Instituto Tecnológico de Aeronáutica

Divisão de Ciência da Computação

Graduação em Engenharia de Computação

Exercícios de Laboratório

Douglas Yamashita de Moura

Dr. Adilson Marques da Cunha

CES – 63 – Sistemas Embarcados

3º Ano Profissional – COMP07

São José dos Campos, 22 de Outubro de 2007

1. Objetivos

Estes laboratórios de 1 a 7 (*Case Study Lab*) têm por finalidade fazer com que os alunos das Disciplinas CES-63 ponham em prática as principais funcionalidades aprendidas do *Rational Rose Real Time – RRRT* na execução do *Take Home Take Lab Test* – THTLT e possam utilizar os conhecimentos adquiridos no desenvolvimento do seu Componente de Software de Computador – CSC, que é constituinte do Item de Configuração de Software de Computador – ISC, resultando no Sistema de Software de Computador – SSC.

2. Conteúdo

Foram realizados os sete labortatórios do *Case Study Labs* do *Rational Rose Real Time – RRRT* e as principais realizações de cada um estão relacionadas abaixo.

2.1. Laboratório 1 – *Controller / Tester*

Neste primeiro laboratório, foi criada uma versão simples do *DyeingSystem* utilizando-se de *capsule classes* existentes *Controller* e *Tester*.

Os objetivos deste laboratório foram:

- Utilizar as ferramentas básicas apresentadas;
- Construir um modelo simples a partir de elementos existentes;
- Compilar, executar, testar e debugar o modelo;
- Gerar um diagrama de seqüência a partir do modelo em execucação;
- Criar um diagrama de classe do modelo.

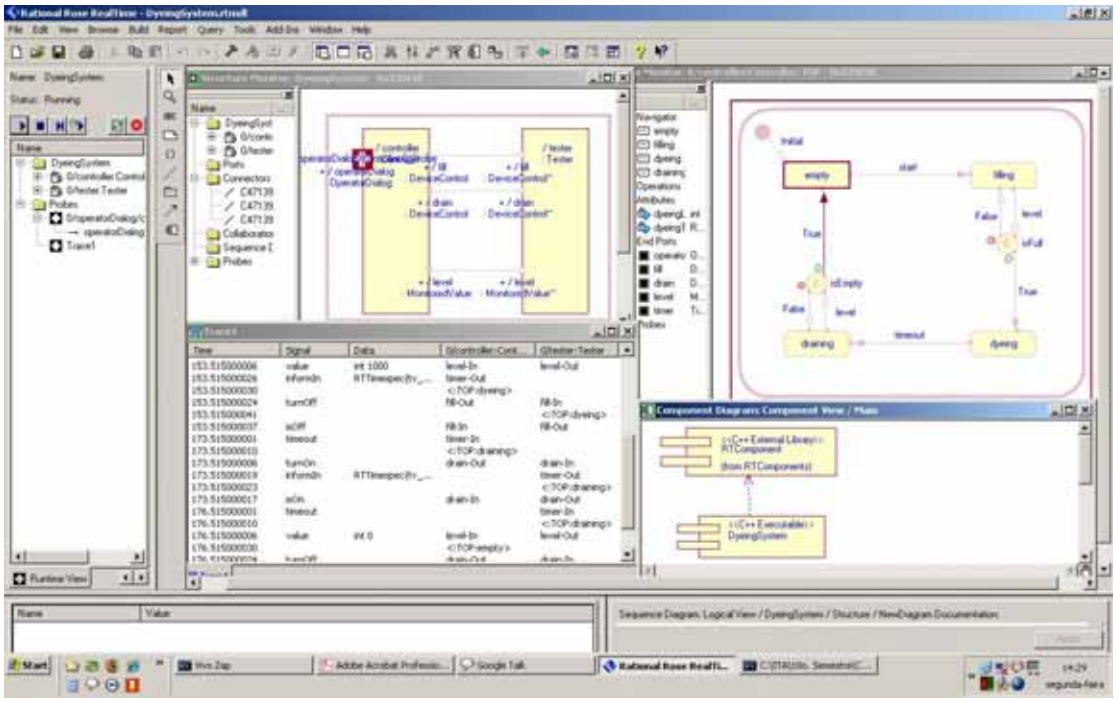

Figura 1 – *Structure Monitors* e *Trace.*

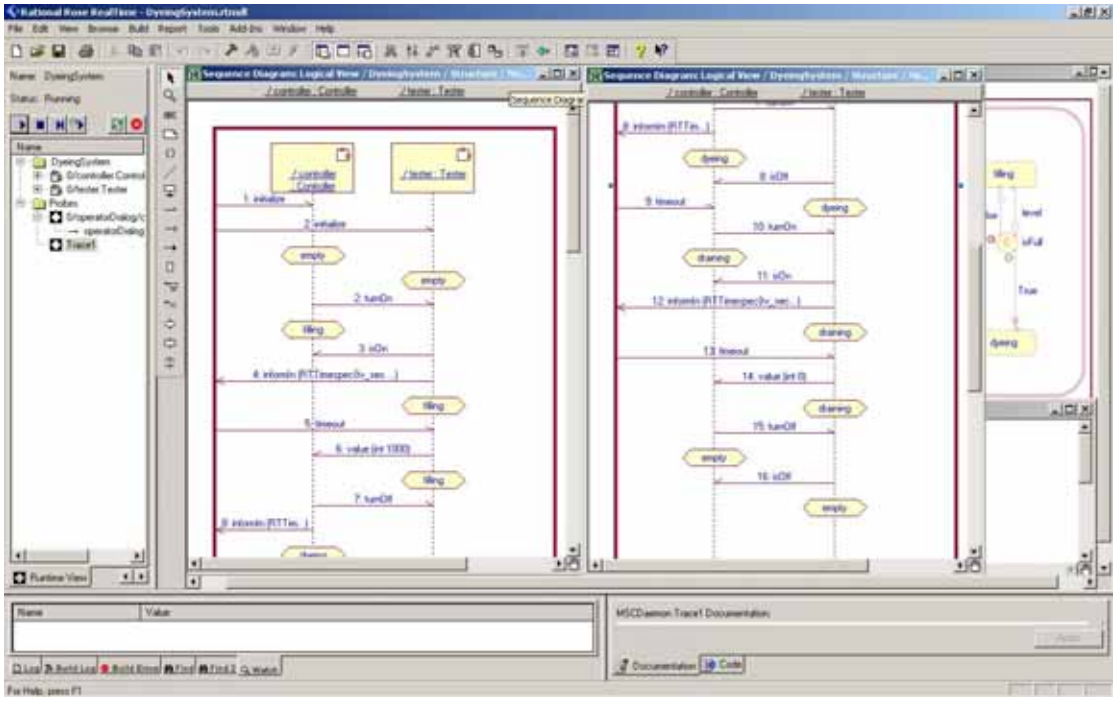

Figura 2 – Diagrama de seqüência.

2.2. Laboratório 2 – *Valve, Start Low-Temp System*

Neste segundo tutorial, foram criadas *capsules* com estruturas e comportamentos simples. Foram construídos o protocolo *Flow* e a *capsule class*

Valve, e o modelo foi compilado e debugado. Posteriormente, foram adicionados as *capsule roles* da *Valve* ao *DyeingSystem.*

Os objetivos deste laboratório foram:

- Criar uma *capsule* com estrutura simples e comportamento simples;
- Iniciar a construção de um modelo complexo (o caso de studo);
- Criar uma classe de protocolo simples.

