Выпускная квалификационная работа

Разработка инструмента анализа зависимостей между компонентами программного комплекса NetCracker для повышения эффективности его сборки

Студент:
Холодилов Дмитрий
Научный руководитель:
к.ф.-м.н. Евдокимов А.В.

Программный компонент

Программный компонент — это обособленная часть программного комплекса, реализующая заданный набор функций и имеющая четко определенный интерфейс взаимодействия (API).

- В течение жизни компонента выделяются его *версии*
- Для хранения компонентов организуется *репозиторий*

В случае, когда один компонент использует функциональность другого (ссылаясь на его интерфейс), между компонентами существует *зависимость*.

Набор компонентов и зависимостей между ними можно представить в виде ориентированного графа.

Прямые зависимости — зависимости от компонентов, длина пути в графе до которых равна 1.

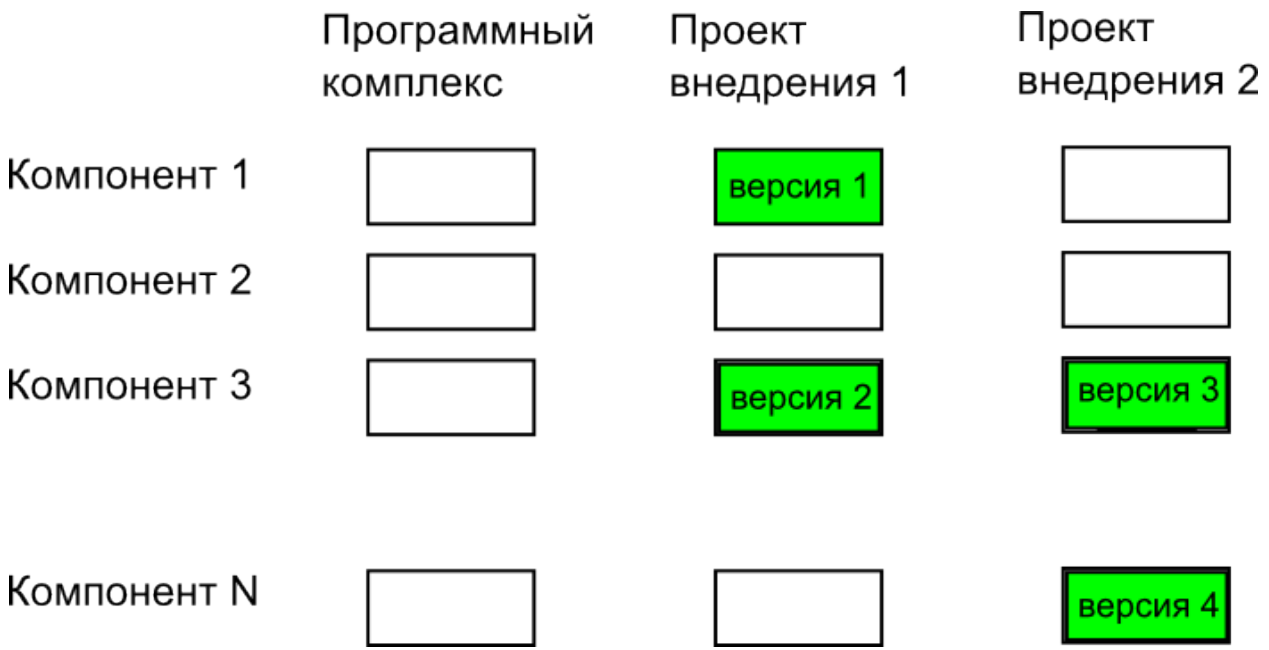
Транзитивные зависимости — зависимости от компонентов, длина пути в графе до которых больше 1.

Программный комплекс

В данной работе рассматривается программный комплекс NetCracker:

- состоящий из большого количества компонентов
- с архитектурой, построенной на взаимодействии с базой данных
- разрабатываемый для заказчиков в рамках *проектов внедрения*

Системная конфигурация — полный набор компонентов (с версиями), используемых проектом внедрения.



Выдача компонентов и сборка программного комплекса

Сборка — процесс формирования дистрибутива программного комплекса готового для передачи заказчику и развертывания (установки на сервер).

Сборка включает в себя:

- Получение из репозитория компонентов, указанных в конфигурации
- Компиляцию проектного кода

Проблема: из-за ошибок в конфигурации возникают ошибки сборки и ошибки во время работы комплекса.

Выдача компонента в проект включает в себя:

- Внесение компонента в конфигурацию
- Добавление данных компонента в сценарий развертывания БД

Требование: порядок выдачи нескольких компонентов должен учитывать зависимости между ними.

Цель работы

Уменьшение трудозатрат на выдачу компонентов и сборку комплекса в рамках проектов внедрения.

