TP2 - Módulo de início de sessão modelado em UPPAAL

Jose Clavo Tafur[1]

[1]Departamento de Ciência da Computação – Universidade de Brasilia (UnB)

Resumo. *Este artigo descreve o modelado de um sistema de transição referente ao módulo de início de sessão de um sistema X, para o qual foi usada a ferramenta UPPAAL. juntamente com isso são especificadas suas propriedades CTL e sua verificação respetivamente.*

1. Introdução

Antes de desenvolver um sistema o uma parte de ele, é preciso modelar e verificar que aquele modelo atenda a especificação requerida, em esse momento, é onde o uso de algumas ferramentas é indispensável. Em este artigo **UPPAAL** será usado, o qual é uma ferramenta para modelar, simular e verificar sistemas. Além disso, tem 3 principais características como seu *linguagem de descrição*, *simulador* e *verificador de modelos* [UPPAAL's-team 2019], as quais nos ajudaram em este processo.

2. Descrição do sistema analisado

O análises será focalizado em um especifico módulo do sistema, o qual é o **início de sessão**.

2.1. Requisitos

Aqueles requisitos com um * serão usados para criar o modelo. O resultado final no site sera como é mostrado na (Figura 1)

- A pagina deve ser a primeira mostrada para o usuário quando acesse ao sistema.
- Os campos usuário e senha são obrigatários.
- Se as credencias dão certas o acesso ao sistema sera permitido, *fluxo na (Figura 6)**
- Se as credencias são incorretas, deve voltar a pagina do início de sessão, *fluxo na (Figura 8)**
- Depois de 3 acessos incorretos, o usuário sera bloqueado 5 minutos, *fluxo na (Figura 9)**
- Adicionar a opção de "esqueceu senha".
- Adicionar a opção de "mudar idioma".

2.2. Arquitetura

A arquitetura do módulo de início de sessão é mostrado na (Figura 2) onde o fluxo que sera modelado esta selecionado de cor verde.

3. Modelagem em UPPAAL

O modelo tem 3 templates, o principal representa o início de sessão e é mostrado na (Figura 3), também tem o template que representa as ações do usuário (Figura 4), finalmente representando a um observador (Figura 4).

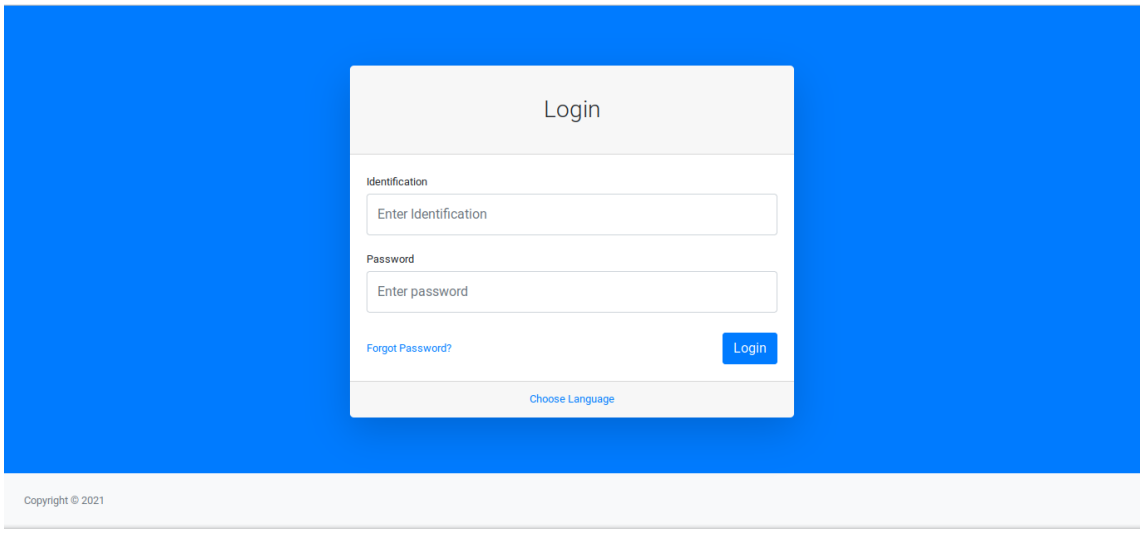

Figure 1. Módulo de início de sessão.