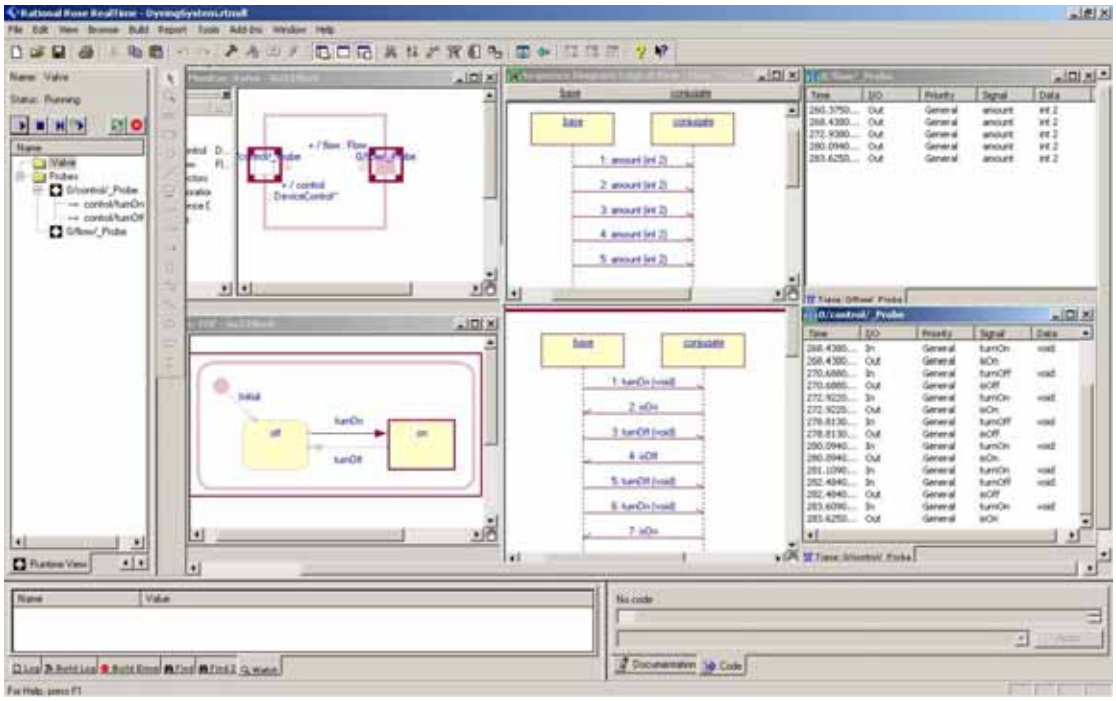

Figura 3 – *Structure Monitor*, diagrama de seqüência, diagrama de estados e *traces.*

2.3. Laboratório 3 – *Complete Low-Temp System*

Neste terceiro laboratório, foi integrado um *timer* à *capsule class Valve* para que a mesma enviasse mensagens *Flow::amount* periódicas. Foram criadas também a classe de protocolo *AcquiredValue* e as *capsule classes Level* e *Dye*. Finalmente, foram adicionadas as *capsule roles* de *Level* e *Dye* à *DyeingSystem* e foi compilado e testado o *DyeingSystem* como um sistema completo.

Os objetivos deste laboratório foram:

- Completar a construção de um modelo complexo;

- Utilizar *timers* em um modelo com o *Timing service* da biblioteca de serviços.

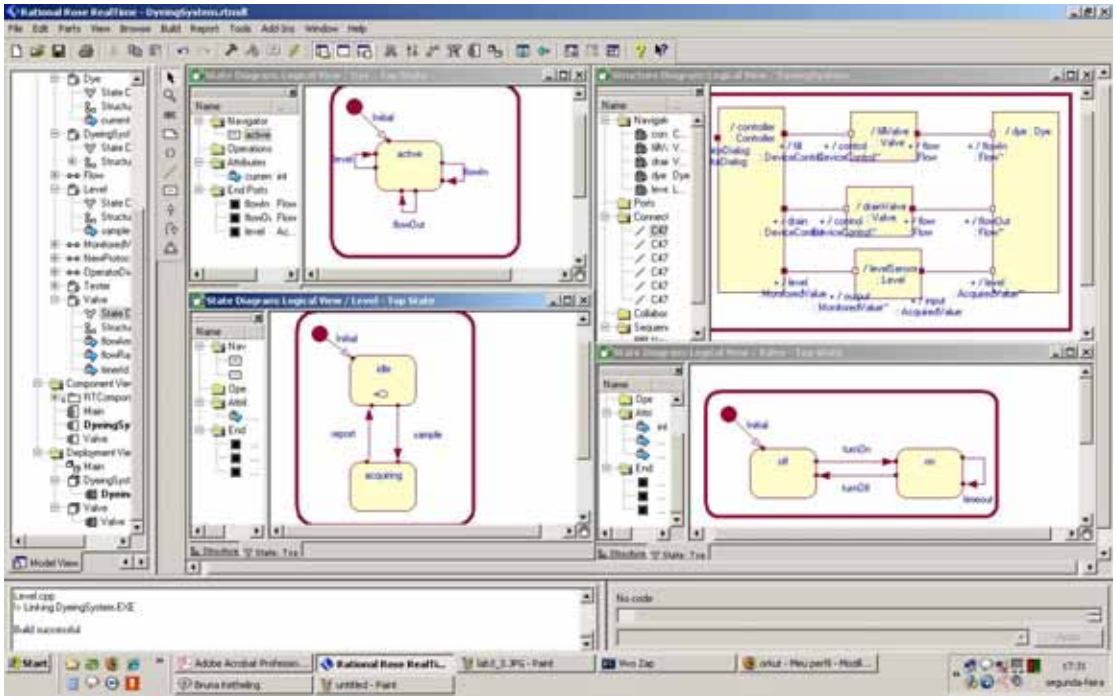

Figura 4 – Diagramas de estado e estrutura do modelo.

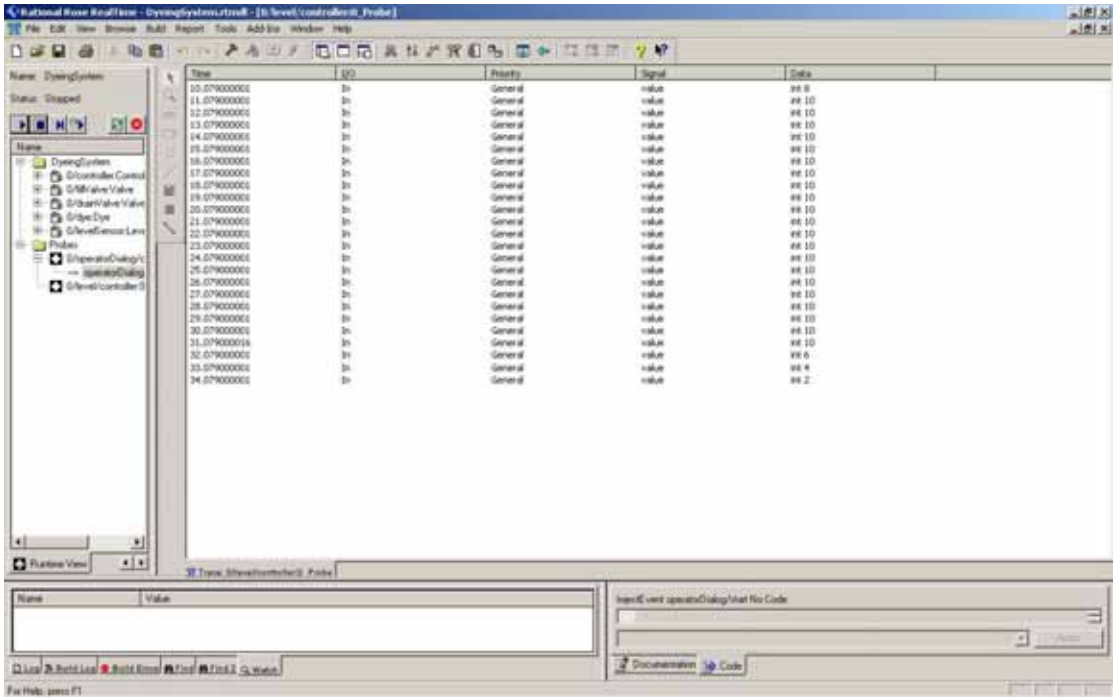

Figura 5 – *Trace.*

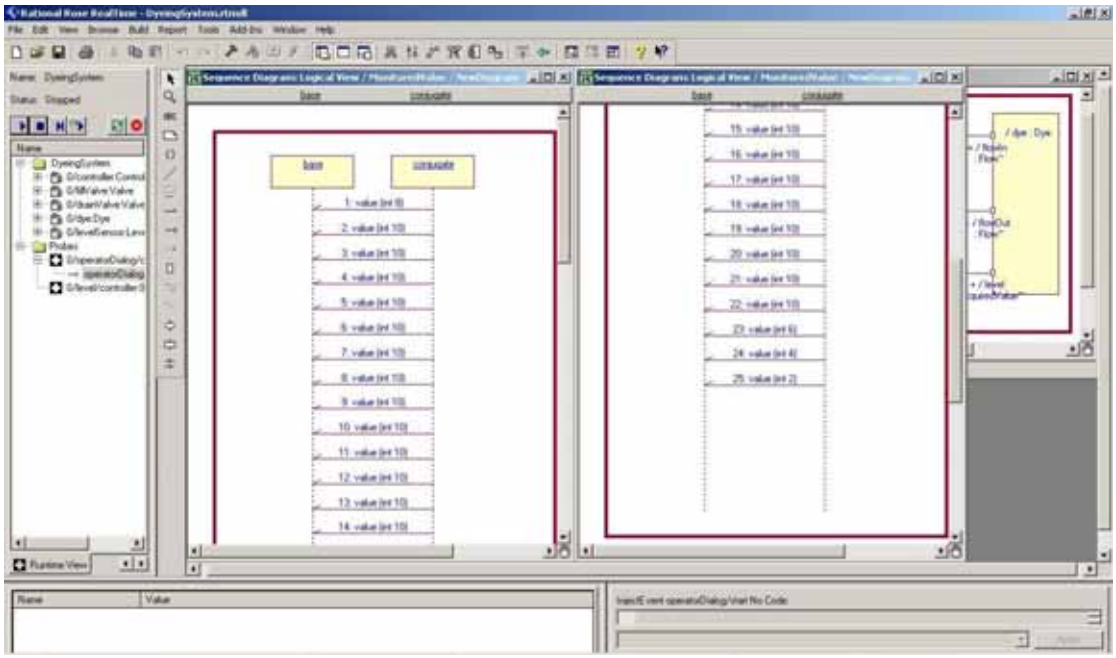

Figura 6 – Diagrama de seqüência.