Задачи:

1) устранение дублирования информации о системных конфигурациях
2) автоматизация анализа зависимостей между компонентами
3) визуальное представления информации о зависимостях

Задача 1: создание централизованного хранилища системных конфигураций

Разработан формат хранилища:

xml файлы

```
<configuration>
  <project name='project1'
           version='1.0'/>
  <product_components>
    <component name='core'
               version='8.0'/>
    ...
  </product_components>
  <third_party_components>
    ...
  </third_party_components>
  ...
</configuration>
```

структура их хранения

```
sc_repository
   project1
      1.0
         project1-1.0.xml
      1.1
         project1-1.1.xml
   project2
      1.0
         project2-1.0.xml
      1.0.1
         project2-1.0.1.xml
      ...
```

Хранилище организовано как файловый сервер с сетевым доступом. Конфигурации помещаются в хранилище системой сборки.

Задачи 2-3: разработка программного инструмента для анализа зависимостей

Инструмент должен предоставлять следующую функциональность:

- Анализ согласованности системных конфигураций
- Анализ состава выдач при добавлении или обновлении компонентов и определение порядка выдачи
  - Нахождение циклических зависимостей между компонентами
- Сравнение системных конфигураций
- Визуализация графа зависимостей

Технические требования:

- Инструмент должен быть кросс-платформенным
- Инструмент не должен зависеть от конкретного устройства репозитория компонентов
- Инструмент должен иметь графический и текстовый (консольный) интерфейс

Анализ существующих инструментов работы с репозиториями компонентов

Apache Maven

Apache Ivy

= Свободная лицензия, кросс-платформенность

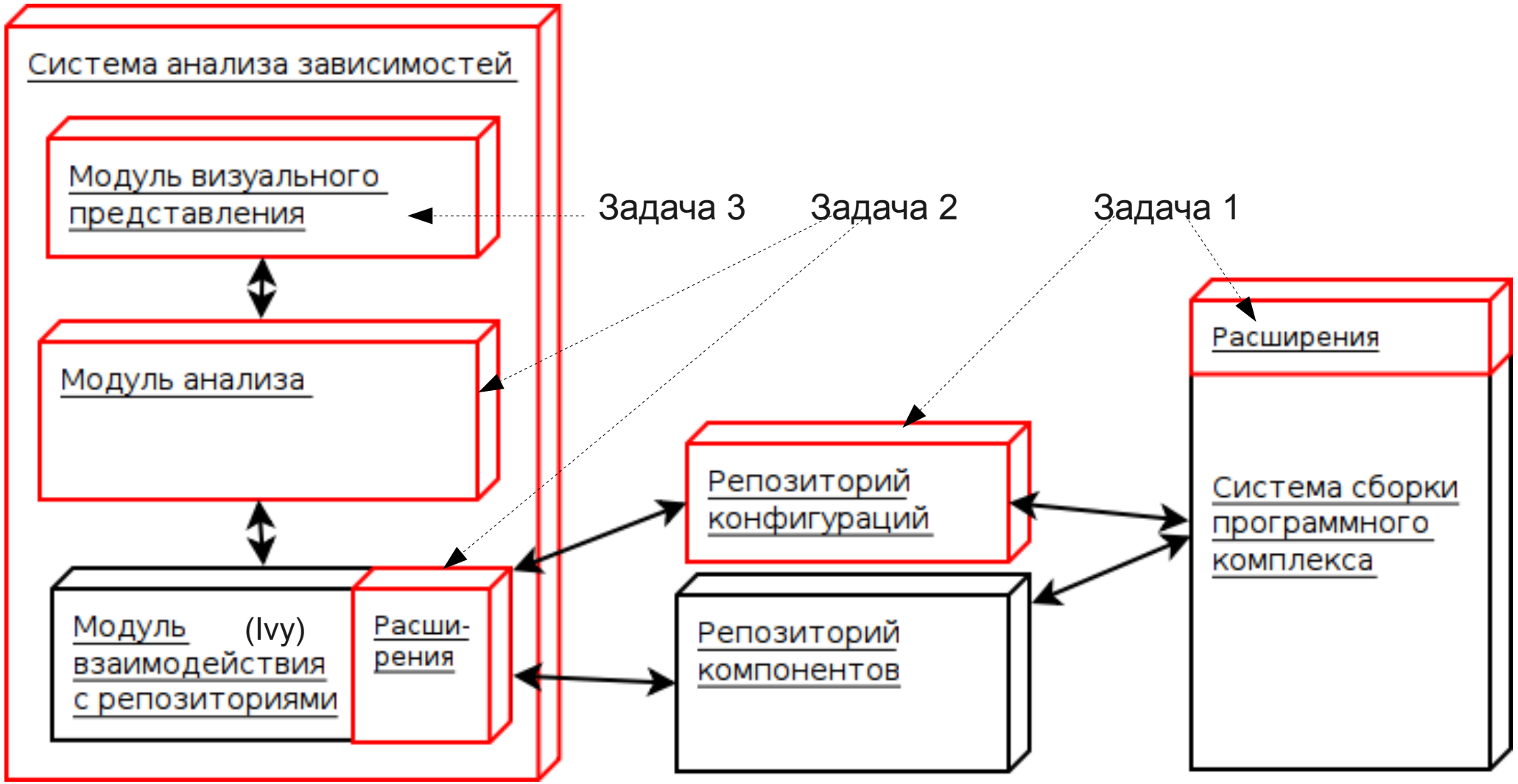
= Поддержка вычисления транзитивных зависимостей