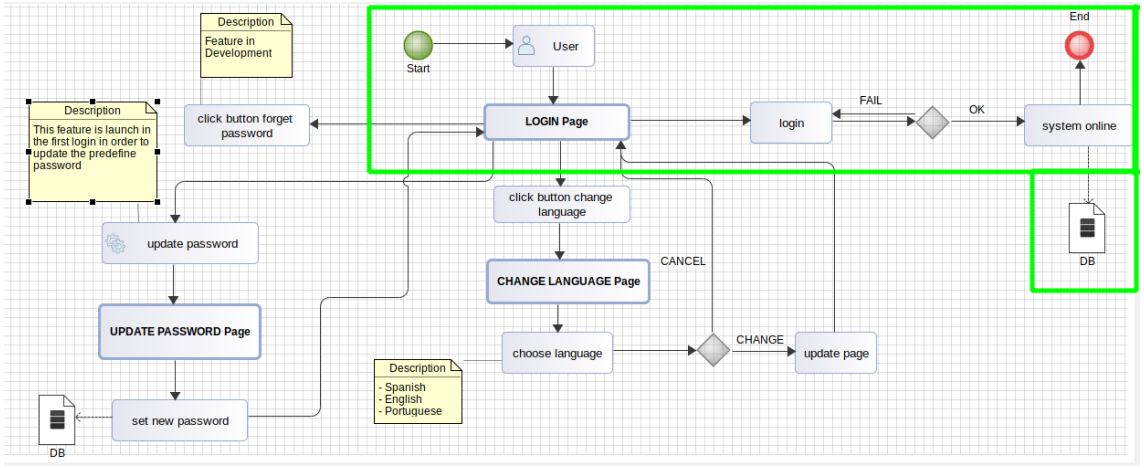

Figure 2. Arquitetura do módulo de início de sessão

3.1. Templates

3.1.1. Template do início de sessão: LOGIN

Este template (Figura 3) é o principal representa o fluxo do início de sessão e tem os seguintes elementos.

- Estados.
  - **initial**: o estado inicial.
  - **login**: é automaticamente alcançado depois do $S_{initial}$
  - **validating**: simula a conexão ao banco de dados, verificando se as credencias são corretas ou não.
  - **app**: representa ao aplicativo quando é acesso, em este caso não faz muito so esperar saída do sistema.
  - **blocked**: é alcançado quando o usuário erra mais de 3 acesso ao sistema. Além disso, tem a propriedade de ser "committed" o que significa que o seguinte estado $S_{initial}$ sera alcançado imediatamente.

- Variáveis
  - **connected**: é um "booleano", recebe o valor 1 so quando as credencias dão certas e alcançá o estado S_{app}
  - **countFails**: é um "inteiro", age como um contador dos erros do acessos ao sistemas, incrementando em 1 por cada erro, usando o "template observador". (Figura 5), se seu valor é > 3 o estado $S_{blocked}$ é alcançado.
- Relógio ou clock.
  - **x**: este relógio é usado para o contagem do tempo que estaria no estado $S_{blocked}$.
- Função.
  - **initialize**(): inicializa os valores para as variaveis *connected = 0* e *countFails = 0*, e para o relogio ou "clock" *x = 0*.
- Sentinelas ou guards.
  - **countFails<3**: se a condição é certa, o estado $S_{validating}$ é alcançado e isso significa que ainda não tem o acesso bloqueado.
  - **countFails>3**: se a condição é certa, o estado $S_{blocked}$ é alcançado sendo o gatilho para o bloqueio.
- Invariante.
  - **x<300**: representa a condição pra sair do estado $S_{blocked}$ usando o relógio x para o contagem.
- Canais.
  - **credentials**: esta na escuta das credencias do usuário enviadas por o "template usuário". (Figura 4)
  - **logout**: esta na escuta que o usuário role a ação sair do sistema, a qual é enviada por o "template usuário". (Figura 4)
  - **fail**: envia um mensagem para o "template observador" (Figura 5) quando as credencias para o início de sessão são incorretas.

