АННОТАЦИЯ К РАБОЧЕЙ ПРОГРАММЕ ДИСЦИПЛИНЫ
«Функциональные языки разработки распределенных систем»

по основной профессиональной образовательной программе по направлению подготовки 09.04.04 «Программная инженерия» (уровень магистратуры)

Направленность (профиль): Разработка программно-информационных систем
Общий объем дисциплины – 4 з.е. (144 часов)
Форма промежуточной аттестации – Экзамен.
В результате освоения дисциплины у обучающихся должны быть сформированы компетенции с соответствующими индикаторами их достижения:

- ПК-1.1: Осуществляет выбор методов программной реализации распределенных информационных систем;
- ПК-1.2: Создает программное обеспечение распределенных информационных систем;
- ПК-5.1: Анализирует существующие подходы к верификации моделей программного обеспечения;
- ПК-5.2: Применяет в профессиональной деятельности существующие подходы к верификации моделей программного обеспечения;
- ПК-10.1: Анализирует и выбирает методы тестирования создаваемого программного обеспечения;
- ПК-10.2: Осуществляет тестирование создаваемого программного обеспечения;

Содержание дисциплины:
Дисциплина «Функциональные языки разработки распределенных систем» включает в себя следующие разделы:
Форма обучения очная. Семестр 2.
1. Введение. Обзор функционального программирования и функциональных языков. Методы программной реализации распределенных информационных систем. Программное обеспечение распределенных информационных систем. Существующие подходы к верификации моделей программного обеспечения. Организация промышленного тестирования создаваемого программного обеспечения. Выбор методов тестирования создаваемого программного обеспечения.
Вычисление в терминах функций. Императивное и функциональное программирование. Функциональные особенности нефункциональных языков. Применение функционального программирования к решению высокопроизводительных задач. Обзор особенностей современных функциональных языков OCaml, F#, Scala и др..
2. Основы языка Erlang. Общие сведения об Erlang. Установка Erlang. IDE для Erlang. Работа в интерпретаторе. Основы языка Erlang. Переменные. Грамматика. Списки. Функции. Атомы. Кортежи. Функторы. Рекурсия. Функциональный стиль. Модульное тестирование и документирование. Пример выполнения задания: умножение матриц..
3. Многопроцессорное и распределённое взаимодействие на Erlang. Акторная парадигма. Общая характеристика процессов в Erlang. Создание процесса. Отправка и получение сообщений. Связи процессов и слежение за состоянием процесса. Словари процессов. Принципы параллельного программирования на Erlang. Анализ работы параллельных процессов в Erlang. Распределённое программирование. Моделирование работы протоколов на Erlang.
4. Взаимодействие Erlang с Java. JInterface - возможности. Подключение JInterface в проект на Java. Узел. Процесс.Типы данных. Пример работы (пинг-понг). Удалённый вызов функций из модулей Erlang. Возможное применение интерфейса Java-Erlang..
5. Верификация акторных распределённых систем. Программа для проверки. Язык Promela и LTL-требования. Установка необходимых средств разработки. Реализация демо-модели. Верификация демо-модели. Верификация модели протокола PoET..
6. OTP фреймворк. Проблемы логической организации Erlang кода. Проблема написания отказоустойчивых приложений. Проблема горячей замены кода приложения без его остановки. Проблема развёртывания Erlang приложений. OTP фреймоворк и шаблоны поведения. Application, supervisor, gen_server и пример построения сервера. OTP gen_event и логирование. Конечные

автоматы в Erlang и OTP gen_fsm. Организация горячей замены кода стандартными средствами и в OTP. Установка приложений OTP.

7. Базовые сведения о Haskell. Общие сведения о Haskell. Установка Haskell. IDE для Haskell. Hello world на Haskell + базовые вещи о монадах. Функции и сопоставление в Haskell. Модульное тестирование. Списки и pattern matching в Haskell. Кортежи и pattern matching в Haskell. Собственные типы данных и операции, а также тестирование с quickCheck Ленивость в Haskell..

8. Параллелизм и межпроцессорное взаимодействие в Haskell. Сборка программы с включением параллельных
инструкций. Создание параллельно выполняющихся действий. О транзакционной памяти.
Взаимодействие через каналы Control.Concurrent.Chan. Межпроцессорное взаимодействие в Haskell с помощью Cloud Haskell. Установка Cloud Haskell. Ping-pong на Cloud Haskell. Пример реализации распределённой взаимодействующей системы на Cloud Haskell. Генерация серверов на Erlang и Haskell и клиентов к ним по стандартизированному описанию..

Разработал:
доцент
кафедры ПМ С.М. Старолетов

Проверил:
Декан ФИТ А.С. Авдеев