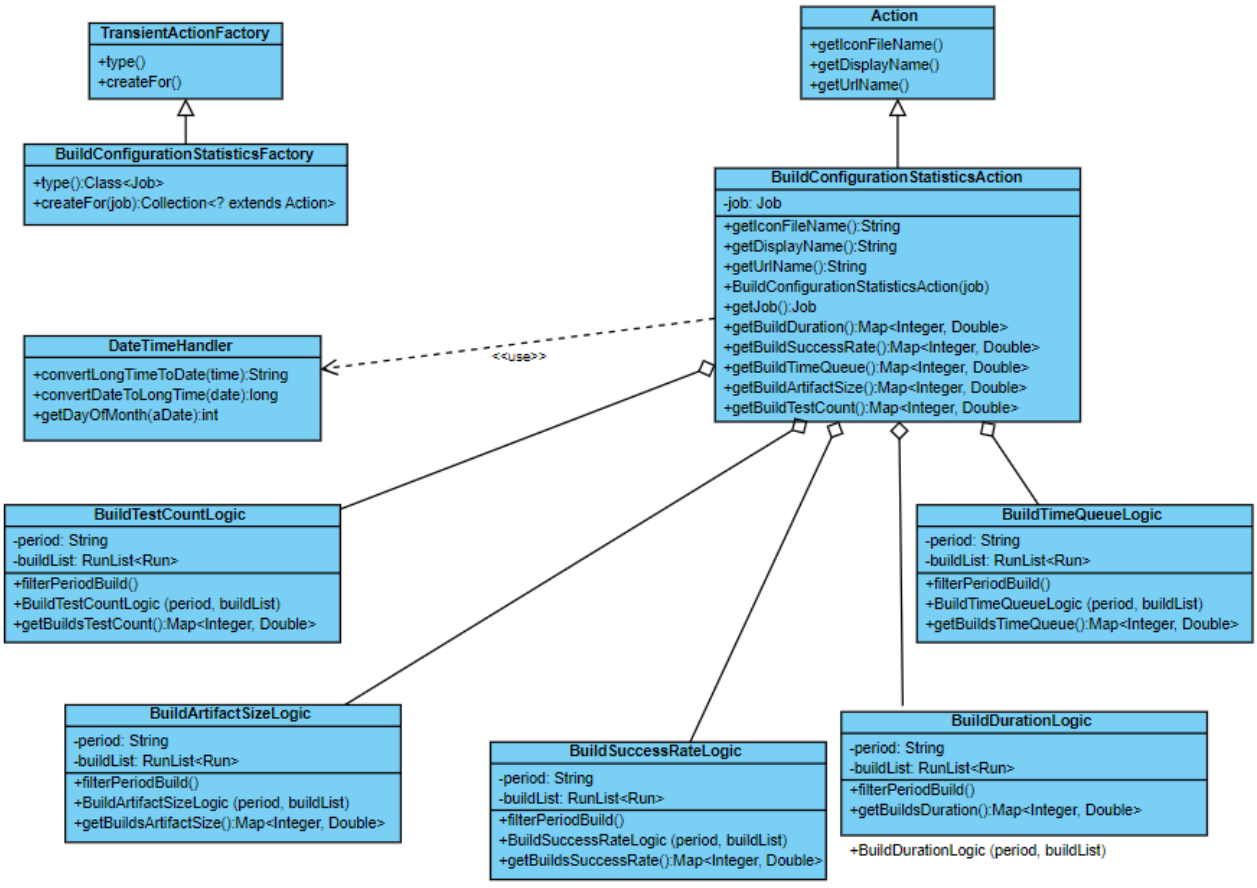

Рис.П1.1. Диаграмма классов плагина

Приложение 2

Idef0

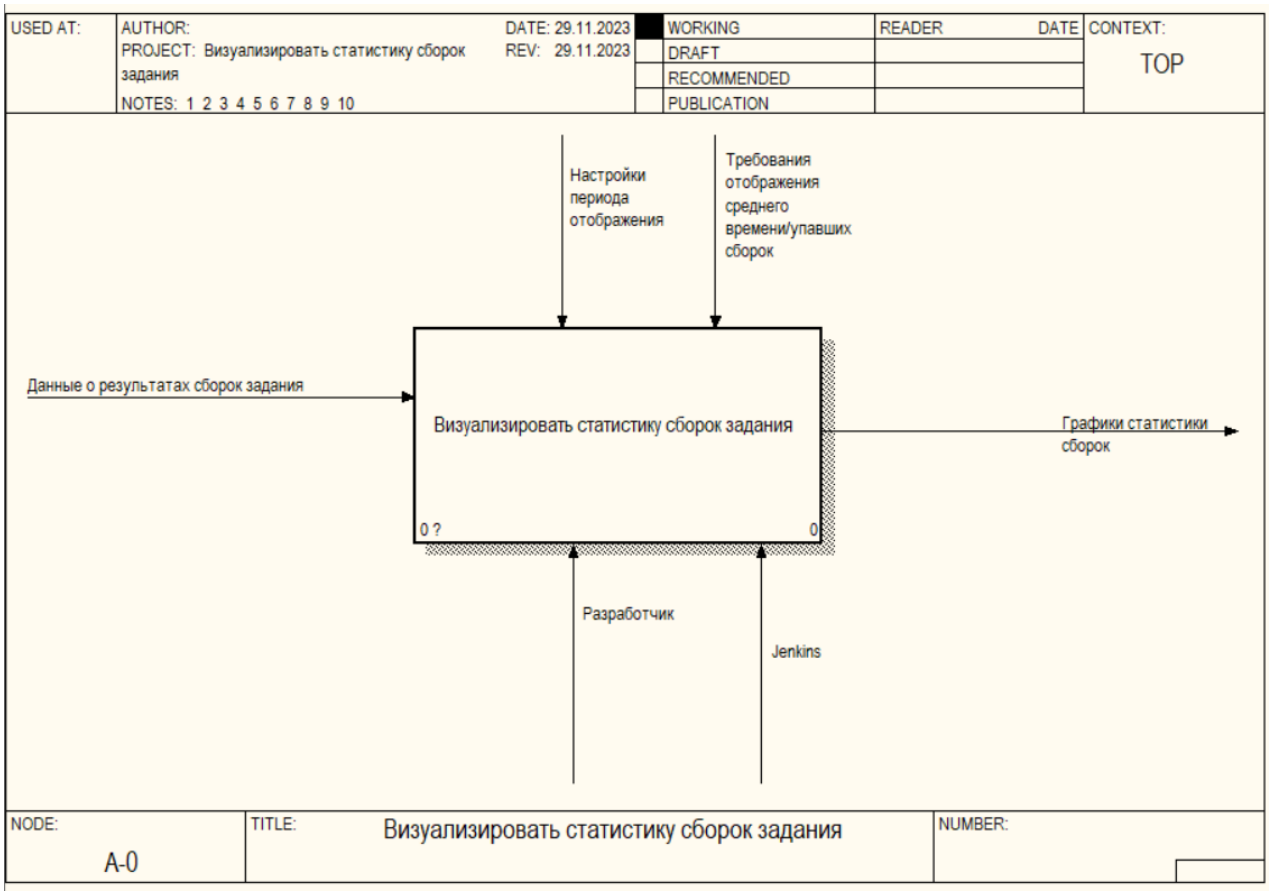

Рис.П2.1. Процесс визуализации сборок

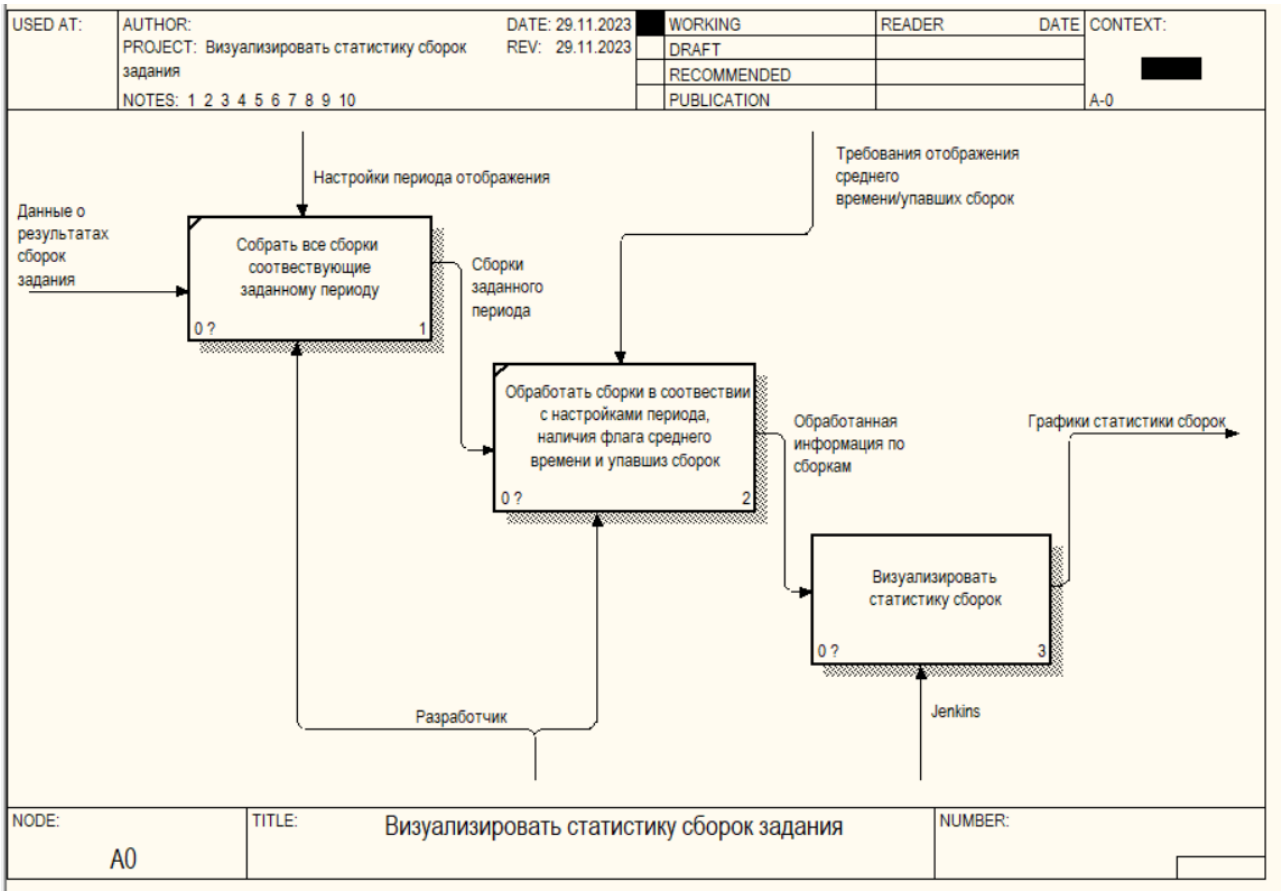

Рис.П2.2. Детализация процесса визуализации

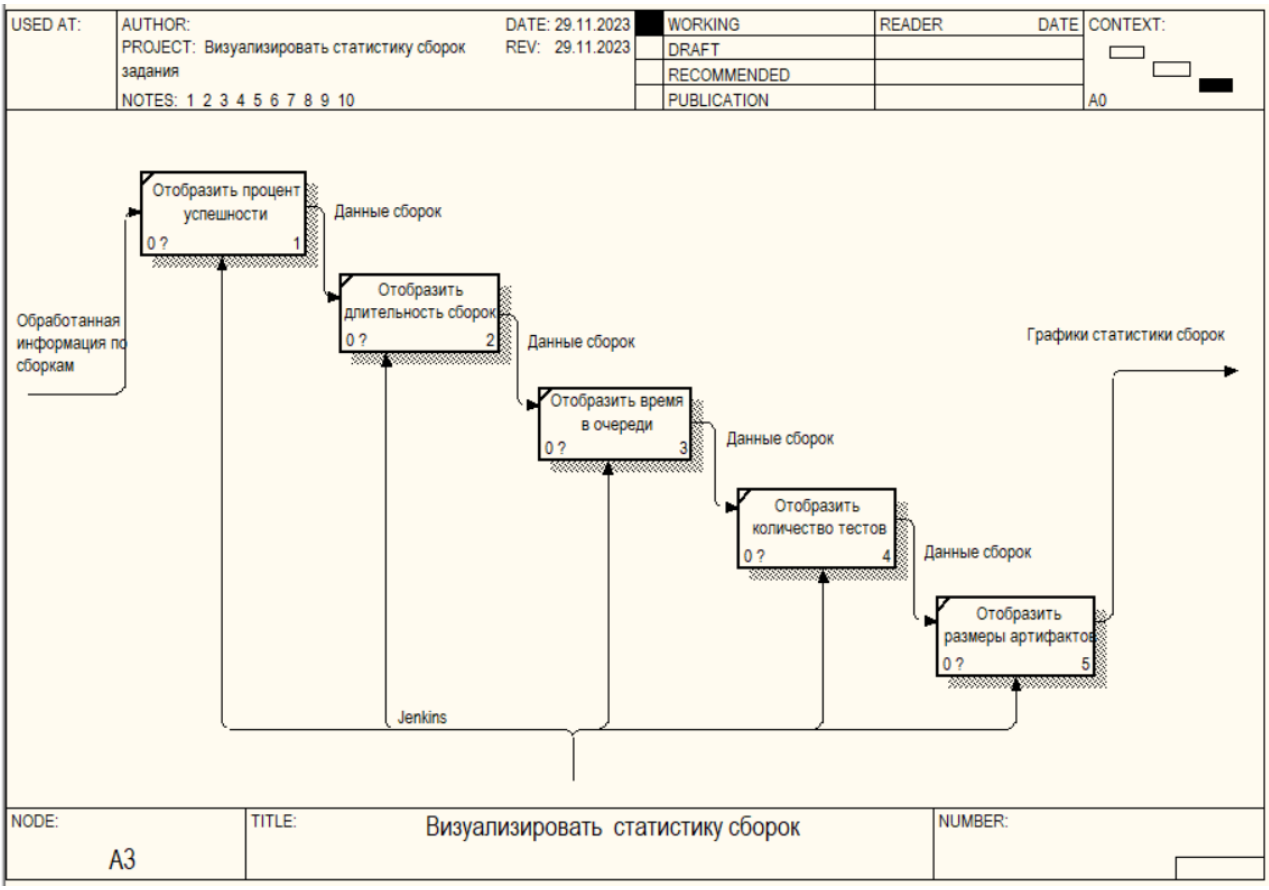

Рис.П2.3. Процесс построения графиков

Приложение 3

Use-case

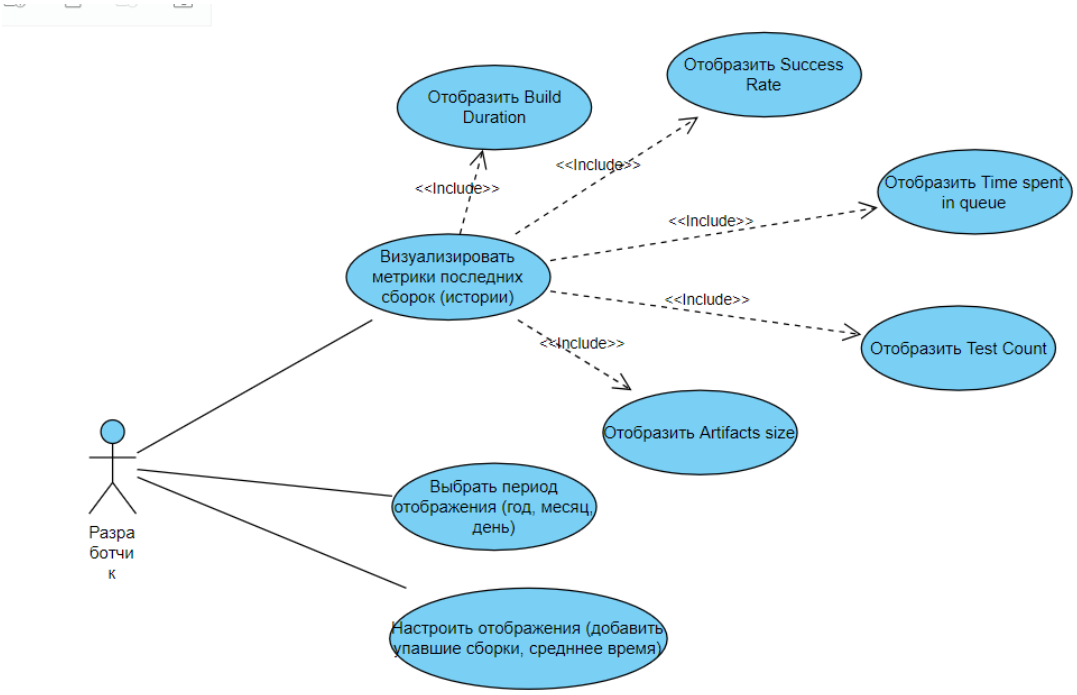

Рис.П3.1. Use case

Приложение 4

Программный код

```
package io.jenkins.plugins.sample;

import hudson.model.Action;
import hudson.model.Job;

import java.text.ParseException;
import java.util.Map;

public class BuildConfigurationStatisticsAction implements Action {

    private Job job;

    public BuildConfigurationStatisticsAction(Job job) {
        this.job = job;
    }

    @Override
    public String getIconFileName() {
        return "document.png";
    }

    @Override
    public String getDisplayName() {
        return "Build Configuration Statistics";
    }

    @Override
    public String getUrlName() {
        return "buildConfigurationStatistics";
    }

    public Job getJob() {
        return job;
    }

    public Map<Integer, Double> getBuildDuration() throws ParseException {
        return new BuildDurationLogic("month", job.getBuilds()).getBuildsDuration();
```

```
    }

}
```

```
package io.jenkins.plugins.sample;


import hudson.Extension;
import hudson.model.Action;
import hudson.model.Job;
import jenkins.model.TransientActionFactory;

import javax.annotation.Nonnull;
import java.util.Collection;
import java.util.Collections;

@Extension
public class BuildConfigurationStatisticsFactory extends
   TransientActionFactory<Job> {
    @Override
    public Class<Job> type() {
        return Job.class;
    }

    @Nonnull
    @Override
    public Collection<? extends Action> createFor(@Nonnull Job
        job) {
        return Collections.singletonList(new
           BuildConfigurationStatisticsAction(job));
    }
}
```

```
package io.jenkins.plugins.sample;

import hudson.model.Run;
import hudson.util.RunList;

import java.text.ParseException;
import java.util.HashMap;
import java.util.Map;

public class BuildDurationLogic {

    String period;
    RunList<Run> buildList;
```

```
    public BuildDurationLogic(String period, RunList<Run>
       buildList) {
        this.period = period;
        this.buildList = buildList;
    }

    public void filterPeriodBuild(){
        if (period == "month") {
            this.buildList = buildList
                    .filter(run -> {
                        try {
                            return run.getStartTimeInMillis()
