Текстовый язык автоматного программирования

В. С. Гуров,
М. А. Мазин,
А. А. Шалыто

Инструментальное средство UniMod

Проектирование

1. Создать концептуальную модель предметной области в виде UML диаграммы классов используя классические методы
2. Выделить поставщики событий, автоматы и объекты управления. Оставшиеся классы – контейнеры для данных.
3. Создать диаграмму связей автоматов
   - Поставщики событий поместить справа
   - Объекты управления слева
   - Автоматы посередине
   - Создать связи между поставщиками событий, автоматами и объектами управления
4. В поставщиках событий определить множество производимых событий (`ex`)
5. В объектах управления создать два множества методов соответствующих:
   - Входным воздействиям (`xj`)
   - Выходным воздействиям (`zk`)
6. Для каждого автомата создать диаграмму состояний

Реализация

7. Реализовать поставщики событий и объекты управления вручную или выбрать готовые из репозитория
8. Запустить модель прямо из среды разработки, либо преобразовать модель в XML-описание для последующей интерпретации, либо преобразовать модель в код на целевом языке программирования для последующей компиляции

Диаграмма классов

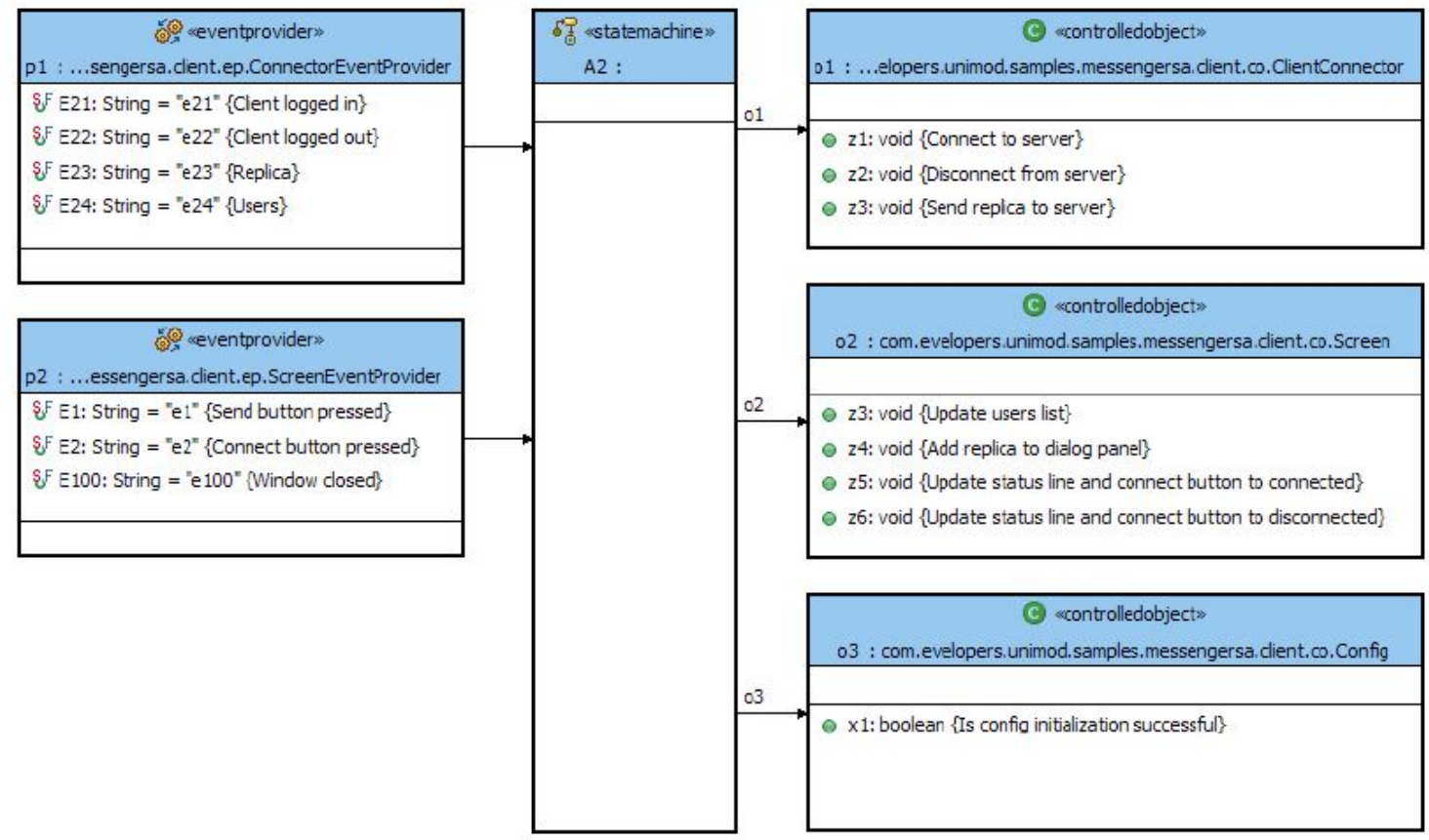