= Поддержка кэширования информации

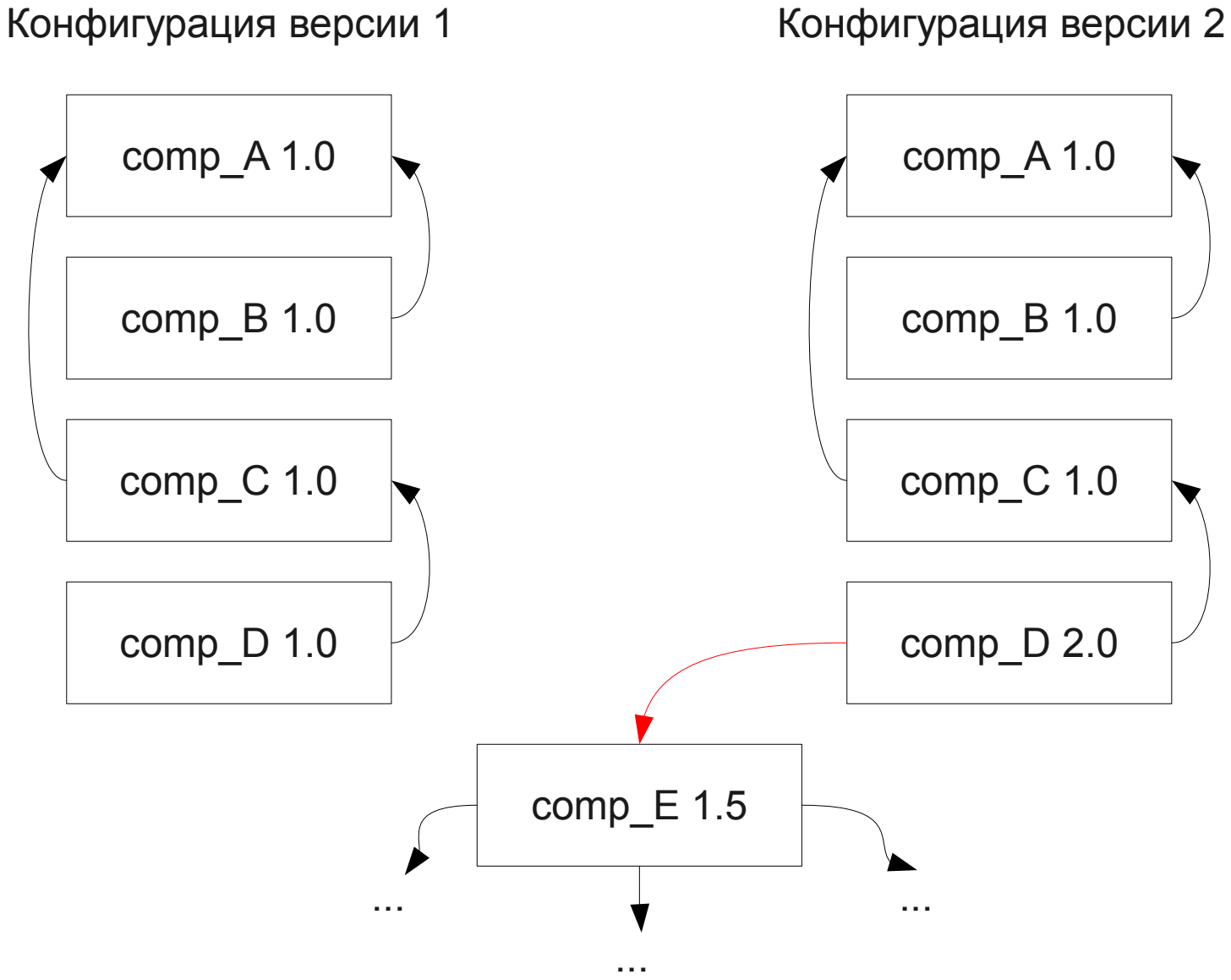
- Фиксированная структура репозитория
- Ограниченные возможности расширения функциональности