                               >= DateTimeHandler.
                               convertDateToLongTime("2023 11
                               01 00:00:00");
                        } catch (ParseException e) {
                            throw new RuntimeException(e);
                        }
                    });
        }
    }

    public Map<Integer, Double> getBuildsDuration() throws
       ParseException {
        filterPeriodBuild();
        Map<Integer, Double> dayDuration = new HashMap<Integer
           , Double>();
        for (Run run : this.buildList) {
            int day = DateTimeHandler.getDayOfMonth(
               DateTimeHandler.convertLongTimeToDate(run.
               getStartTimeInMillis()));
            if (dayDuration.containsKey(day)){
                dayDuration.put(day,dayDuration.get(day) + run
                   .getDuration()/1000.0);
            } else {
                dayDuration.put(day,run.getDuration()/1000.0);
            }
        }
        return dayDuration;
    }
}
```

```
package io.jenkins.plugins.sample;

import java.text.Format;
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class DateTimeHandler {
    public static String convertLongTimeToDate(long time){
        Date date = new Date(time);
        Format format = new SimpleDateFormat("yyyy MM dd HH:mm
            :ss");
        return format.format(date);
    }
    /*
    * date format = yyyy MM dd HH:mm:ss
    *
    * */
    public static long convertDateToLongTime(String date)
       throws ParseException {
        SimpleDateFormat formatter = new SimpleDateFormat("
            yyyy MM dd HH:mm:ss");
        return formatter.parse(date).getTime();
    }


    public static int getDayOfMonth(String aDate) throws
       ParseException {
        SimpleDateFormat formatter = new SimpleDateFormat("
            yyyy MM dd HH:mm:ss");
        Calendar cal = Calendar.getInstance();
        cal.setTime(formatter.parse(aDate));
        return cal.get(Calendar.DAY_OF_MONTH);
    }
}
```

```
const labels = Array.from({length: 30}, (_, i) => i + 1);
const data = {
  labels: labels,
  datasets: [{
    label: 'Success rate',
    data: Array.from({length: 30}, () => Math.floor(Math.
       random() * 100)),
    backgroundColor: [
      'rgba(0, 255, 0, 0.5)',
    ],
    borderColor: [
```

```
      'rgb(0, 69, 36)',
    ],
    categoryPercentage: 1,
    borderWidth: 1,
    barPercentage: 1,
  }]
};

var buildDuration = document.querySelectorAll(".buildDuration
   ");
var dataBuildDurationValues = Array.from({length: 30}, () =>
   0);
for (var i=0; i<buildDuration.length; i++){