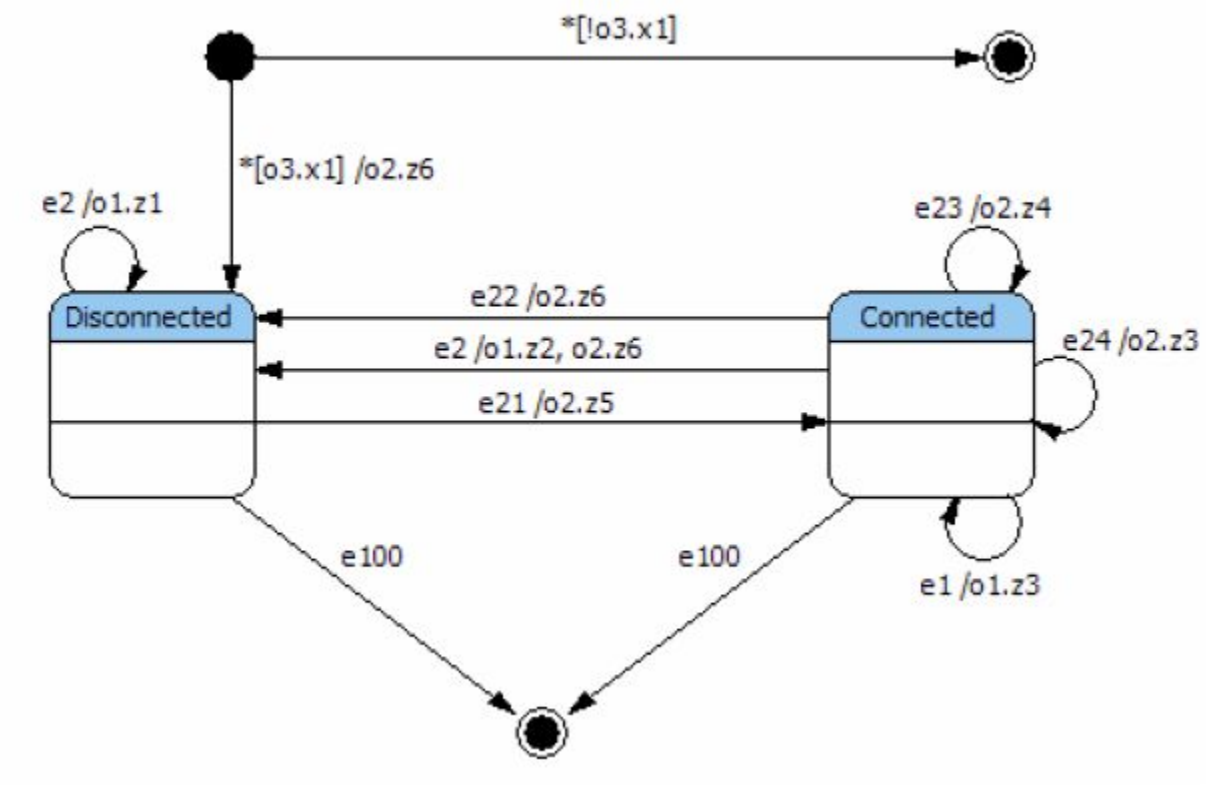


Диаграмма состояний



Графический язык программирования

- Достоинства средства UniMod
  - Диаграммы более выразительны, чем текст
  - UML-нотация — общепринята
- Недостатки средства UniMod
  - Вводить диаграммы неудобно
  - Программисты предпочитают текстовый код

Текстовый язык автоматного программирования

- Проблемно-ориентированный автоматный язык
- Автомат описывается в терминах автоматного программирования
- Диаграммы генерируются по мере ввода текста программы

Разработка текстового языка программирования

- Транслятор
  - Лексический анализатор
  - Синтаксический анализатор
  - Семантический анализатор
  - Генератор кода
- Интегрированная среда разработки
  - «Подсветка» ошибок
  - Автоматическое завершение ввода
  - Навигация по коду

Система метапрограммирования MPS

- Принимает на вход
  - абстрактный синтаксис
  - конкретный синтаксис
  - систему типов
  - кодогенератор
- Автоматически строит интегрированную среду разработки

Виды проблемно-ориентированных языков

- Проблемно-ориентированные расширения существующих языков
  - Например,
    - язык доступа к базе данных,
    - язык описания регулярных выражений и т.д.