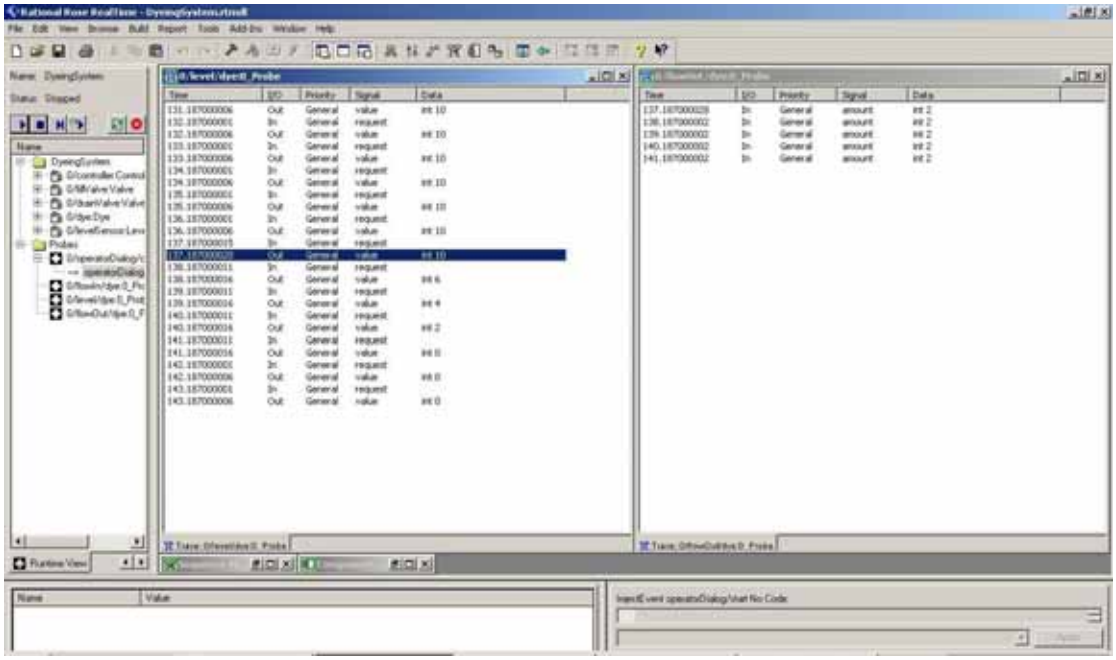

Figura 7 – *Trace.*

2.4. Laboratório 4 – *Master and Tank Containers*

Neste quarto laboratório, a tarefa foi dividida em duas etapas, nas quais houve uma reorganização do modelo do sistema de tingimento para fazer com que fosse mais fácil trabalhar com o mesmo. Na primeita etapa, foi utilizada a função *aggregation* do *Rational Rose RealTime* para criar as *container capsule classes Tank*

e *Master*. Já na segunda etapa, foi utilização replicação para criar múltiplos tanques e controladores.

Os objetivos deste laboratório foram:

- Criar estruturas hierárquicas (c*apsules* contendo outras *capsules*) em um modelo existente;
- Aplicar a técnica de agregação;
- Aplicar replicação (cardinalidade) para *capsules* e portas;
- Compilar, executar, e debugar um modelo estruturalmente complexo.

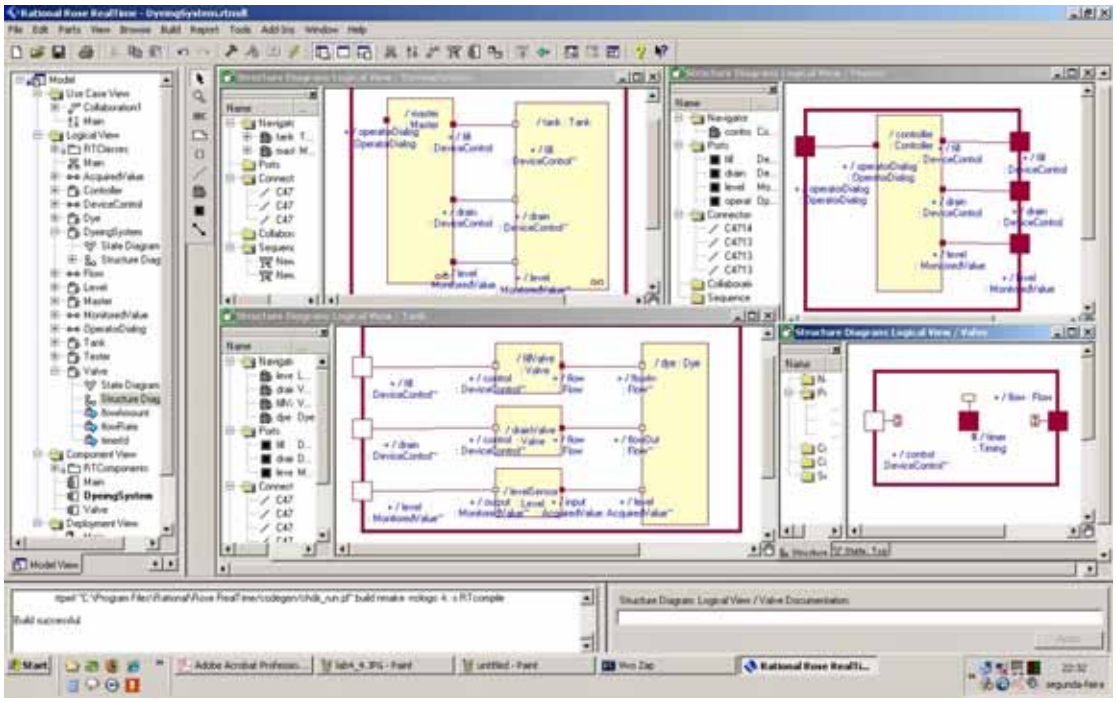

Figura 8 – Diagramas de estado e de estrutura.

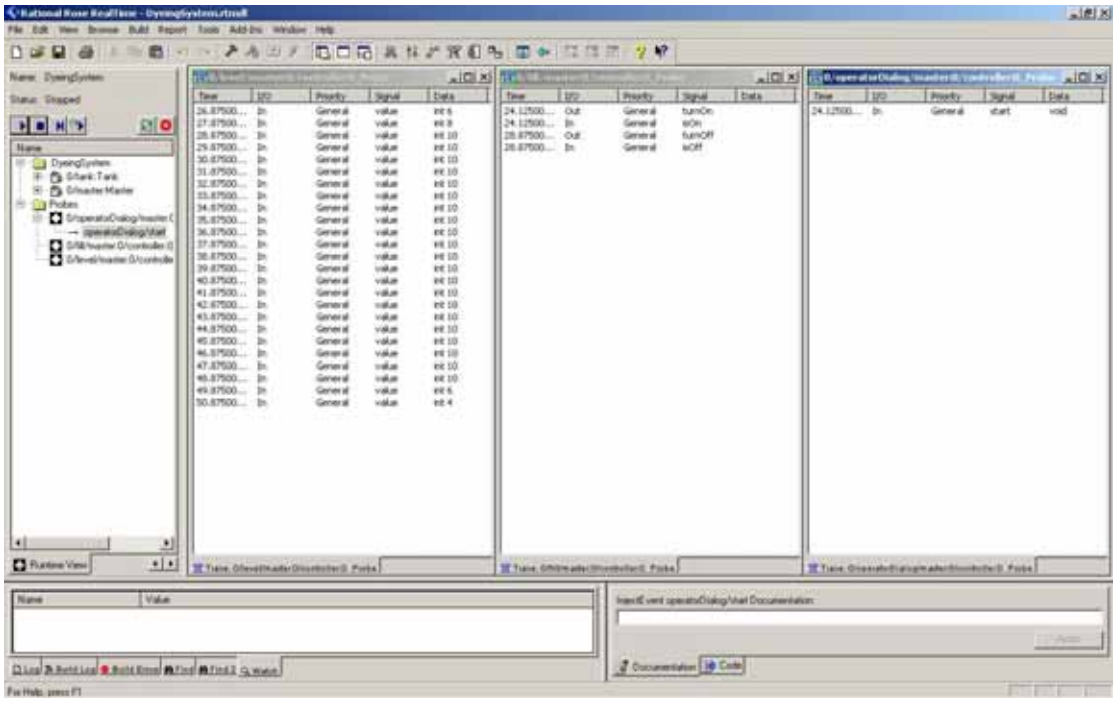

Figura 9 – *Traces* da primeira etapa.