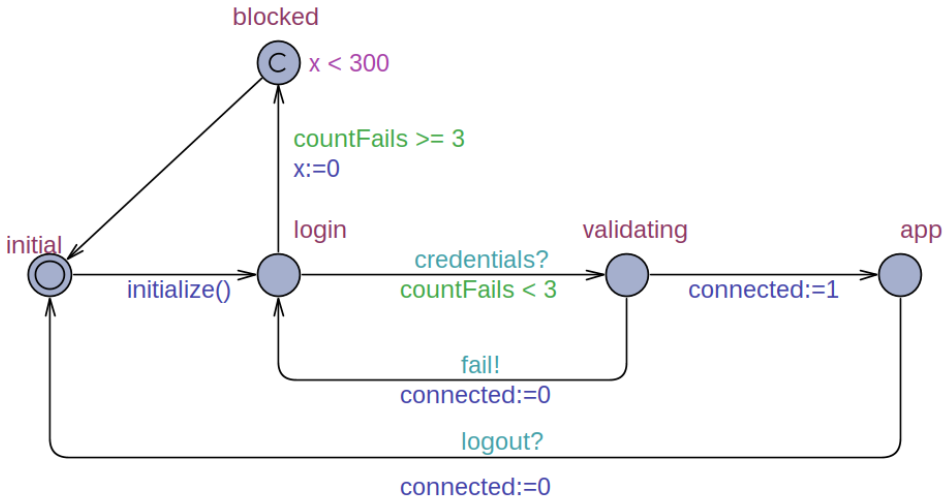

Figure 3. Template do início de sessão: LOGIN

3.1.2. Template do usuário: USER

Este template (Figura 4) representa as ações de fazer login e sair do sistema realizadas por o usuário e tem os seguintes elementos.

- Estados.
  - **idle**: o estado inicial.
  - **in**: representa a ação quando o usuário tenta fazer login, além disso, tem a propriedade de ser "committed".
  - **out**: representa a ação quando o usuário tenta sair do sistema, além disso, tem a propriedade de ser "committed".
- Relógio ou clock.
  - **y**: este relógio é usado para o contagem do tempo que estaria no estado S_{idle}, antes de alcançar outro estado. sendo reiniciado depois de alcançar eles.
- Sentinelas ou guards.
  - **connected==0**: usando a variável global "connected" , se a condição é certa o que significa o usuário não ha iniciado sessão sera possível alcançar o estado S_{in}
  - **connected==1**: usando a variável global "connected" , se a condição é certa o que significa o usuário ha iniciado sessão sera possível alcançar o estado S_{out}
- Invariante.
  - **y<100**: representa a condição que permite ao usuário sair do estado S_{iddle} usando o relógio y para o contagem.
- Canais.
  - **credentials**: envia um mensagem para o "template do inicio de sessão" (Figura 3) quando o usuário tenta fazer login.
  - **logout**:envia um mensagem para o "template do inicio de sessão" (Figura 3) quando o usuário tenta sair do sistema.

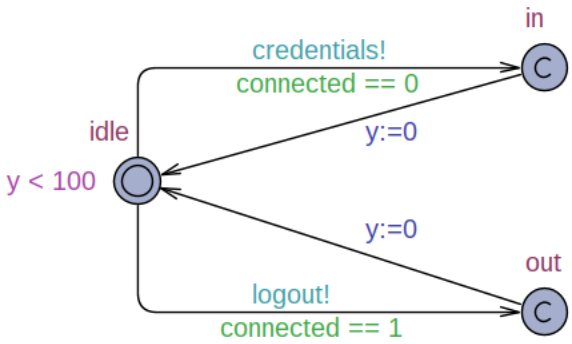

Figure 4. Template do usuário: USER

3.1.3. Template do observador: OBSERVER

Este template (Figura 5) representa a um observador, que faz o contagem quando as credencias são erradas e tem os seguintes elementos.

