A Utilização de Múltiplos Métodos Formais na Análise de Requisitos de Software

Simone C. dos Santos (scs@di.ufpe.br)
Fabio Q. B. da Silva (fabio@di.ufpe.br)
Departamento de Informática
Universidade Federal de Pernambuco

Sumário

Este artigo propõe a utilização de múltiplos formalismos na análise de requisitos de software para especificar aspectos diferentes de um sistema complexo, explorando o poder de expressão de cada método e considerando a necessidade de relacionamento entre as especificações. Para isto, um sistema de caldeiras (*Boiler System*) é utilizado como estudo de caso, para o qual duas especificações formais são desenvolvidas, constituindo dois modelos diferentes do sistema: consistência de medidas dos dispositivos (descrito em TLA) e transições de estados considerando falhas na comunicação entre subsistemas controladores (descrito em Z). Uma relação matemática rigorosa entre TLA e Z é estabelecida, permitindo a integração entre as duas especificações através de provas de consistência. Os resultados descritos aqui fazem parte do trabalho desenvolvido em [3].

Abstract

This article proposes the use of multiple formalisms in the software requirements analysis to describe distinct aspects of a complex system, exploring the expressive power of each method and considering the relationship between the specifications. For this purpose, a boiler system is used as a case study, in which two formal specifications are developed, building two different models of the system: consistency of the device's measures (described using TLA) and state transitions taking into account communication failures between the controlling subsystems (described using Z). A rigorous mathematical relation is established between Z structures and TLA structures, allowing the integration of the specifications via formal proofs of consistency. The results reported here are part of the work developed in [3].

1 Introdução

A análise e especificação de requisitos representa o primeiro estágio do modelo clássico do ciclo de vida de software. O primeiro passo a ser tomado pelo engenheiro de software neste estágio é entender o problema que se deseja resolver possibilitando a verificação de que o sistema a ser construído atende aos objetivos estabelecidos.

Considerando que o engenheiro de software conhece pouco sobre o sistema, torna-se fundamental aumentar a precisão dos requisitos descritos nesta fase, especificando o que o sistema deve fazer e omitindo detalhes de como deve ser feito através de uma linguagem matemática precisa e não ambígua, isto é, formal. Ao especificar formalmente um sistema, o engenheiro de software é obrigado a entender melhor o problema, descrevendo o sistema de forma precisa e clara, com a possibilidade de provar matematicamente certas propriedades que devem ser satisfeitas pelo sistema.

Considerando o desenvolvimento de sistemas complexos, os requisitos iniciais podem ser considerados sob vários aspectos diferentes, por exemplo: comportamento determinístico, concorrente ou temporal. Desta forma, a especificação de requisitos pode ser dividida em subfases menores ao considerar cada um destes aspectos separadamente, reduzindo a quantidade de detalhes a serem considerados por vez e, com isso, proporcionando um maior entendimento e facilidade para explorar as características de cada um destes aspectos.

Entretanto, a capacidade de descrever todos estes aspectos geralmente não é encontrada em um único método formal. Embora algumas linguagens possam descrever mais de um aspecto, é difícil encontrar um formalismo capaz de tratar com a mesma facilidade todas as características de um sistema complexo. Por exemplo, a linguagem MaMOOZ [5] permite expressar considerações de tempo além de aspectos deterministas, mas não modela concorrência.

Desta forma, a abordagem proposta por este artigo utiliza formalismos diferentes para especificar cada aspecto do sistema, explorando o poder de expressão de cada método e considerando a necessidade de relacionamento entre as várias especificações.

Para que as especificações possam co-existir no mesmo projeto, elas devem ser consistentes entre si. Desta forma, é necessário provar matematicamente alguma relação que defina esta consistência. A integração de tais especificações é realizada através de provas de consistência que preservam as propriedades de segurança satisfeitas por cada uma delas separadamente. Desta forma, uma especificação consistente, mais completa e confiável é obtida.