+ Возможность управления структурой репозитория
+ Гибкое расширение функциональности (плагины)

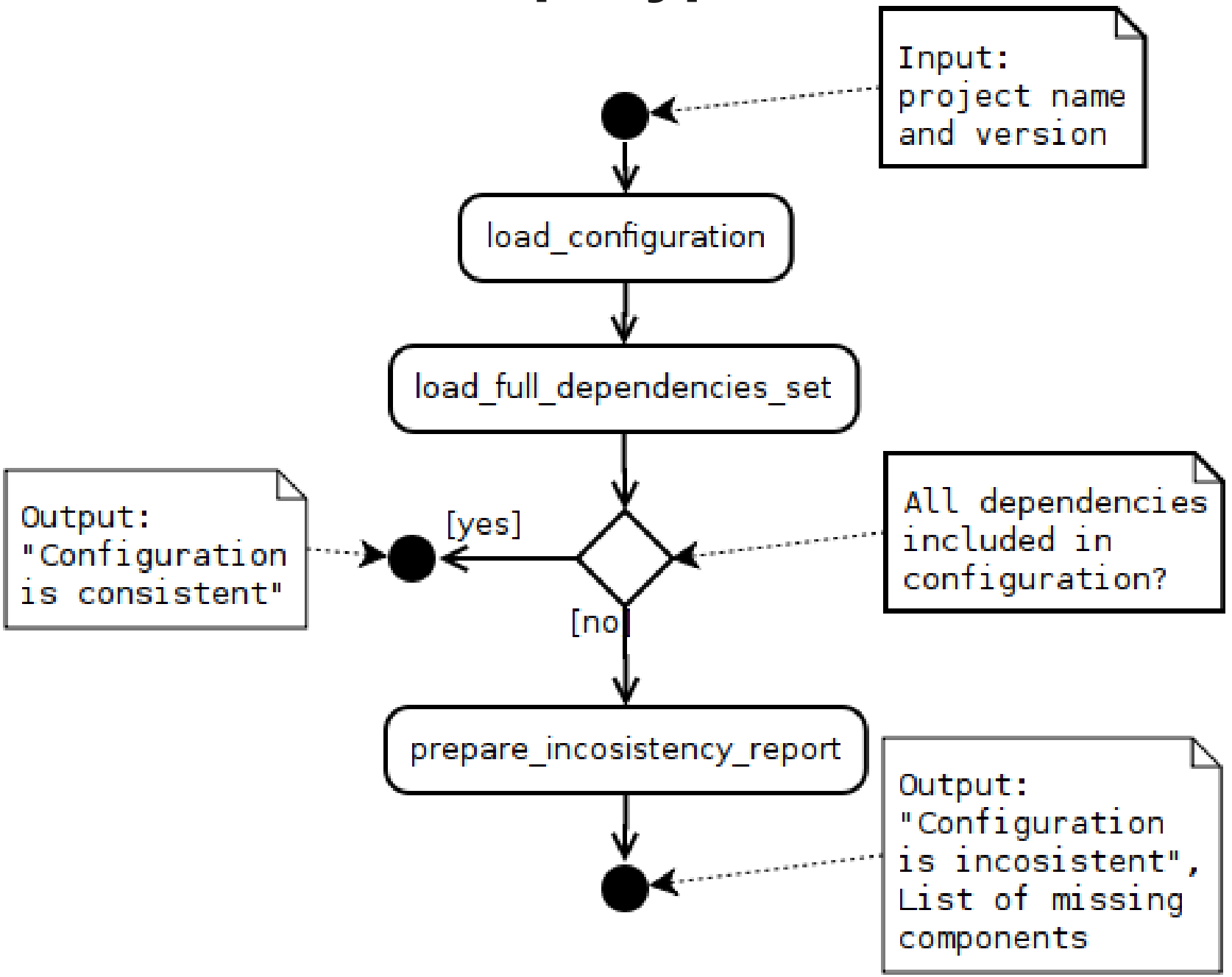
Разработанная система



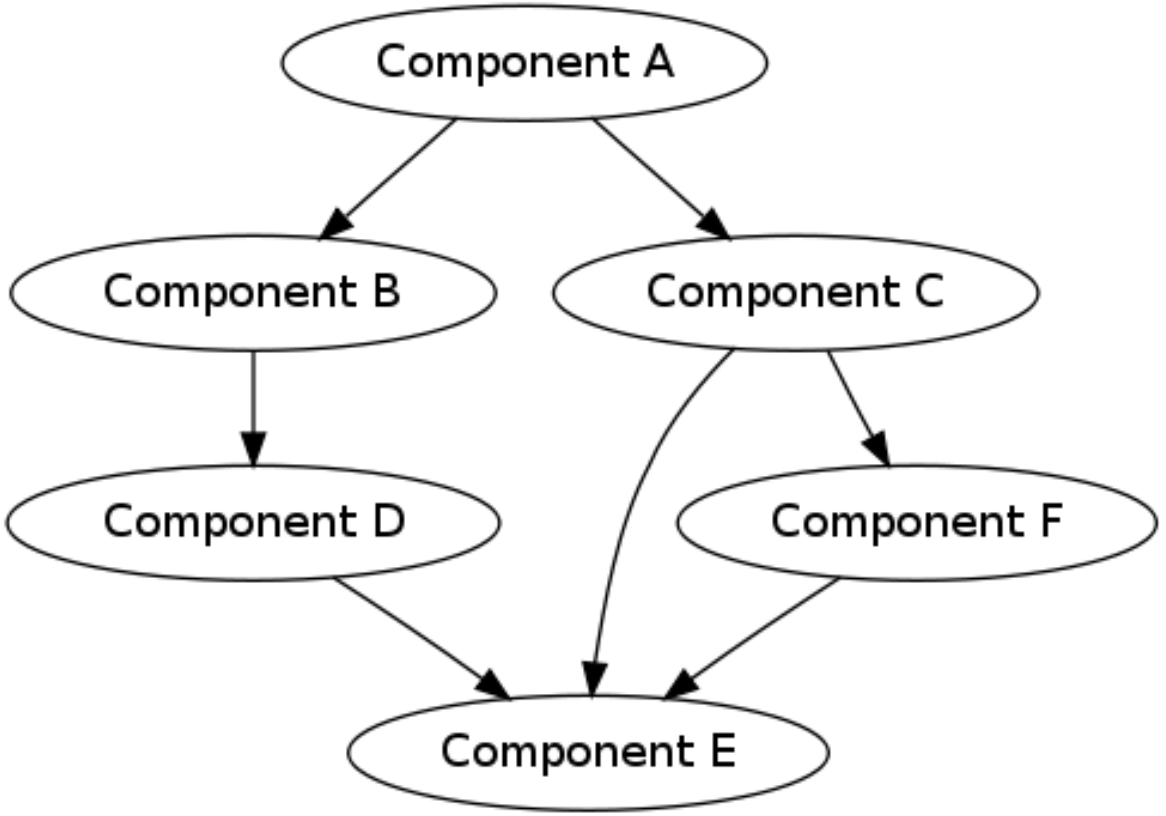
Согласованность системной конфигурации



Проверка согласованности конфигурации

