БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра технологий программирования

Аннотация к дипломной работе

Разработка анализатора архитектуры Java-приложений

Тарасенко Дмитрий Алексеевич

Научный руководитель — ст. преподаватель кафедры технологий программирования Давидовская М. И.

Минск, 2022

РЕФЕРАТ

Дипломная работа, 82 с., 18 рис., 7 таблиц.

АРХИТЕКТУРА, АНАЛИЗАТОР, КОМПОНЕНТ, JAVA, МОДУЛЬ, МЕТРИКИ.

Объект исследования — объектом исследования является архитектура кода на языке программирования Java и принципы её оптимального построения. В качестве предмета исследования рассматривается разработка и исследование характеристик анализатора архитектуры приложения на языке программирования Java.

Цели работы — рассмотреть концепции и методы анализа архитектуры кода, реализовать решения для их применения, а также разработать приложение для анализа архитектуры Java-приложений и вывода статистических показателей оценки архитектурного анализа.

На этапе сопровождения программного продукта основной задачей разработчика является внесение изменений в готовый продукт с целью удовлетворения новых требований. Понимание этого привело к разработке критериев архитектуры программного обеспечения. Данные критерии заключаются как в правилах, которым должны следовать разработчики при написании (например, широко применяющиеся при объектно-ориентированном программировании принципы SOLID), так и в конкретных количественных метриках, позволяющих оценить уже разработанную архитектуру. На основе этих критериев производится анализ архитектуры с помощью программных средств, которые в дальнейшем будем называть анализаторами архитектуры приложений.

Анализатор, разработанный в ходе данной работы на данном этапе, включает в себя стандартный функционал для компонентов: подсчет базовых характеристик в виде устойчивости, абстрактности, числа входящих и исходящих зависимостей, цикломатической сложности; обнаружение циклических зависимостей среди пакетов и классов, а также способен определить простейшее разбиение анализируемого проекта на компоненты различными методами. Также анализатор включает в себя функционал, которым не обладают рассмотренные в сравнительном анализе решения: поиск возможного разбиения модели по интерфейсам, представление многоуровневой схемы приложения. Данные алгоритмы позволяют избежать бесполезных отклонений графа зависимостей приложения от древовидного, исключить ромбовидные зависимости, повысить абстрактность кода при необходимости и определить уровень дальнейшего расширения функционала приложения.

РЭФЕРАТ

Дыпломная работа, 82 с., 18 мал., 7 табліц

АРХІТЭКТУРА, АНАЛІЗАТАР, КАМПАНЕНТ, JAVA, МОДУЛЬ, МЕТРЫКА.

Аб'ект даследавання — аб’ектам даследавання з’яўляюцца архітэктура кода на мове праграмавання Java і прынцыпы яе аптымальнай пабудовы. У якасці прадмета даследавання разглядаецца распрацоўка і даследаванне характарыстык аналізатара архітэктуры прыкладання на мове праграмавання Java.

Мэты працы — разгледзець метады аналізу архітэктуры кода, распрацаваць метады іх сумеснага прымянення, а таксама распрацаваць прыкладанне для вырашэння задачы, апісанай на этапе праектавання.

На этапе суправаджэння праграмнага прадукта асноўнай задачай распрацоўніка з'яўляецца занясенне змен ва ўжо які функцыянуе прадукт з мэтай задавальнення новых патрабаванняў. Разуменне гэтага прывяло да распрацоўкі крытэрыяў архітэктуры праграмнага забеспячэння. Дадзеныя крытэрыі заключаюцца як у правілах, якіх павінны прытрымлівацца распрацоўшчыкі пры напісанні (напрыклад, шырока прымяняюцца пры аб'ектна-арыентаваным праграмаванні прынцыпы SOLID), так і ў канкрэтных колькасных метрыках, якія дазваляюць ацаніць ужо распрацаваную архітэктуру. На аснове гэтых крытэрыяў праводзіцца аналіз архітэктуры з дапамогай праграмных сродкаў, якія ў далейшым будзем называць аналізатарамі архітэктуры прыкладанняў.

Аналізатар, распрацаваны падчас дадзенай працы на дадзеным этапе, уключае ў сябе стандартны функцыянал для кампанентаў: падлік базавых характарыстык у выглядзе ўстойлівасці, абстрактнасці, колькасці ўваходзяць і выходных залежнасцяў, цыкламатычнай складанасці; выяўленне цыклічных залежнасцяў сярод пакетаў і класаў, а таксама здольны вызначыць найпростае разбіццё аналізуемага праекта на кампаненты рознымі метадамі. Таксама аналізатар уключае ў сябе функцыянал, якім не валодаюць разгледжаныя ў параўнальным аналізе рашэння: пошук магчымага разбіцця мадэлі па інтэрфейсах, прадстаўленне шматузроўневай схемы прыкладання. Дадзеныя алгарытмы дазваляюць пазбегнуць бескарысных адхіленняў графа залежнасцяў прыкладання ад дрэвападобнага, выключыць ромбападобныя залежнасці, павысіць абстрактнасць кода пры неабходнасці і вызначыць узровень далейшага пашырэння функцыяналу прыкладання.

ABSTRACT

Graduate work, 82 p., 18 illustrations, 7 tables.

ARCHITECTURE, ANALYZER, COMPONENT, JAVA, MODULE, METRICS.

Object of research — the object of research is the architecture of the code in the Java programming language and the principles of its optimal construction. The subject of the study is the development and study of the characteristics of the analyzer of the application architecture in the Java programming language.

Purpose — to consider methods of code architecture analysis, to develop methods of their joint application, as well as to develop an application to solve the problem described at the design stage.

At the stage of maintaining a software product, the main task of the developer is to make changes to an already functioning product to meet new requirements. This understanding led to the development of software architecture criteria. These criteria consist both in the rules that developers must follow when writing (for example, the SOLID principles that are widely used in object-oriented programming), and in specific quantitative metrics that allow evaluating an already developed architecture. Based on these criteria, the architecture is analyzed using software tools, which will be referred to as application architecture analyzers.

The analyzer developed in the course of this work at this stage includes the standard functionality for components: calculation of basic characteristics in the form of stability, abstractness, the number of incoming and outgoing dependencies, cyclomatic complexity; detection of cyclic dependencies among packages and classes and is also able to determine the simplest division of the analyzed project into components by various methods. The analyzer also includes functionality that the solutions considered in the comparative analysis do not have: searching for a possible partition of the model into interfaces, representing a multi-level application scheme. These algorithms allow you to avoid useless deviations of the application's dependency graph from a tree-like one, eliminate diamond-shaped dependencies, increase the abstractness of the code if necessary, and determine the level of further expansion of the application's functionality.