    console.log(buildDuration[i]);
    dataBuildDurationValues.splice(parseInt(buildDuration[i].
       querySelector('.key').textContent, 10) - 1, 1,
       parseFloat(buildDuration[i].querySelector('.value').
       textContent));
    console.log(dataBuildDurationValues[i]);
}



const dataBuildDuration = {
  labels: labels,
  datasets: [{
    label: 'Build duration',
    data: dataBuildDurationValues,
    borderColor: [
      'rgba(0, 180, 33, 1)',
    ],
    tension: 0.1

  }]
};

const dataTimeSpentQueue = {
  labels: labels,
  datasets: [{
    label: 'Time Spent In Queue',
    data: Array.from({length: 30}, () => Math.floor(Math.
       random() * 60)),
    borderColor: [
      'rgba(0, 180, 33, 1)',
    ],
```

```
    tension: 0.1

  }]
};
const dataTestCount = {
  labels: labels,
  datasets: [{
    label: 'Test Count',
    data: Array.from({length: 30}, () => Math.floor(Math.
       random() * 60)),
    borderColor: [
      'rgba(0, 180, 33, 1)',
    ],
    tension: 0.1

  }]
};
const dataArtifactsSize = {
  labels: labels,
  datasets: [{
    label: 'Artifacts Size',
    data: Array.from({length: 30}, () => Math.floor(Math.
       random() * 60)),
    borderColor: [
      'rgba(0, 180, 33, 1)',
    ],
    tension: 0.1

  }]
};









var allPerf = {
                        type: 'bar',
                        data: data,
                        options: {
                            scales: {
                                  y: {
                                    beginAtZero: true
```

```
                                    }
                                }
                            }
                        };


var settingsBuildDuration = {
                            type: 'line',
                            data: dataBuildDuration,
                        };
var settingsTimeSpentQueue = {
                            type: 'line',
                            data: dataTimeSpentQueue,
                        };
var settingsTestCount = {
                            type: 'line',
                            data: dataTestCount,
                        };
var settingsArtifactsSize = {
                            type: 'line',
                            data: dataArtifactsSize,
                        };
var ctx = document.getElementById("successRateChart").
    getContext("2d");
var ctxBuild = document.getElementById("buildDurationChart").
    getContext("2d");
var ctxTimeSpentQueue = document.getElementById("
    timeSpentQueue").getContext("2d");
var ctxTestCount = document.getElementById("testCount").
    getContext("2d");