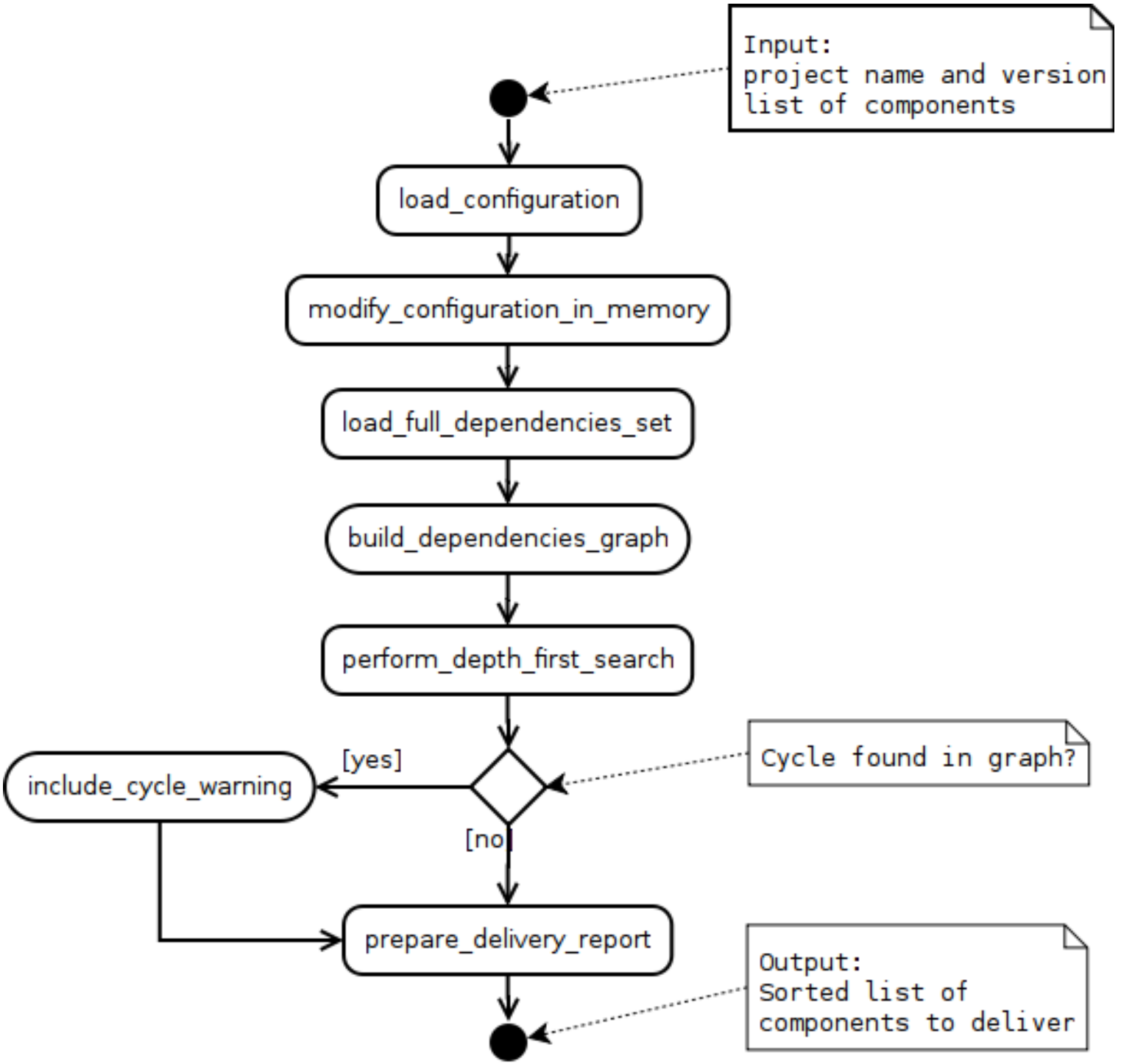


Анализ состава выдачи



Для определения порядка выдачи компонентов строится граф зависимостей между компонентами и производится топологическая сортировка его вершин.
В данном примере: E D B F C A
В случае обнаружения обратной дуги выдается предупреждение о наличии цикла в графе.

