ТЕСТВАНЕ С МАШИНА НА КРАЙНИТЕ СЪСТОЯНИЯ

доц. д-р Десислава Петрова-Антонова

Съдържание

- Машини на крайните състояния: базови концепции
- Тестване с машини на крайните състояния
- Case study: тестване на уеб приложения с машина на крайните състояния

Машина на крайните състояния: необходимост

Тестов модел с поддръжка на множество състояния

- Особености на моделите с дърво за взимане на решение, йерархичните контролни списъци и многомерните контролни списъци
  - Наличие на краен брой фази или списъци
  - Всяка фаза или списък са уникални
  - Крайното решение (резултат) се определя еднозначно от елементите в списъците или фазите в дървовидната структура, през които се преминава
- Особености при обработката на информация в софтуераните продукти
  - Поведението на софтуера е свързано с повторение на състояния и изпълнението на задачи в цикъл
- Моделиране на повтарящо се поведение
  - Замяна на точките за взимане на решение в списъците или фази на работа със състояния
  - Замяна на избора на елемент от даден списък или взимане на решение със преход между състояния
  - Осигуряване на връщане назад към предходно състояние

Машини на крайните състояния: базови концепции

- Елементи на машините на крайните състояния
  - Статични елементи
    - ✓ Състояния и преходи
  - Динамични елементи
    - ✓ Входове и изходи
- Поведение на машините на крайните състояния
  - Във всеки момент от време машината е в определено състояние или в преход между две състояния
    - ✓ При игнориране на фактора време, машината винаги е в точно определено състояние, наречено текущо състояние
  - Детерминирана машина на крайните състояния
    - ✓ Изходът и следващото състояние се определят еднозначно от входа и текущото състояние
- Графично представяне на машините на крайните състояния с модел на Мили
  - Състоянията се представят с възли на граф
  - Преходите се представят с насочени дъги между два възела в граф
  - Входовете и изходите се представят с теглови коефициенти на дъгите в графа