- Estados.
  - **idle**: o estado inicial.
  - **taken**: representa a ação quando as credencias dão erradas, além disso, tem a propriedade de ser "committed". e quando volta para o estado S_idle incrementa em 1 a variável global "countFails" a qual e usado por o "template do inicio de sessão" (Figura 3).
- Canal.
  - **fail**: está na escuta se as credencias deram erradas em o "template do inicio de sessão" (Figura 3).

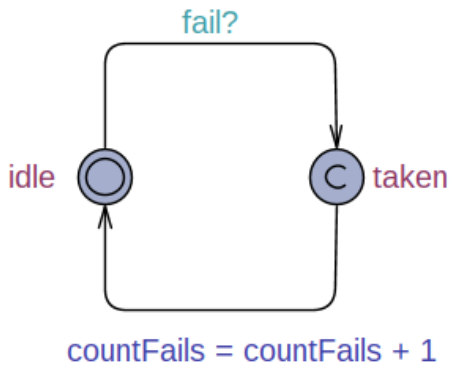

Figure 5. Template do observador: OBSERVER

3.2. Fluxos

3.2.1. Inicio de sessão: sucesso

Como é mostrado na (Figura 6), o estado *initial* **(1)** começa o fluxo, inicializando os valores. logo o estado *login* **(2)** é alcançado e fica na escuta através do canal "credentials". Logo depois o usuário tenta fazer login o qual é possível porque o sentinela é cumprido já que a variável "connected" tem o valor "0" então dispara um mensagem ao mesmo tempo o relogio "y" e reinicializado **(3)**, este mensagem é capturado e como não houve credencias incorretas anteriormente, a variável "countFails" cumpre a condição do sentinela então o estado *validating* **(4)** é alcançado, logo as credencias são validadas e o resultado é correto o qual permite acessar ao sistema alcançando o estado *app* **(5)**. Além disso, a variável "connected" recibe o valor 1 indicando que o usuário acessou ao sistema.

3.2.2. Sair do sistema

O detalhe do ponto **(1)** já foi explicado na (Figura 6) encontrando-se agora no estado *app* e fica na escuta através do canal "logout". Logo depois o usuário tenta sair do sistema o

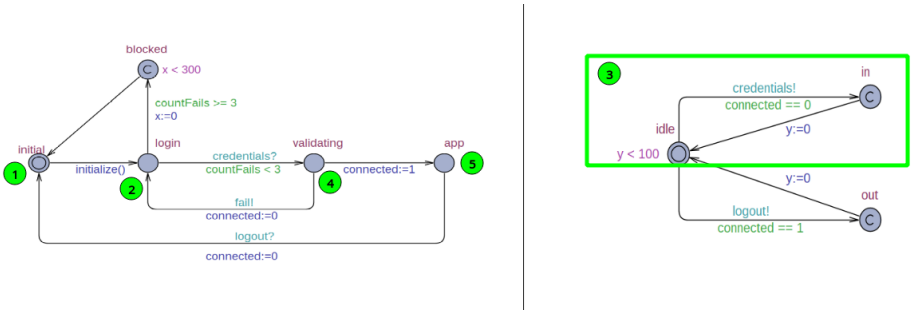

Figure 6. inicio de sessão: sucesso

qual é possível porque o sentinela é cumprido já que a variável "connected" tem o valor "1" porque em esse momento o usuário esta no sistema, então dispara um mensagem ao mesmo tempo o relogio "y" e reinicializado **(2)**, este mensagem é capturado e o modelo volta para o estado *initial* **(3)**, Além disso, a variável "connected" recibe o valor 0 indicando que o usuário saiu do sistema. O explicado é mostrado na (Figura 7)