Алгоритм формирования списка выдачи



Интерфейс системы: пример отчета о сравнении конфигураций

Comparison of 'test_project 1.1' with 'test_project 1.0'

Swap

Back to configuration

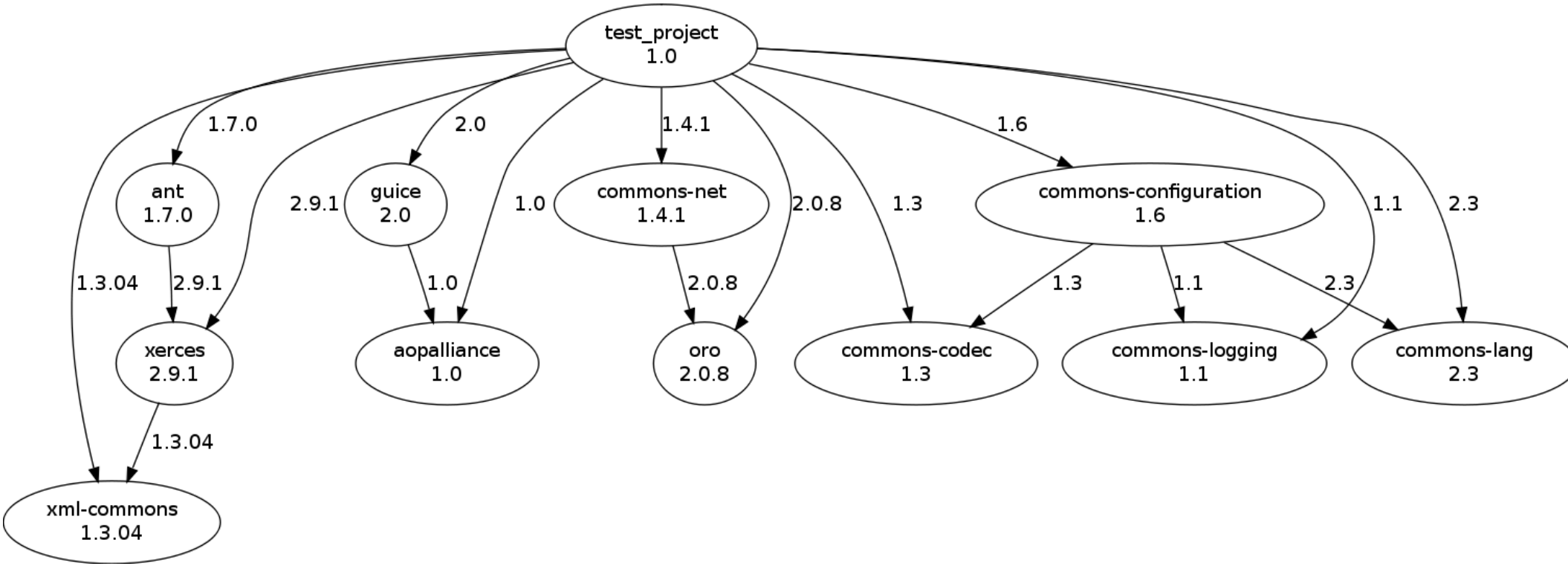
Product components:

test_project 1.0	test_project 1.1

Third-party components:

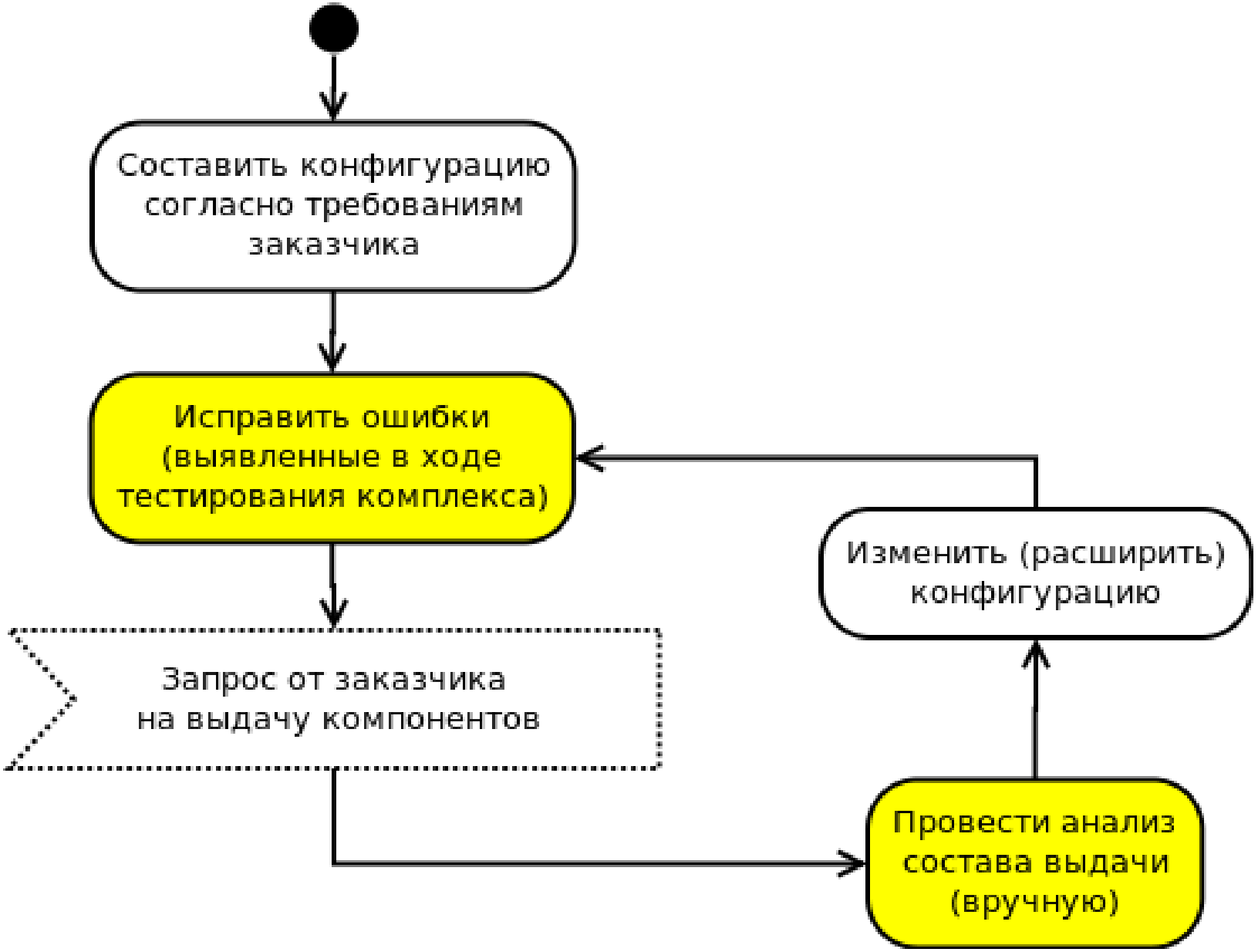
test_project 1.0	test_project 1.1
apache ant 1.7.0	
apache commons-configuration 1.6	apache commons-configuration 2.0
apache commons-net 1.4.1	apache commons-net 1.4.1
	apache commons-ssh 3.0
google guice 2.0	google guice 2.1