- Независимые языки
  - Например,
    - язык планировщика задач операционной системы

Автоматное расширение языка Java

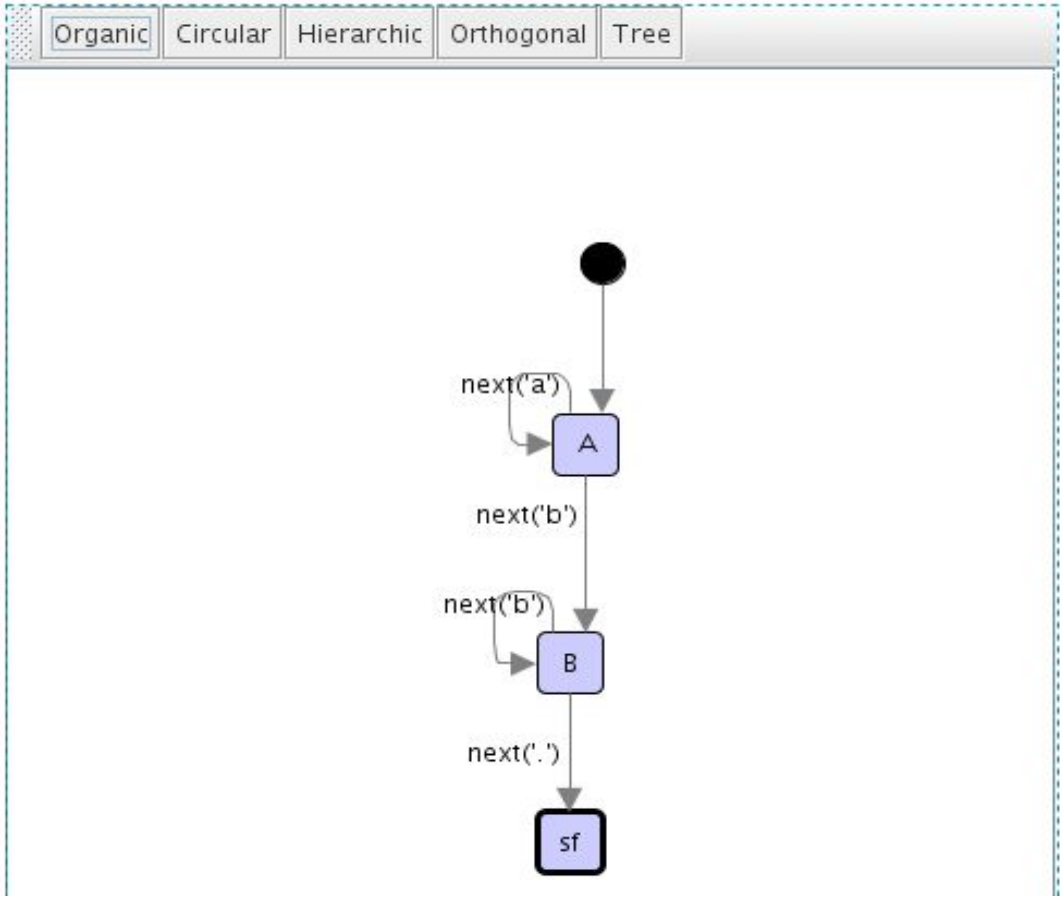
- Позволяет описывать поведение Java-класс в виде автомата
  - Отправка событий — вызов специальных методов класса
  - Состояние храниться в переменной объекта
  - Автомат описывается в терминах автоманого программирования

Независимый автоматный язык

- Все приложение описывается в терминах автоманого программирования
- Возможно использовать для генерации не Java-кода
- Соответствует понятию запускаемых спецификаций

Генерация диаграмм

```
statemachine AcceptString {
   << associations >>
   void next ( string );
   initial state s0 {
     transiteto A
   }
   state A {
      << onenter >>
      << onexit >>
      on next ( "a" )  transiteto A  ;
      on next ( "b" )  transiteto B  ;
      << inner states >>
   }
   state B {
      << onenter >>
      << onexit >>
      on next ( "b" )  transiteto B  ;
      on next ( "." )  transiteto sf  ;
      << inner states >>
   }
   final state sf {}
}
```



Результаты

- Разработанное средство позволяет
  - разрабатывать автоматные программы
  - описывать поведение Java-классов в виде автоматов
  - использовать преимущества текстового ввода программ и представления автоматов в виде диаграмм

Спасибо