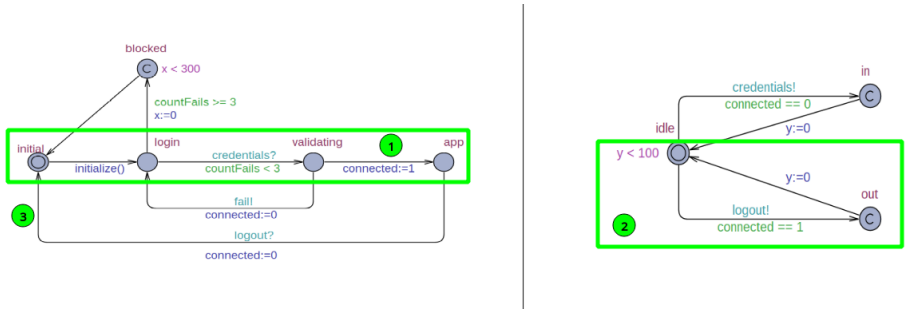

Figure 7. sair do sistema

3.2.3. Inicio de sessão: falha

Os detalhes dos pontos **(1,2,3,4)** já foram explicados na (Figura 6) encontrando-se agora no estado *validating*, logo as credencias são validadas e o resultado da erro então dispara um mensagem, , este mensagem é capturado **(5)** alcançando o estado *taken* e ao mesmo tempo o variável "countFails" é incrementa seu valor em 1. Simultaneamente o modelo volta para o estado *login* **(6)** , além disso, a variável "connected" recibe o valor 0 indicando que o usuário ainda não acessou ao sistema. O explicado é mostrado na (Figura 8)

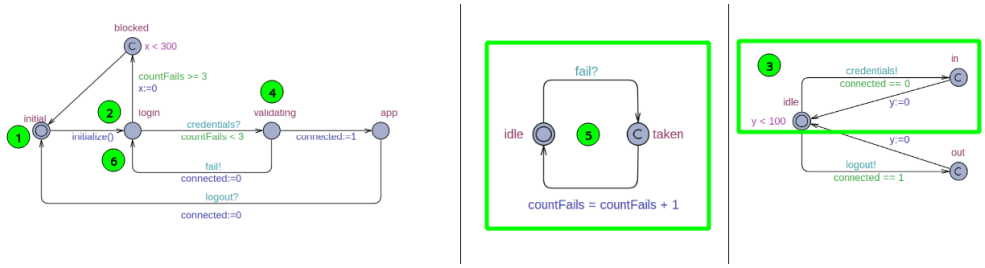

Figure 8. inicio de sessão: falha

3.2.4. Inicio de sessão: bloqueado

O detalhe do ponto **(1)** já foi explicado na (Figura 8) encontrando-se agora no estado *login*, se este fluxo de falha acontece 3 vezes, a condição do sentinela para o estado *blocked* **(2)** é cumprido, além disso, o relógio "x" é reinicializado. Estando no estado *blocked* aguardando a condição da invariante cumprir usando o relógio para o contagem, posteriormente, o modelo volta para o estado *initial* **(3)**.

