Modelagem por Diagramas de Estados

Prof. Hugo Vieira Neto
2020/1

Objetivo

- Usar diagramas de estado como ferramenta para modelagem dinâmica de sistemas embarcados:
  - Estados e transições
  - Ações e atividades
  - Eventos e condições

Sistemas Embarcados

- Artefatos reativos
  - Reagem a eventos externos
  - Podem gerar eventos internos
  - Reagem a esses eventos internos
- Reagir = gerar saídas, mudar de estado, alterar variáveis internas (parte do estado)

Sistemas Embarcados

- É necessário representar o comportamento dinâmico do sistema em função do tempo e de eventos específicos, indicando como este irá reagir a estes eventos (modelagem)
- Diagramas de estados (*statecharts*) servem muito bem ao propósito de modelagem do comportamento dinâmico de um sistema

Diagramas de Estados

- Estados possíveis pelos quais um determinado sistema pode passar, bem como transições entre eles (associadas a cada evento e sob quais condições)
- Usados por projetistas de hardware e de software para representar máquinas de estados finitos (MEF)

Máquinas de Estados Finitos

- Estado = situação estável de uma MEF
- Transição = indica a possibilidade de sair de um estado e entrar em outro
  - Evento (*trigger*): evento que ocasiona a transição
  - Condição (*guard*): condição necessária para efetuar a transição (além da ocorrência do evento)
  - Ação (*behavior*): ação realizada *durante* a transição
- Evento = acontecimento relevante em instante de tempo bem definido – pode ser interno ou externo

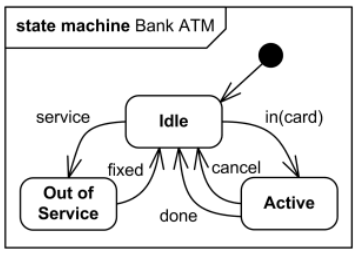
Origens: Statecharts

- Professor David Harel
  - Instituto Weizmann (Israel), 1984
- Contribuições:
  - História
  - Hierarquia
  - Concorrência
- Referência:
  - Harel, D., Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming 8(3), pp. 231-274, 1987.

Notação Gráfica Básica

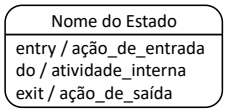
- Quadro com caixa para o nome da MEF
- Pseudo-estado inicial ●
- Pseudo-estado final ◉
- Estado ▭
- Transição ⟶

Notação Gráfica Básica



Notação Gráfica: Estados

- Estado (nome do estado)
  - Atividade interna: **do /**
  - Ações geradas por evento interno: **evento /**
  - Ações geradas por entrada ou saída: **entry /**, **exit /**



Ações x Atividades

- Ações ocorrem em transições (atômicas)
  - Ações do entry são executadas quando a transição cruza a fronteira do estado entrando
  - Ações do exit são executadas quando a transição cruza a fronteira do estado saindo
- Atividades ocorrem durante a permanência no estado (término → mudança de estado)

Notação Gráfica: Transições

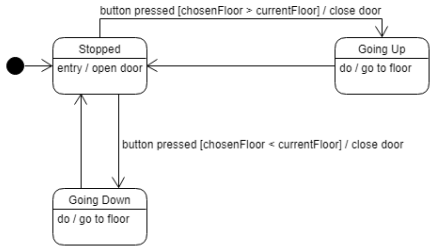
- Transição:
  - Evento (nome do evento)
  - Condição: expressão lógica (**if** implícito)
  - Comportamento: lista de ações separadas por ";"
    - Atribuição, chamada de função, ativação de saída, etc.

evento [condição] / comportamento ⟶

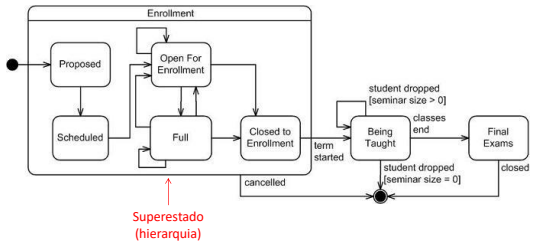
Eventos x Condições

- Tipos de eventos:
  - Chamada de função
  - Chegada de sinal assíncrono: IRQ, mensagem
  - Passagem do tempo (contada a partir da entrada no estado): **after(período)** ou **at(instante)**
- Tipos de condições:
  - Operadores lógicos: ==, !=, <, >, ...
  - Operadores genéricos: is_in(estado), ...
  - [else]