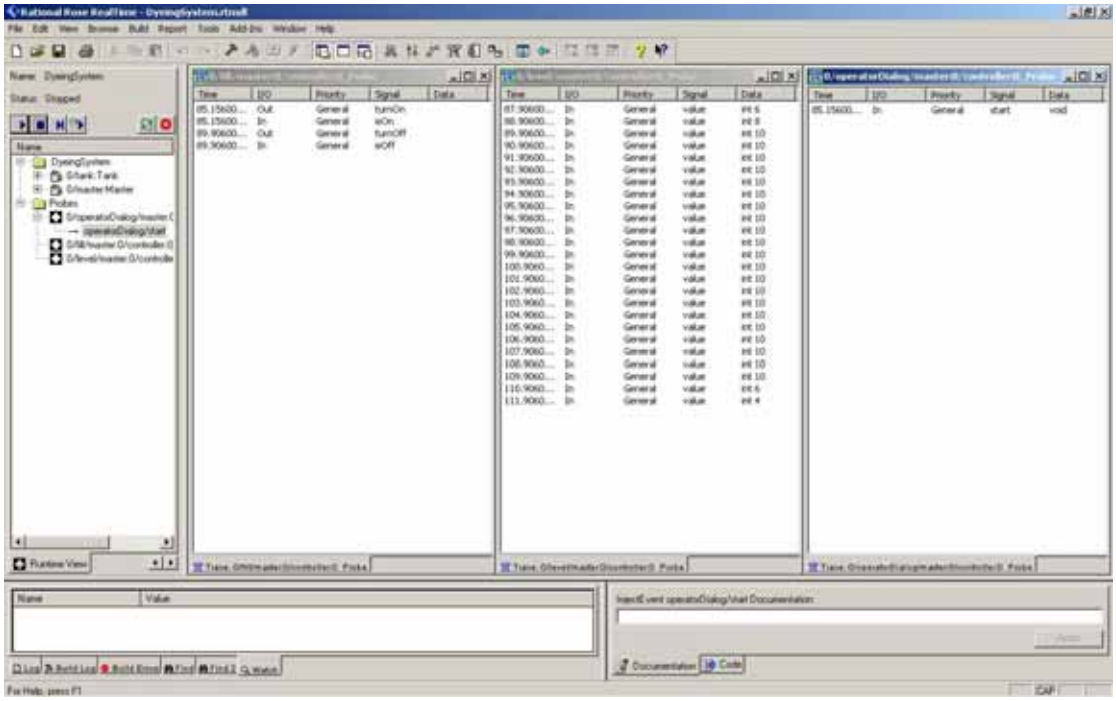

Figura 10 – *Traces* demonstrando os tempos com problemas.

Observação: Conforme pode ser observado, houve uma mudança de *int 10* para *int 6*, ao invés de ser para *int 8*. Tal fato ocorreu devido ao tempo demorado para haver a impressão dos dados, sendo que houve um decremento a mais do que o esperado. No entanto, o funcionamento do sistema mostra-se correto.

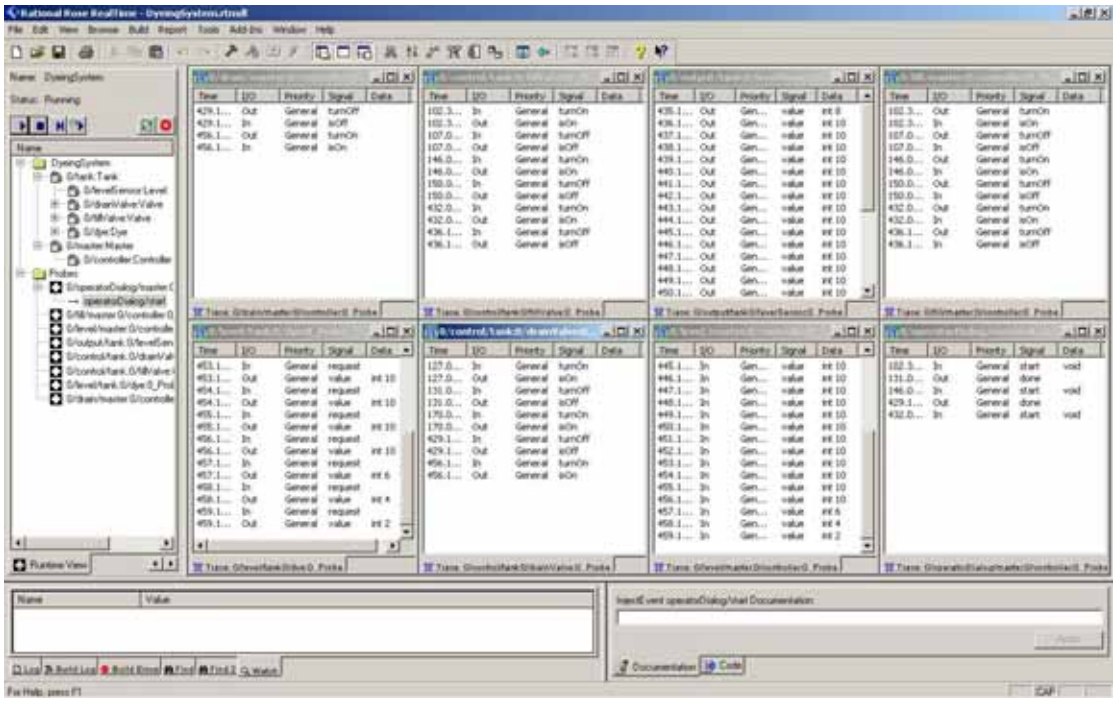

Figura 11 – *Traces* da segunda etapa.

2.5. Laboratório 5 – *Start High-Temperature System*

Neste quinto laboratório, foi assimilado o gerenciamento de complexidade de um modelo utilizando-se *inheritance hierarchies.* Foi criado um sistema de tingimento a alta temperatura através da especialização e generalização de *capsule*, protocolos, e classes passivas existentes no *DyeingSystem.*

Os objetivos deste laboratório foram:

- Criar um sistema totalmente novo através da contrução de *inheritance hierarchies*;
- Aplicação de modelamento *bottom-up* para uma *inheritance hierarchy* (generalização de uma classe existente);
- Aplicação de modelamento *top-down* para uma *inheritance hierarchy* (especialização de uma classe existente);
- Compilar, executar, e debugar um mdelo contendo *inheritance hierarchies*.