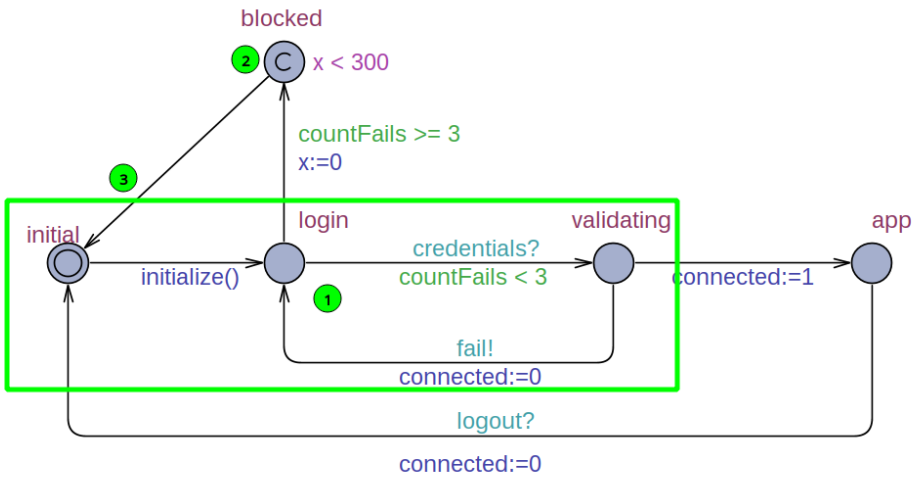

Figure 9. inicio de sessão: bloqueado

4. Especificação das Propriedades em CTL

4.1. Deadlock

O modelo encontra-se livre de "deadlock", consequentemente, a propriedade é satisfeita.

A [] not deadlock

4.2. Reachability

- Existe algum caminho, que eventualmente vai alcançar o estado $S_{blocked}$ no template do início de sessão?
  A resposta é afirmativa, por conseguinte, a propriedade é satisfeita.

E <> Login.blocked

- Existe algum caminho, que eventualmente vai alcançar o estado S_{app} no template do início de sessão?
  A resposta é afirmativa, assim, a propriedade é satisfeita.

E <> Login.app

4.3. Bounded Liveness

- Para todos los caminhos, sempre que o estado $S_{blocked}$ no template do início de sessão seja alcançado significara que a variável "countFails" é maior ou igual a 3?
  A resposta é afirmativa, portanto, a propriedade é satisfeita.

A[] (Login.blocked $imply$ **countFails>=3)**

- Para todos los caminhos, sempre que o estado S_{in} no template do usuário seja alcançado significara que o relógio "y" é menor que 100? A resposta é afirmativa, de modo que, a propriedade é satisfeita.

E[] (User.in $imply$ **User.y<100)**

4.4. Liveness

No template do início de sessão, estando no estado $S_{blocked}$ eventualmente o estado S_{login} sera alcançado?
A resposta é afirmativa, desse modo, a propriedade é satisfeita.

Login.blocked $--> $ **Login.login**

4.5. Mutex

No template do usuário, os estados S_{in} e S_{out} nunca serão alcançados ao mesmo tempo?
A resposta é afirmativa, por conseguinte, a propriedade é satisfeita.

A[] not (User.in and User.out)

5. Resultados Obtidos e Análise

- Foi modelado o fluxo do módulo de início de sessão, usando dois modelos auxiliares que representam as ações do usuário e a um observador. Estos modelos ao ser testados tiveram um resultado ótimo.
- O uso das propriedades em CTL foi de muita ajuda na verificação dos modelos criados.
- A verificação de modelos é um ótimo abordagem para ser aplicado em cenários práticos.
- O arquivo com o conteúdo do artigo no formato UPPAAL pode ser baixado aqui:

6. Conclusão

UPPAAL é uma ferramenta muito boa no momento de fazer o modelagem, e não só de um módulo do sistema, é possível modelar tudo ele com esta ferramenta. Alem disso, tem a opção de simular e verificar o modelo criado. Na parte da verificação permite a especificação e uso das propriedades em CTL. No processo de modelagem esta ferramenta foi de muita ajuda para mim. Ademais, outra fonte de ajuda foi o livro [Baier 2008], o qual revisamos na aula e tem muita informação interessante sobre as propriedades. Finalmente, eu recomendo o uso desta ferramenta para o modelagem, ao mesmo de um módulo pequeno como foi comprovado em este artigo.

References

[Baier 2008] Baier, C., . K. J. P. (2008). *Principles of model checking*. MIT press.

[UPPAAL's-team 2019] UPPAAL's-team (2019). Uppaal features.