Визуализация графа зависимостей

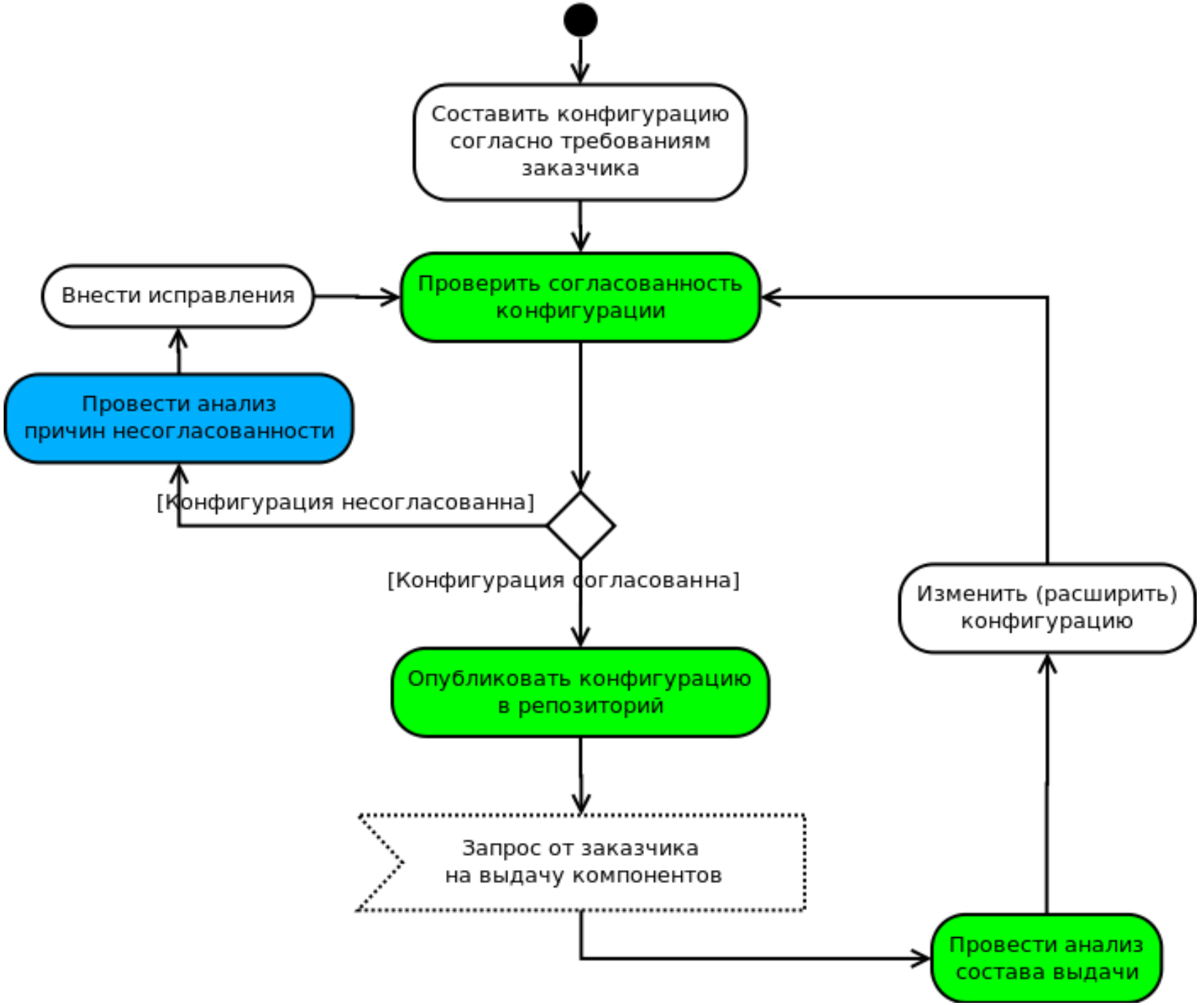


Для визуализации генерируется файл описания графа, который передается на вход утилите dot из пакета GraphViz

Процесс работы проекта до внедрения системы



Улучшенный процесс работы проекта



Заключение

Разработана система хранения и автоматического анализа системных конфигураций, позволяющая:

- Проверять согласованность конфигураций
- Автоматически составлять список компонентов для выдачи с учетом порядка
- Сравнивать конфигурации
- Визуализировать граф зависимостей

Внедрение системы:

- Существенно сократило расходы на подготовку выдачи компонентов для проектов внедрения
- Позволило находить ошибки в конфигурациях на раннем этапе (до передачи комплекса заказчику)

Спасибо за внимание!