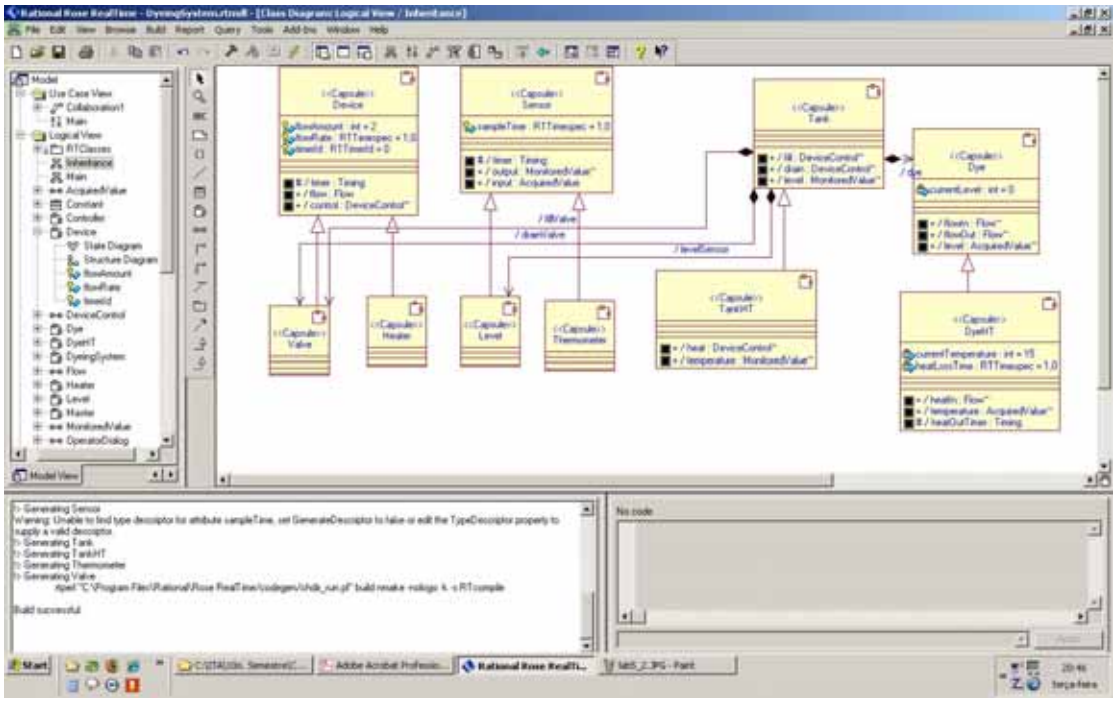

Figura 12 – Diagrama de classes.

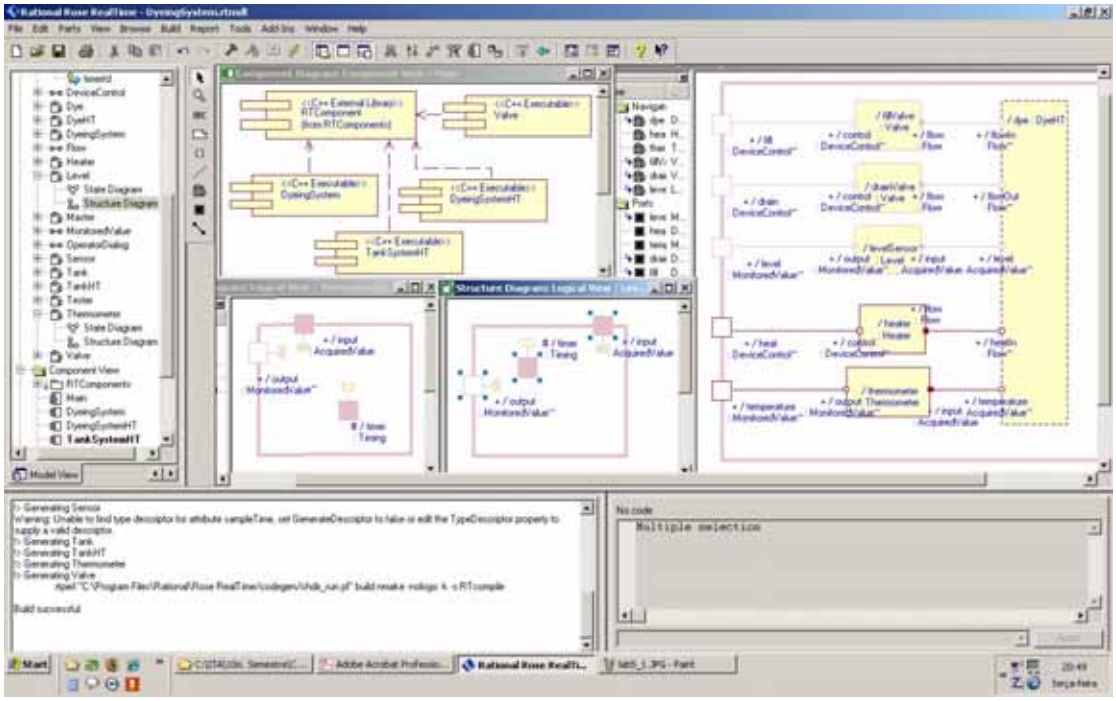

Figura 13 – Diagramas de estrutura e de estado.