Машини на крайните състояния: базови концепции

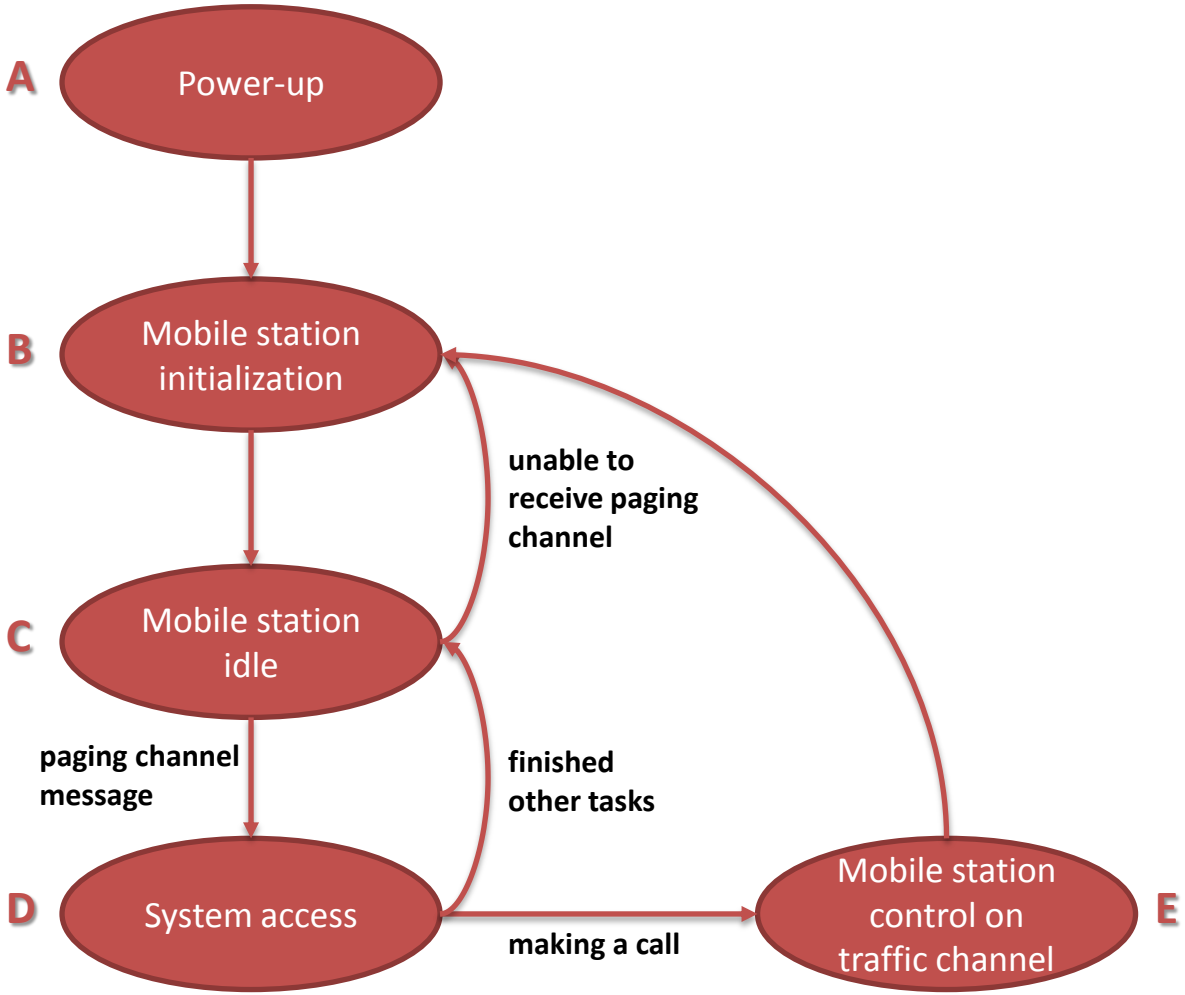
Машини на крайните състояния: примери

- Изпълнение на програма като машина на крайните състояния
  - Инициализиращо състояние: стартиране на програмата (1)
  - Преход към следващо състояние при използване на потребителска функция или изпълнение на израз или процедура (2)
  - Многократно извършване на преходи с възможност за повторение на състояния (3)
  - Крайно състояние: приключване на изпълнението на програмата (4)
  - При преходите между състоянията може да е необходим вход или да се генерира изход (5)
- Използване на уеб като машина на крайните състояния
  - (1) Зареждане на страница по подразбиране или потребителски дефинирана страница
  - (2) Щракване върху линк или директно зареждане от адресната лента на страница
  - (3) Многократно преминаване между страници
  - (4) Затваряне на браузъра или спиране на заявките за нови страници
  - (5) Преходите могат да се асоциират със заявки за страница, съобщения за грешка и др.

Конструране на машина на крайните състояния

Машини на крайните състояния: примери

- Обработка на разговор при клетъчните комуникационни системи



Начини за представяне

	A	B	C	D	E
A	na	-/-	na	na	na
B	na	na	-/-	na	na
C	na	noC/-	na	msg/-	na
D	na	na	done/-	na	call/-
E	na	-/-	na	na	na

- Формално представяне на графа
  - Множество на състоянията: {A, B, C, D, E}
  - Множество на преходите: пример {C, B, “unable to receive paging channel”, –}
- Таблично представяне
  - Състоянията се разписват по редове и колони
  - Редовете представят началните състояния, а колоните – крайните състояния
  - Клетките представят входа и изхода за преход от едно състояние в друго
  - Клетките, които са празни или маркирани с “na” показват невъзможност за преход между съответните състояние
- Списъчно представяне, базирано на формалното представяне на граф
  - Множеството на състоянията се представя със списък
  - Възможните преходи се представят със списък от вида {C, B, “unable to receive paging channel”, –}

Проблеми при моделирането

- Проблеми със състоянията
  - Некоректно състояние, определено от некорекно поведение на системата
  - Липсващо състояние
  - Излишно състояние, определено от недостижимо състояние, до което няма път от инициализиращото състояние
- Проблеми с преходите
  - Липсващ преход
  - Излишен преход, определен от множество преходи до едно и също състояние при един и същ вход
  - Некоректен преход, определен от преход към неочаквано състояние или състояние, произвеждащо неочакван изход
- Проблеми с входовете
  - Проблемите с входовете се интерпретират като проблеми със състоянията и преходите
  - Разширение с цел прихващане на невалидни входове
    - ✓ Игнориране на невалиден вход посредством запазване на текущото състояние
    - ✓ Диектно прихващане на невалиден вход посредство произвеждане на изход със съобщение за грешка или преход към състояние за прихващане на грешки
- Проблеми с изходите
  - Проблемите с изходите се интерпретират като проблеми с преходите

Конструиране на модел

- Стъпка 1: Определяне на източник на информация и събиране на данни
  - Външна продуктова спецификация или очаквани потребителски сценарии
  - Вътрешна продуктова информация, налична в документите за проектиране на продукта и програмния код
- Стъпка 2: Конструиране на първоначален модел
  - Идентифициране и изброяване на състоянията
    - ✓ Използване на вложени или йерархични машини на състоянията в случай на прекалено голям брой състояния
  - Идентифициране на преходите с помощта на входни стойности
    - ✓ Създаване на класове на еквивалентност за преходите при много голям или безкраен брой входни стойности
  - Идентифициране на входно-изходните зависимости
- Стъпка 3: Подобряване на модела и валидация
  - Идентифициране на липсващи състояния и преходи посредством създаване на контролни списъци, базирани на източниците на информация
  - Идентифициране на излишните преходи и състояния посредством **съпоставяне с информацията в източниците** за създаване на модела или извършване **на анализ за достижимост**
  - Идентифициране на некорекни състояния и преходи