var ctxArtifactsSize = document.getElementById("artifactsSize
    ").getContext("2d");
perfChartJsCharts["successRateChart"] = new Chart(ctx, allPerf
    );
perfChartJsCharts["buildDurationChart"] = new Chart(ctxBuild,
    settingsBuildDuration);
perfChartJsCharts["timeSpentQueue"] = new Chart(
    ctxTimeSpentQueue, settingsTimeSpentQueue);
perfChartJsCharts["testCount"] = new Chart(ctxTestCount,
    settingsTestCount);
perfChartJsCharts["artifactsSize"] = new Chart(
    ctxArtifactsSize, settingsArtifactsSize);
```

```
<?jelly escape-by-default='true'?>
```

```
<j:jelly xmlns:j="jelly:core" xmlns:l="/lib/layout" xmlns:st="
   jelly:stapler">
        <head>
                <style>
                        .buildDuration{
                                display:none;
                        }
                        .graph-container {
                        width: 90%;

                        }
                        .graph-block{
                                padding: 5px;
                                border: 1px solid grey;
                                margin: 10px;
                                display: flex;

                        }
                        .canvas-container{
                        width: 80%;

                        }
                        .settings{
                        padding: 5px;

                        margin: 10px;
                        display: flex;
                        flex-direction: column;

                        }
                        form label{
                        margin: 5px;
                        }
                </style>
        </head>

        <st:adjunct includes="io.jenkins.plugins.sample.
           BuildConfigurationStatisticsAction.
           declareChartJsClickArray"/>
        <l:layout title="Build Configuration Statistics">
                <l:side-panel>
                        <st:include page="sidepanel.jelly" it
                           ="${it.job}" optional="true" />
                </l:side-panel>
                <l:main-panel>
```

```
<h1>Statistics for job ${it.job.name}</h1>
<j:forEach var="type" items="${it.getBuildDuration()}">
    <p class="buildDuration">
        <span class="key">${type.key}</span>
        <span class="value">${type.value}</span>
    </p>
</j:forEach>
<div class="graph-container">
<div class="graph-block">
    <div class="canvas-container">
    <canvas id="successRateChart" width="90" height="15"></canvas>
    </div>
    <form class="settings">
        <label>
            Range:
            <select>
                <option>Month</option>
                <option>Day</option>