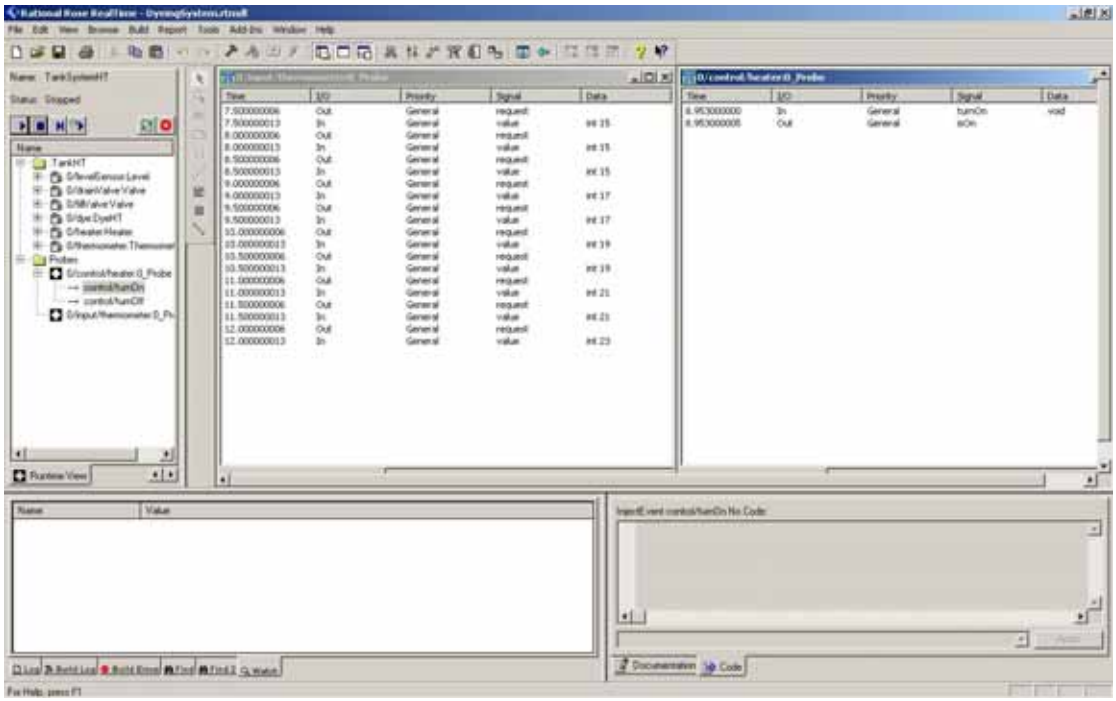

Figura 14 – *Traces.*

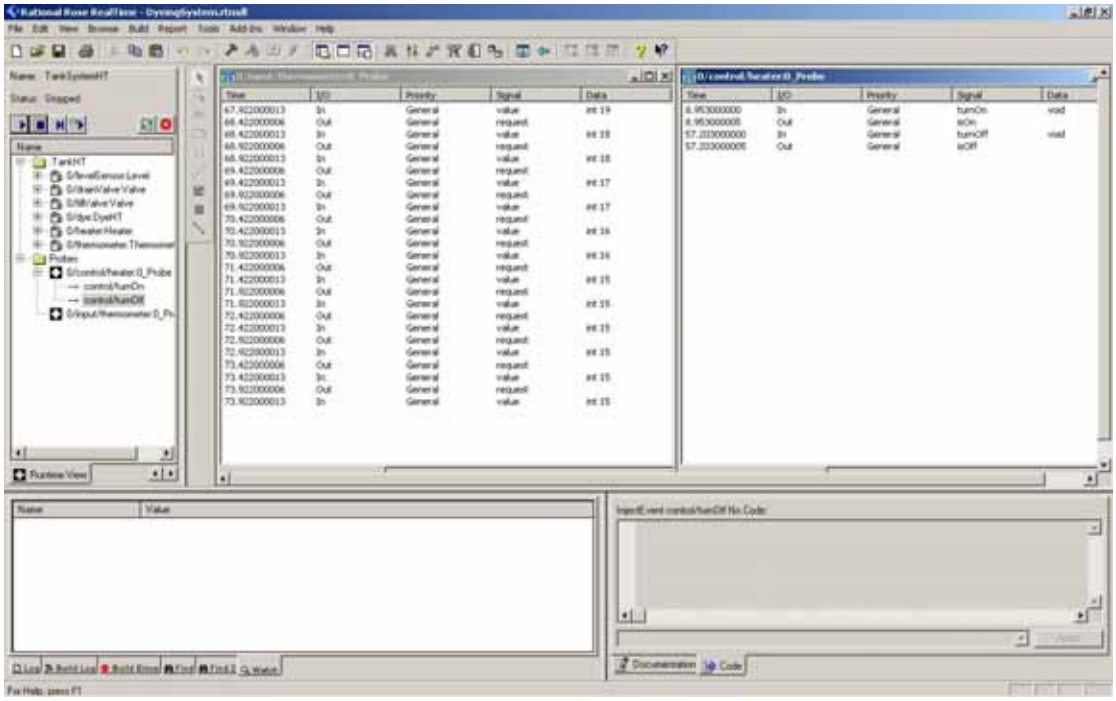

Figura 15 – *Traces.*

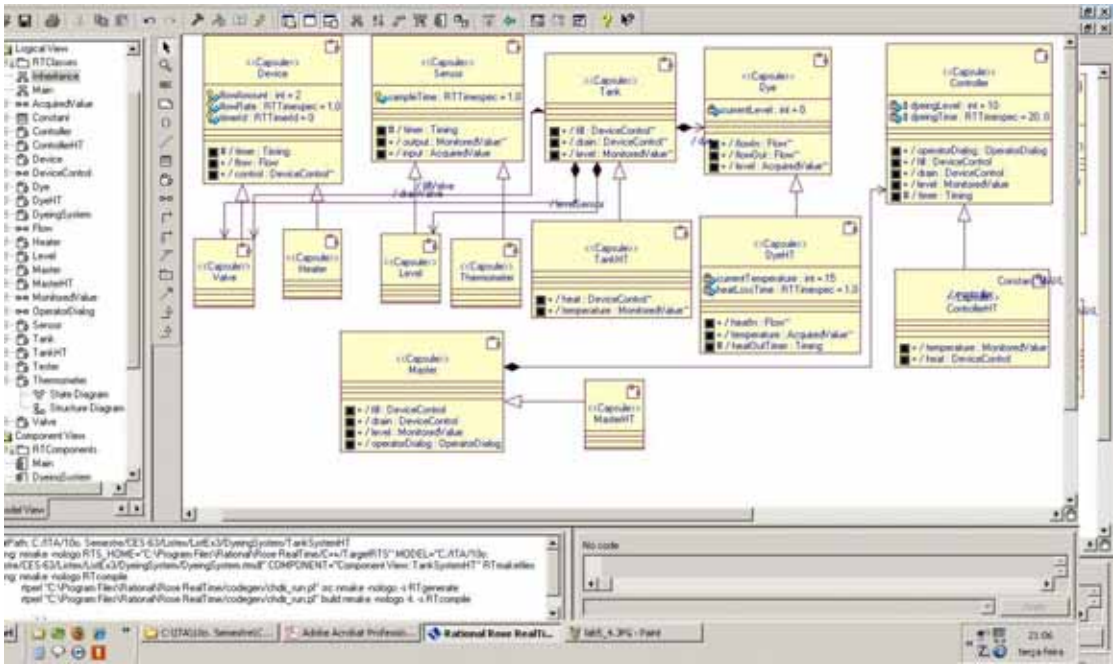

Figura 16 – Diagrama de classes.

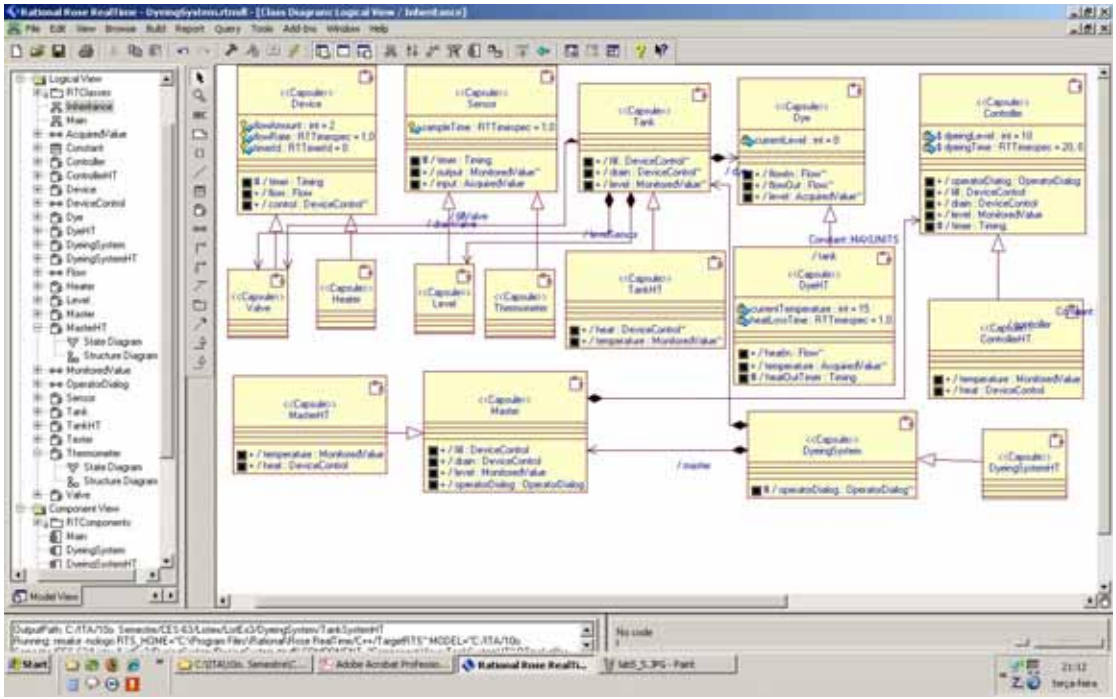

Figura 17 – Diagrama de classes.

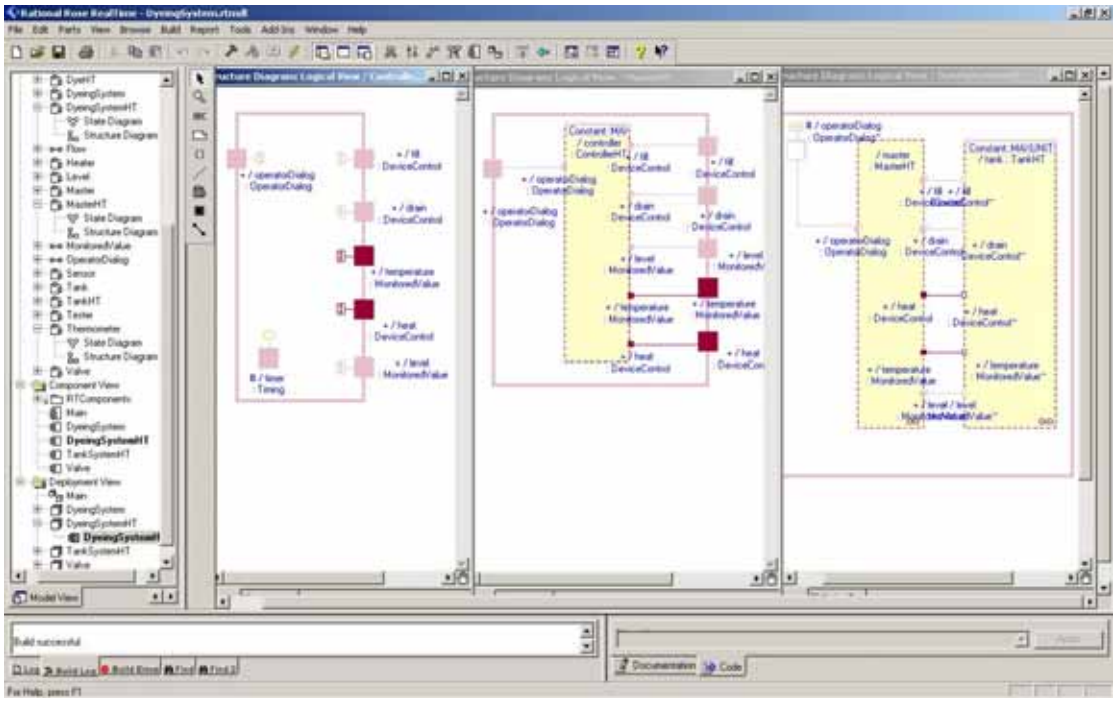

Figura 18 – Diagramas de estrutura.

2.6. Laboratório 6 – *Complete High-Temperatura System*

Neste sexto laboratório, foi assimilado o gerenciamento de complexidade de comportamento através da construção de máquinas de estados finais (FSMs) hierárquicas. Foi modificado o comporamento de *Controller* lpara que se torna-se uma máquina de estado hierárquica. Depois, foi adicionado um comportamento termostático para *ControllerHT* para que possa regular apropriadament a temperatura do líquido de tingimento no tanque.