Exemplo: Controle de Elevador



Exemplo: Processo de Matrícula



Exercício 1

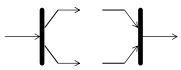
- Esboce um diagrama de estados que modele o comportamento dinâmico das trocas de modo de operação de um núcleo ARM Cortex-M4
- Considere apenas a situação de operação em que os registradores especiais PRIMASK = 1, FAULTMASK = 1 e CONTROL = 0
- Comece questionando quais são os eventos de interesse para a situação de operação acima

Exercício 1

- Considere estados intermediários de *stacking*, *unstacking* e *tail-chaining*, se for o caso
- Na representação das transições ocasionadas pela execução de uma instrução que ocasione retorno de exceção (ex: BX LR), considere possíveis exceções pendentes (bit)
- Utilize a notação UML para a caracterização das transições: evento[condição]/ação

Notação Gráfica Avançada

- Pseudo-estado inicial ●
- Pseudo-estado final ◉
- Junção •
- Seleção ◇
- Terminação ×
- Fork / Join (concorrência)

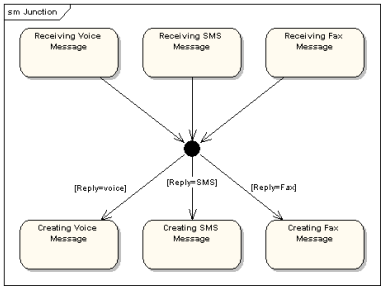


- História rasa e história profunda Ⓗ Ⓗ*

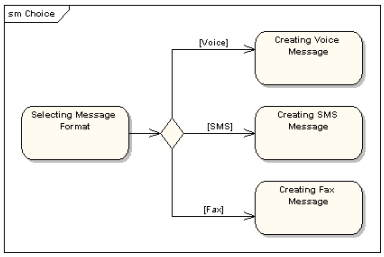
Pseudo-estados Inicial e Final

- Os estados ligados ao pseudo-estado inicial são aqueles nos quais o sistema pode entrar quando é inicializado
  - Pelo menos um é necessário
  - Se houver mais de um, as condições de entrada para cada estado devem ser especificadas
- Os estados ligados ao pseudo-estado final são aqueles dos quais o sistema não mais sairá
  - Qualquer quantidade é permitida
  - Sem transições para outros estados

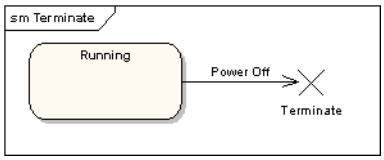
Exemplo: Junção



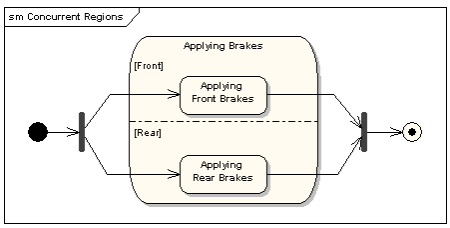
Exemplo: Seleção



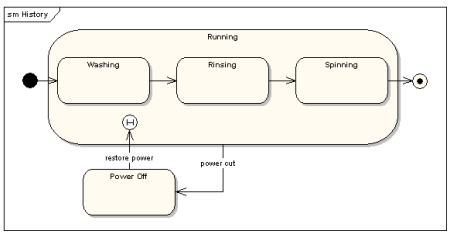
Exemplo: Terminação



Exemplo: Concorrência (Fork/Join)



Exemplo: História Rasa



Implementação de MEF

- Abordagens:
  1. Seleção por Estado
  2. Seleção por Evento
  3. Matriz Estado-Evento
- Identificação de estados:
  - Situação (saídas)
  - Memória (variáveis)

Implementação de MEF

- O estado corrente é guardado em uma variável, normalmente uma enumeração dos estados que compõem a MEF
- O evento é detectado (exemplo: IRQ) e sua ocorrência informada por:
  - Variável alterada por ISR (*bare metal*)
  - Mensagem assíncrona de ISR para tarefa (RTOS)

