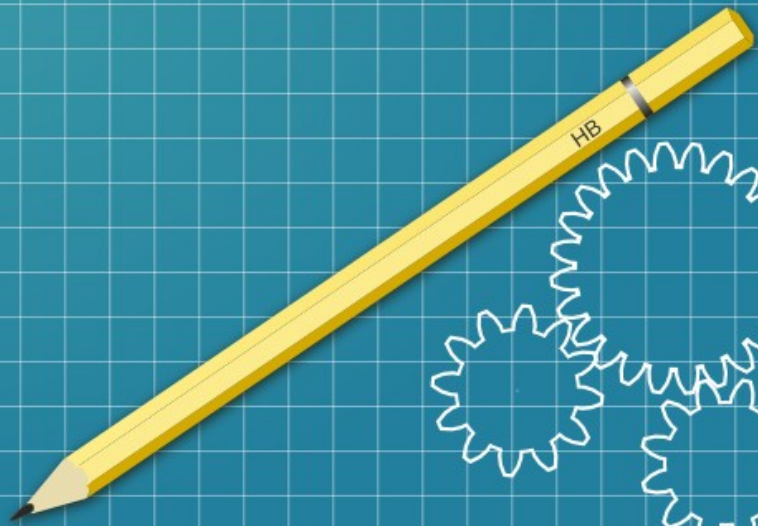
Model-finder
Alloy analyzer

Software design



Введение

- Правильное, безошибочное, формально верифицированное, но плохое ПО.
  Разве такое бывает?
- Примеры корректного, полностью отвечающего спекам, плохого ПО.
- Плохие спецификации. Плохие формальные спецификации.
  Примеры плохих спек.

Введение

- У нас много хороших инструментов, методик и пр. для проверки, верификации и тд. ПО на соответствие спекам.
- Но весьма мало (в сравнении) методов, инструментов и пр. для разработки качественных спецификаций.
- Почему такая ассиметрия? Виновник — Дейкстра :).
- Что такое качественная спецификация и архитектура ПО?
- Почему важно качество базовых спецификаций и базовой архитектуры? Проблема контроля сложности ПО.
- Концептуальный дизайн. Что такое концепт?

Встречаем: Майкл Джексон

- Нет, это не тот Джексон, о котором вы подумали :)
- Это ученый, который много лет занимался проблемами разработки ПО с точки зрения методологии формулировки задач на разработку, на основе анализа проблемной области, для которой разрабатывается ПО.
- Основные труды: «Problem Frames: Analysing and Structuring Software Development Problems», «Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices»

и Дэниел Джексон, его сын

- Профессор CS в MIT
- Продолжил развивать идеи своего отца
- Автор языка Alloy и инструментов для анализа моделей, описанных на этом языке
- Основная книга про Alloy: «Software Abstractions»
- Ещё одна весьма интересная книга «Design by Concept: A New Way to Think about Software»

Основной вклад отца и сына в CS

- Сформулировали задачу разработки хороших спецификаций
- Разработали подходы и методологию анализа задачи предметной области и формулировки этой задачи в области разработки автоматизированных систем (аппаратура, ПО и др) для решения этой задачи.
- Дэниел разработал формальный декларативный язык Alloy и инструмент (совместно со многими другими разработчиками) Alloy analyzer
- Всё это даёт нам хороший инструментарий разработки и анализа хороших (и не очень :)) спецификаций, мы можем зарание прототипировать важные элементы архитектуры систем, найти новые интересные свойства, лучше понять и сформулировать стоящие перед нами задачи разработки и пр.

It’s Party Time

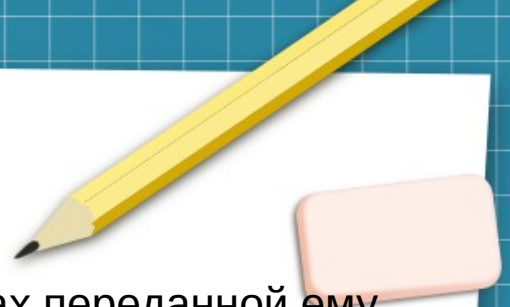
- Шпаргалки по языку Alloy:
  https://esb-dev.github.io/mat/alloy-cheatsheet.pdf
  https://www.monperrus.net/martin/alloy-quick-ref.pdf
  https://alloytools.org/download/alloy-language-reference.pdf
- Ставим Alloy Analyzer отсюда:
  https://github.com/AlloyTools/org.alloytools.alloy/releases