Os objetivos deste laboratório foram:

- Criar uma FSM hierárquica em um modelo existente;
- Distribuir comportamento através da FSM hierárquica do modelo.
- Compilar, executar, e debugar o modelo contendo uma FSM hierárquica.

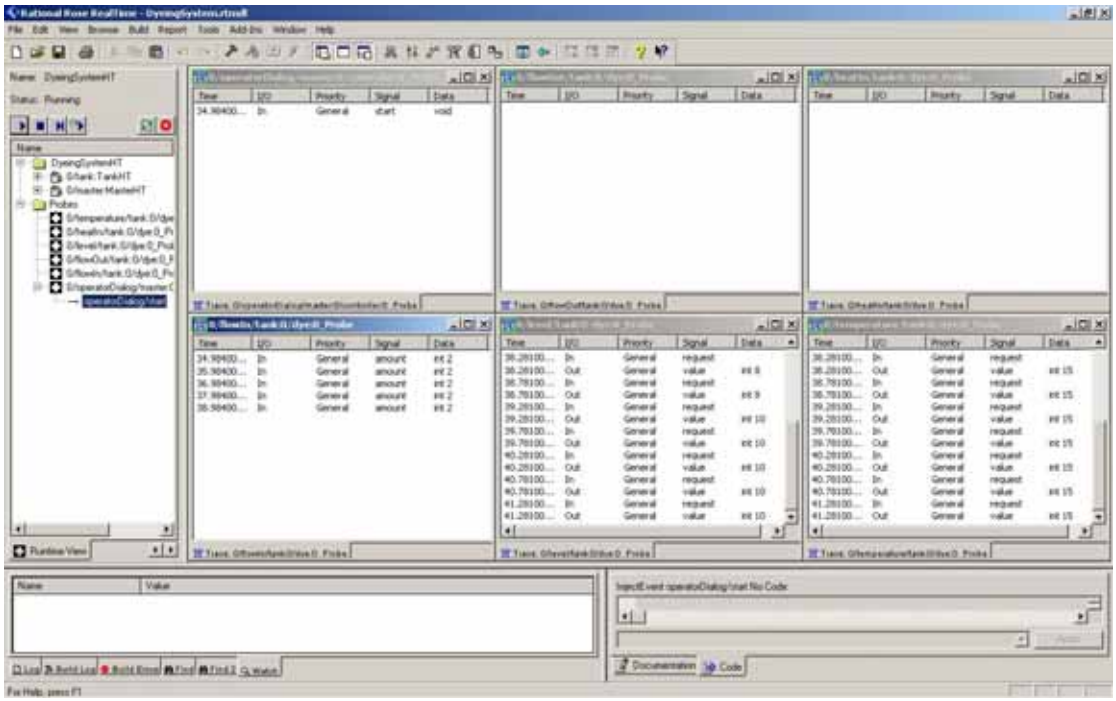

Figura 19 – *Traces.*

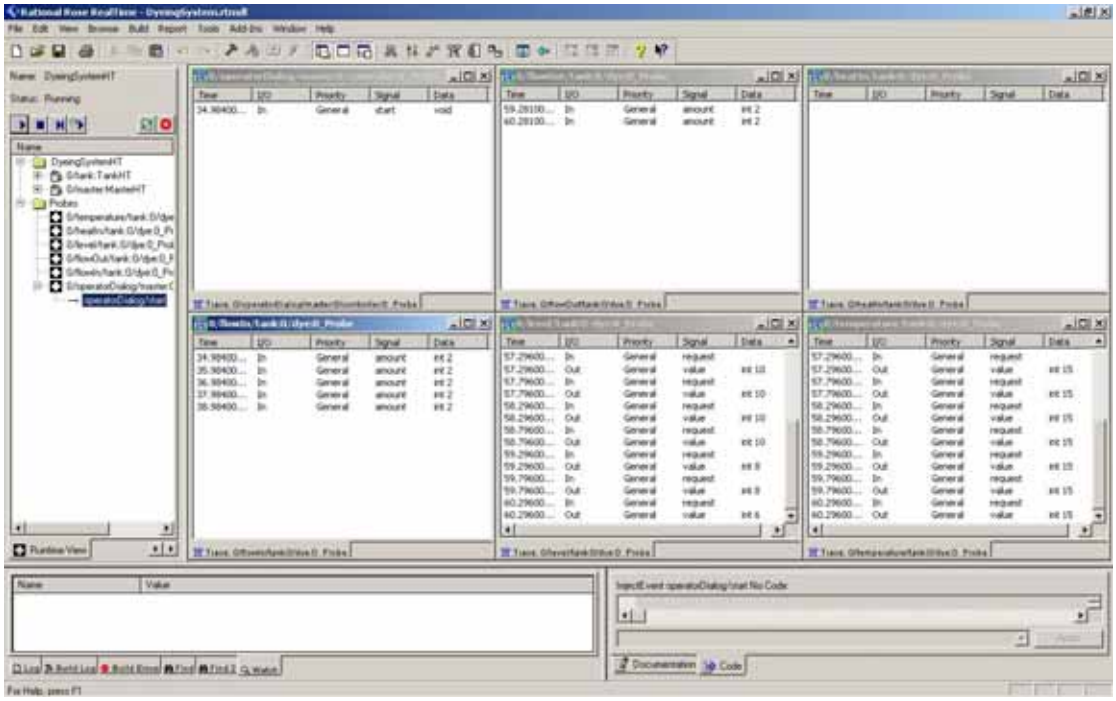

Figura 20 – *Traces.*

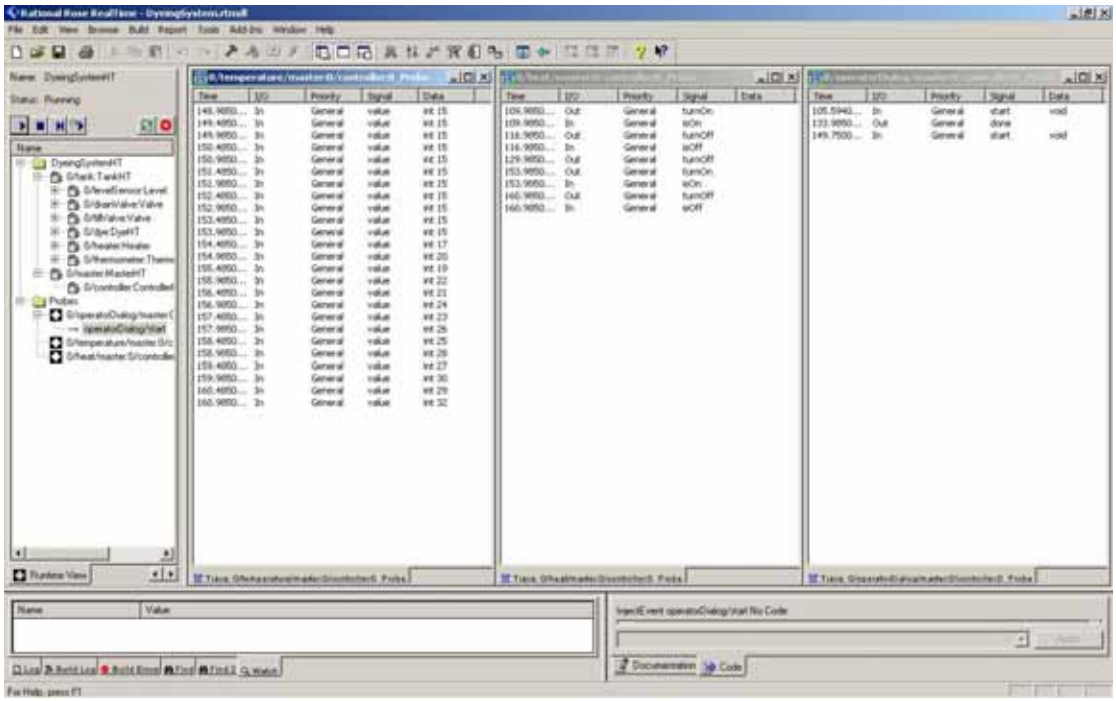

Figura 21 – *Traces* verificando o comportamento do modelo.