Тестване с на машина на крайните състояния

- Покритие на състоянията
  - Тестовият сценарий е множество от входни стойности, които осигуряват преход от определено начално състояние до определено крайно състояние
  - Един тестов сценарий може да осигури пълно покритие на състоянията при наличие на точно едно начално и едно изходно състояния
  - При наличие на множество начални и крайни състояния броят на тестовите сценарии нараства
- Покритие на преходите
  - Входовете се класифицират в класове на еквивалентност, от които се генерират тестови сценарии
- Генериране на тестови данни
  - Използват се входните стойности, необходими за преход между състоянията
- Изпълнение на тестовите сценарии
  - Възможност за възстановяване на “текущо състояние” при стартиране на тестовете
- Проверка на резултата
  - Използват се изходите, получени след преход към следващо състояние

Приложения и ограничения

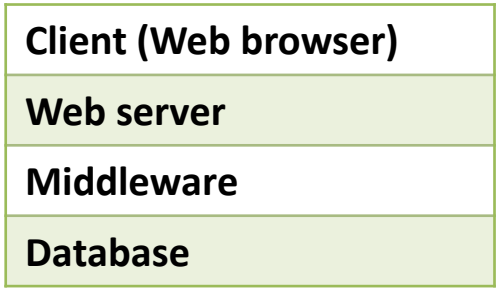
- Приложения на машините на крайните състояния
  - Софтуер с потребителски интерфейс
  - Системи с ясно идентифицируеми състояния и преходи като системи, работещи в реално време, и управляващи системи
  - Обектно-ориентирани системи
- Ограничения
  - Невъзможност за моделиране на голям брой състояния
  - Строга йерархия при йерархичните машини на крайните състояния
  - Трудно постигане на покритие при големи системи
    - ✓ Голям брой йерархични машини на крайните състояния
    - ✓ Неравномерно разпределение на проблемите и честотата на използване

Тестване на уеб базирани приложения

CASE STUDY

Особености на уеб базираните приложения

- Фокусиране главно върху информацията и документите
  - Традиционният софтуер извършва главно изчислителни операции
- Смесване на навигацията с изчислителната обработка
  - При традиционния софтуер навигацията и изчислителната обработка за отделени
- Наличие на множество стартови и изходни точки
  - Традиционният софтуер притежава една стартова точка и ограничен брой изходни точки
- Голям брой навигационни страници
  - Традиционният софтуер притежава ограничен набор от менюта
- Необходимост от разнообразна поддръжка за софтуерната архитектура и прилежащата инфраструктура



Какво да тестваме?

- Дефиниция на уеб повреда
  - Неспособност за коректно доставяне на информация или документ
- Източници на повреди
  - Хостинг и мрежови повреди
    - ✓ Анализират се със съществуващите техники за системни и мрежови проблеми
  - Повреди в браузърите
    - ✓ Повредите се третират като повреди в софтуерен продукт и се третират със съществуващите техники за тестване
  - Повреди в кода или съдържанието
    - ✓ Тестване на удобство за употреба (usability)
    - ✓ Тестване на надеждност (reliability)
    - ✓ Уеб компоненти (HTML документ, Java, JavaScript и ActiveX, SGI-Bin скриптове, База от данни, Мултимедийни компоненти)

Типове тестване при уеб приложенията

- Функционално тестване
- Тестване за производителност
- Тестване на надеждност
- Тестване на удобство за употреба
- Тестване на натоварването и стрес тестване
- Тестване за съвместимост с браузъра

Тестване с машина на крайните състояния

- Конструиране на машина на крайните състояния
  - Представяне на всяка страница като състояние, при което всяка страница може да бъде начално или крайно състояние
  - Представяне на уеб навигацията с преходи
    - ✓ Обикновено преходите се асоциират с хипервръзки предвид непредсказуемостта при директно въвеждане на уеб адрес в браузъра и избор на съхранена страница (bookmarked favorites)
  - Входовете се представят като щракване върху хипервръзките
  - Изходите се представят като зареждане на страница със съответно съобщение, съдържащо HTML статус, грешка и т.н.
- Ограничения
  - Наличие на голям брой страници, което води до голям брой състояния
  - Разпределението на проблемите и използваемостта на различните софтуерни компоненти е неравномерно, което изисква извършване на статистическо тестване, базирано на употреба


Q&A