  для Linux/Windows нужно ещё установить Java:
  https://java.com/ru/download/windows_manual.jsp?locale=ru
  запуск: java -jar org.alloytools.alloy.dist.jar

It’s Party Time

- Основные концепции Alloy (немного рисунков на доске):
  - Сигнатуры
  - Отношения
  - Предикаты
  - Функции
  - Выражения
  - Макросы
  - Модули (и их параметры, объяснить про exactness параметров-сигнатур)

It’s Party Time



- Попробуем определить модуль, который вводит порядок на элементах переданной ему в параметрах сигнатуры
- Что такое порядок? (как раз пример небольшого концепта)
- Свойства порядка? Свойства корректности?
- Пишем немного кода
- Run them all! (смотрим, что получается)
- Разбираем возможности отображения моделей, настройки отображения и пр. Умение читать модели.
- Check them all! (пробуем найти контрпримеры, которые бы нарушали свойства корректности)
- Обсуждаем примеры и контрпримеры.

Alloy dynamics

- Alloy только для статических структур? Нет!
- Как ввести динамические свойства?
- Способы моделирования операций над структурами.
- Концепция времени, моменты времени. Вспоминаем про модуль порядка, вот тут он нам и пригодится. (пример использования концепта и композиции концептов)
- Пару слов про private части модулей

It’s Party Time Again!

- Изобретаем колесо — идею двусвязного списка
- Думаем, что это такое (да не колесо, а двусвязный список)?
- Свойства? Какие отношения между элементами?
- Корректность?
- Операции над элементами. Время. Пример использования концепта — порядок над элементами времени.
- Смотрим, что получается.
- Уточняем свойства, смотрим ещё раз.
- Пытаемся хакнуть свойства корректности и найти контрпримеры.

Встречаем: Electrum2!

- Берём тут: https://github.com/haslab/Electrum2/releases/tag/v2.1.5 запускать так же, как и оригинал.
- Electrum2 = Alloy + time + temporal logic. Тут подробнее про расширение языка: https://github.com/haslab/Electrum/wiki/Alloy-Language-Reference
- Var — это то, что меняется. Разница между var у сигнатур и var у отношений.
- Always, eventually, before, after, once, historically, until, releases, since, triggered — это же вроде из другой оперы, PLTL? Но теперь и в Alloy!
- Штрих теперь не просто штрих
- Немного рисунков на доске, чтобы запутать ещё больше :)

Electrum2

- Особенности отображения моделей (по сравнению с Alloy)
- Особенности настройки отображения моделей
- Переход между статичными моделями (новые сигнатуры и отношения)
- Переход между трассами в текущей модели
- Проход по временнОй трассе

Опять 25

- Теперь реализуем концепт двусвязного списка на Electrum2
- Переформулируем некоторые свойства на временнУю логику
- Посмотрим, что получилось
- Уточним, поправим, отремонтируем
- Посмотрим ещё раз
- Сравним к классическим Alloy (удобство использования var, удобство использования операторов always, eventually, after, etc)

Electrum2

- Можно проверять временнЫе свойства на бесконечности, для этого нужно поставить NuSMV/NuXMV + electrod
- Подробности тут: http://haslab.github.io/Electrum/
- Подробнее останавливаться не будем, иначе точно выйдем за временнЫе рамки воркшопа

Alloy Analyzer ограничен :(, да здравствует Higher Order Alloy !

- Основные ограничения Alloy Analyzer: первый порядок, невозможность квантификации по отношениям и множествам.
- Варианты обхода этих ограничений в классическом Alloy Analyzer: введение дополнительных сигнатур (доп. косвенность), переформулирование свойств в первый порядок, ослабление некоторых свойств, разбиеие сложных свойств на простые из которых оно следует (доказательство следствия можно вынести в другие инструменты)
- А если не хочется всем этим заморачиваться, то берём Higher-Order Alloy (Alloy*). Берём тут: https://aleksandarmilicevic.github.io/hola/
- Предупреждение: Alloy* на описаниях с высшими порядками тормозит очень сильно, так что часто может быть выгоднее день потерять потом за пять минут долететь (в смысле повозиться с переформулированием описания на первый порядок и потом получить высокую скорость поиска примеров/контр-примеров)

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.
It makes use of the works of Mateus Machado Luna.