                <option>Year</option>
            </select>
```

```
65                                  </label>

                            </form>
                        </div>
                        <div class="graph-block">
70                          <div class="canvas-container">
                            <canvas id="buildDurationChart
                               " width="90" height="15"></
                               canvas>
                            </div>
                            <form class="settings">
                                <label>
75                                  Range:
                                    <select>
                                        <
                                           option
                                           >
                                           Month
                                           </
                                           option
                                           >
                                        <
                                           option
                                           >
                                           Day
                                           </
                                           option
                                           >
                                        <
                                           option
                                           >
                                           Year
                                           </
                                           option
                                           >
80                                  </select>

                                </label>
                                <label>
                                Average:
85                              <input type="checkbox
                                   "/>
                                </label>
                                <label>
                                    Show failed:
```

```
                                        <input type="
                                           checkbox"/>
90                                  </label>

                            </form>
                    </div>
                    <div class="graph-block">
95                          <div class="canvas-container">
                            <canvas id="timeSpentQueue"
                               width="90" height="15"></
                               canvas>
                            </div>
                            <form class="settings">
                                    <label>
100                                         Range:
                                            <select>
                                                    <
                                                       option
                                                       >
                                                       Month
                                                       </
                                                       option
                                                       >
                                                    <
                                                       option
                                                       >
                                                       Day
                                                       </
                                                       option
                                                       >
                                                    <
                                                       option
                                                       >
                                                       Year
                                                       </
                                                       option
                                                       >
105                                         </select>

                                    </label>
                                    <label>
                                            Average:
110                                         <input type="
                                               checkbox"/>
                                    </label>
```

```
                </form>
            </div>
            <div class="graph-block">
                <div class="canvas-container">
                <canvas id="testCount" width
                   ="90" height="15"></canvas>
                </div>
                <form class="settings">
                    <label>
                        Range:
                        <select>
                            <
                               option
                               >
                               Month
                               </
                               option
                               >
                            <
                               option
                               >
                               Day
                               </
                               option
                               >
                            <
                               option
                               >
                               Year
                               </
                               option
                               >
                        </select>

                    </label>
                    <label>
                        Show failed:
                        <input type="
                           checkbox"/>
                    </label>

                </form>
            </div>
            <div class="graph-block">
```

```
                <div class="canvas-container">
                <canvas id="artifactsSize"
                   width="90" height="15"></
                   canvas>
                </div>
140             <form class="settings">
                        <label>
                                Range:
                                <select>
                                        <
                                           option
                                           >
                                           Month
                                           </
                                           option
                                           >
145                                        <
                                           option
                                           >
                                           Day
                                           </
                                           option
                                           >
                                        <
                                           option
                                           >
                                           Year
                                           </
                                           option
                                           >
                                </select>

                        </label>
150                     <label>
                                Average:
                                <input type="
                                   checkbox"/>
                        </label>
                        <label>
155                             Show failed:
                                <input type="
                                   checkbox"/>
                        </label>

                </form>
```

```
                        </div>
                        </div>

                </l:main-panel>
        </l:layout>
        <st:adjunct includes="io.jenkins.plugins.sample.
           BuildConfigurationStatisticsAction.chartLogicBox"/>
</j:jelly>
```