Seleção por Estado

```
typedef enum {Estado_0, Estado_1, Estado_2} state_t;
volatile uint8_t Evento = 0; // alterado por ISR

void tarefa(void){
  state_t Estado = Estado_0; // estado inicial da MEF
  while(1){
    switch(Estado){
      case Estado_0:
        if(Evento == 1){
          // ações e mudança de estado
        } // if
        break;
      ⋮
    } // switch
  } // while
} // tarefa
```

Projeto "fsm_state" da área de trabalho "EK-TM4C1294XL_IAR8"

Exercício 2

- Esboce um diagrama de estados que descreva o comportamento dinâmico do projeto "fsm_states"
- Como implementar ações de entrada e saída (entry / e exit /) nos estados e suas atividades (do /) na abordagem de seleção por estado?

Seleção por Evento

```
typedef enum {Estado_0, Estado_1, Estado_3} state_t;
volatile uint8_t Evento = 0; // alterado por ISR

void tarefa(void){
  state_t Estado = Estado_0; // estado inicial da MEF
  while(1){
    if(Evento){
      switch(Estado){
        case Estado_0:
          // ações e mudança de estado
          break;
        ⋮
      } // switch
    } // if
  } // while
} // tarefa
```

Projeto "fsm_event" da área de trabalho "EK-TM4C1294XL_IAR8"

Exercício 3

- Altere o projeto "fsm_event" de forma a apresentar a sequência crescente do Código de Gray de 3 bits nos LED D1, D2 e D3 do kit EK-TM4C1294XL
- Sugestão: utilize um estado diferente para cada padrão binário de saída
  - 000 → 001 → 011 → 010 → 110 → 111 → 101 → 100 → 000 → ...

Matriz Estado-Evento

- Define-se as funções de cada estado:

```
state_t func1(state_t curr){
  // ações
  return next; // mudança do estado
} // func1
```

- Cria-se uma matriz de ponteiros para as funções:

```
state_t (*matriz[N_EV][N_ST])(state_t) =
  {{func1, func2, func3},
   {func4, func5, func6}};
```

- Executa-se a função correspondente a cada evento detectado:

```
Estado = (*matriz[Evento][Estado])(Estado);
```

Implementação Matriz Estado-Evento

```
state_t f_0(state_t curr){
  // ações
  return next; // mudança de estado
} // f_0
!
void tarefa(void){
  state_t (*matriz[N][M])(state_t) = {{f_0, f_1, f_2},
                                     {f_3, f_4, f_5}};
  state_t Estado = Estado_0; // estado inicial da MEF
  while(1){
    if(Evento){
      Estado = (*matriz[Evento - 1][Estado])(Estado);
      Evento = 0;
    } // if
  } // while
} // tarefa
```

Projeto "fsm_matrix" da área de trabalho "EK-TM4C1294XL_IAR8"

Exercício 4

- Esboce um diagrama de estados que utilize o conceito de hierarquia para descrever o comportamento dinâmico do projeto "fsm_matrix"
- Como implementar ações de entrada e saída (entry / e exit /) nos estados e suas atividades (do /) na abordagem de seleção por matriz estado-evento?

Material Suplementar

- Software based Finite State Machine (FSM) with general purpose processors (Joseph Yiu)
- Blog: Máquina de Estados em C (Sergio Prado):
  - https://sergioprado.org/maquina-de-estados-em-c/
- Vídeos: Diagramas de Estados (YouTube):
  - https://www.youtube.com/watch?v=_6TFVzBW7oo
  - https://www.youtube.com/watch?v=UzUUZRK_Q6Y
  - https://www.youtube.com/watch?v=ABA3TGQVhTg

Ferramentas Úteis

- Draw.io (desenho na nuvem)
  - https://www.draw.io/
- UMLetino (desenho na nuvem)
  - http://www.umlet.com/umletino/umletino.html
- Yakindu Statechart Tools (desenho/simulação)
  - http://statecharts.org/

Referências

- UML State Machine Diagrams
  - http://www.uml-diagrams.org/state-machine-diagrams.html
- State Machine Diagrams: An Agile Introduction
  - http://agilemodeling.com/artifacts/stateMachineDiagram.htm
- Sparx Systems – State Machine Diagram Tutorial
  - https://sparxsystems.com.au/resources/tutorials/uml2/state-diagram.html
- Lucidchart - State Machine Diagram Tutorial
  - https://www.lucidchart.com/pages/uml-state-machine-diagram