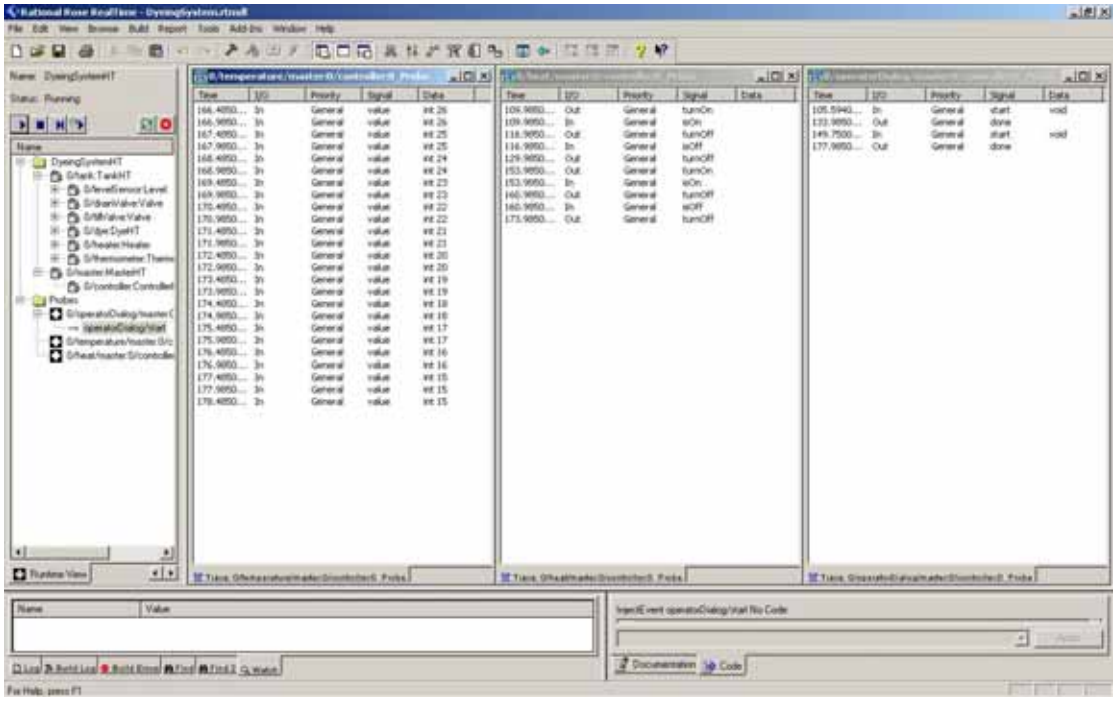

Figura 22 – *Traces* do modelo final.

2.7. Laboratório 7 – *Deliverable and Test Harness / Low and High Temperature*

Neste sétimo laboratório, foi assimilado o gerenciamento da complexidade do modelo através da utilização de *packages*. A aplicação *DyeingSystem* foi organizada

em duas categorias, *Deliverable* e *Test Harness*, e em *Low Temperature* e *High Temperature*.

Os objetivos deste laboratório foram:

- Criar *packages* e alocar elementos existentes do modelos naquelas;
- Criar uma *containing capsule*;
- Adicionar estrutura simples para a *capsule* através da adição de *capsule roles*;
- Compilar, executar e debugar o modelo.

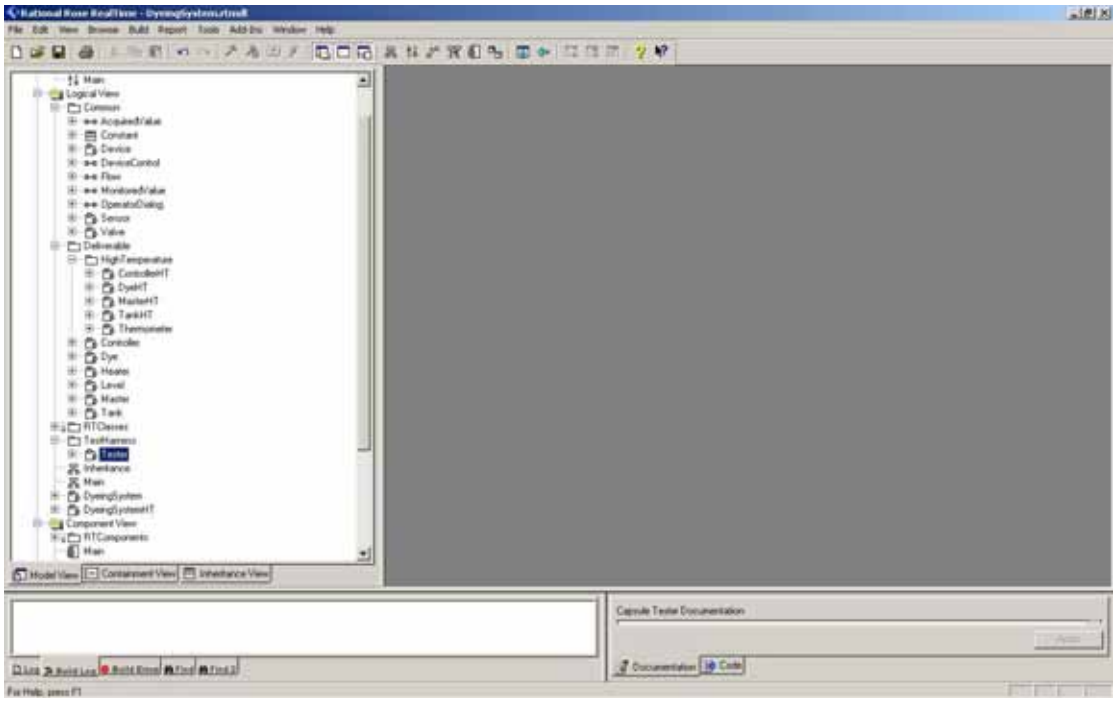

Figura 23 – *Packages*.

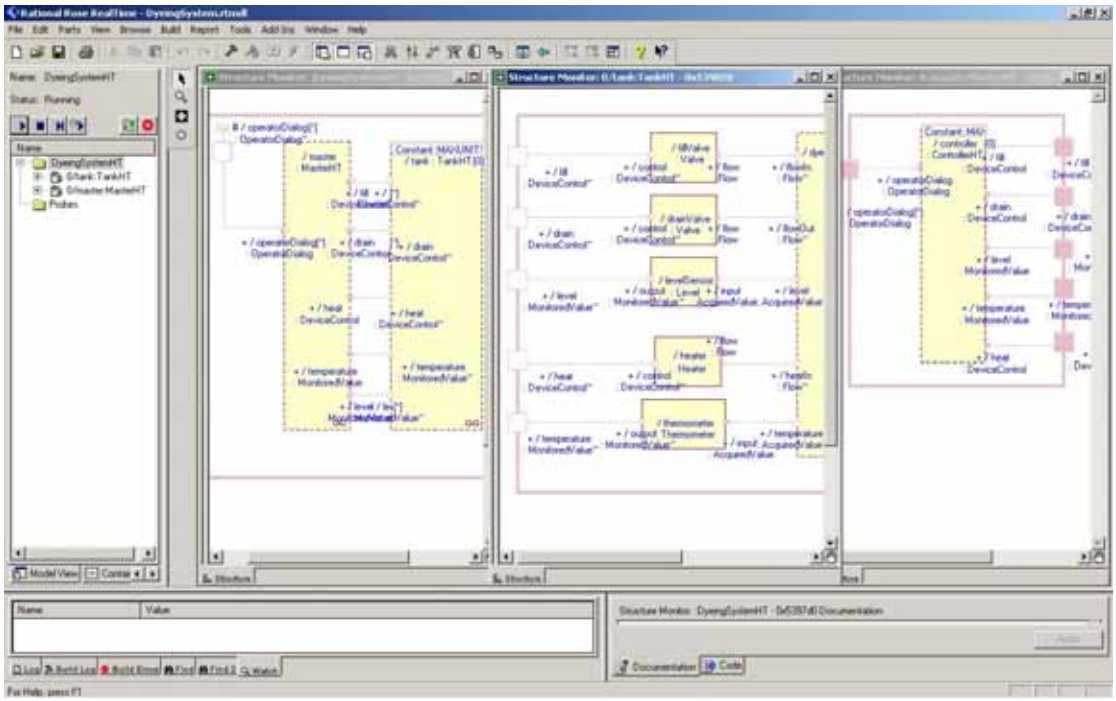

Figura 24 – *Structure Monitors*.

3. Conclusão

A realização destes laboratórios foi importante para a melhor assimilação das principais funcionalidades do RRRT, as quais serão praticadas na integração do projeto, desde a implementação da CSC até integração final do projeto VANT-EC-SAME.

Após a realização dos laboratórios, o alunos já está apto a implementar seu Componente de Software de Computador – CSC e poderá, assim, partir para o próximo nível de integração.

Dessa forma, ficou evidente a utilidade de se realizar laboratórios a fim de assimilar e praticar as principais funcionalidades do software *Rational Rose Real Time – RRRT*.