Neste artigo, apresentaremos o enfoque discutido acima através de um estudo de caso, um sistema crítico de controle de caldeiras, denominado ao longo do texto por *Boiler System*. A linguagem TLA (*Temporal Logic of Actions* [6]) é utilizada para descrever as propriedades de segurança deste sistema. As características funcionais são especificadas utilizando a linguagem Z [10]. As justificativas e motivações para a escolha de tais métodos ficam claras no decorrer do texto.

Para tornar mais clara a compreensão dos conceitos, vários detalhes formais foram omitidos neste artigo. O leitor interessado nos detalhes técnicos e na abordagem mais teórica deste problema é convidado a ler [3].

O trabalho é organizado em 6 seções. A Seção 2 descreve rápida e informalmente o sistema de caldeiras que será usado como estudo de caso. As Seções 3 e 4 apresentam fragmentos da especificação formal do sistema descritos em TLA e Z. A Seção 5 mostra como provar a consistência entre as duas especificações e, por fim, a Seção 6 apresenta as conclusões e considerações finais.

2 Estudo de Caso: um Sistema de Caldeiras

Esta seção descreve um sistema de caldeiras utilizado como estudo de caso para abordagem proposta por este artigo, referenciado ao longo do texto por *Boiler System*. Aspectos importantes referentes

às propriedades de segurança e funcionalidade deste sistema serão especificadas posteriormente utilizando TLA e Z, respectivamente. A escolha deste exemplo é justificada pela inerente complexidade deste sistema considerado de segurança crítica, onde características como concorrência, transição de estados e propriedades temporais precisam ser consideradas e descritas.

A descrição de tal sistema foi proposta no Simpósio Internacional de Projeto e Sistemas de Software Relacionados a Segurança (Waterloo, Canadá, [8]), com a finalidade de promover uma competição entre os participantes através do enfoque de desenvolvimento formal aplicado a um problema genérico.

2.1 O *Boiler System*

O *Boiler System* é composto por um vasilhame aquecido, produzindo vapor d'água saturado através de um duto no topo do vasilhame e cujo fluxo de água em seu interior é gerado por quatro bombas. O sistema possui, ainda, instrumentos que realizam as medições do nível de água no vasilhame, da taxa de vapor expelida e do fluxo de água das bombas (veja Figura 1).

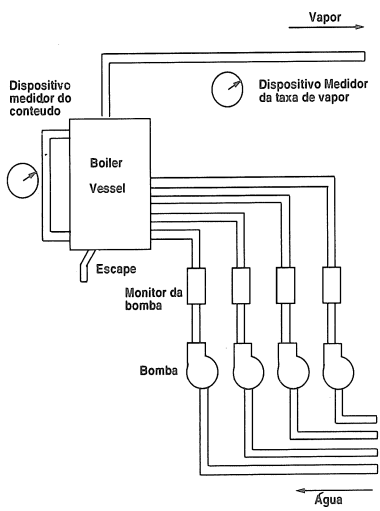

Figura 1: Boiler System

Um sistema de instrumentação (IS) realiza as medições de tais dispositivos de medida e um sistema de monitoração/controle (MCS) decide quando o sistema deve ser desligado, a partir de leituras de medidas fornecidas pelo sistema de instrumentação. Falhas podem ocorrer nos dispositivos de medida e na comunicação de mensagens entre os sistemas de instrumentação e monitoração/controle.

As mensagens que IS pode emitir a MCS são descritas completamente em [8]. Algumas delas são: *SYNC* (deverá aparecer em toda transmissão e é responsável pela sincronização entre IS e MCS), *START* (deve ser enviada por IS após a mensagem *READY* ter sido recebida, indicando que o sistema assumiu o modo **systest**), *PUMPINon/off* (informa se as bombas estão ligadas ou não), *WATERLEVEL* (informa o nível de água), entre outras. MCS também emite mensagens a IS, por exemplo, a mensagem *SYNC* (cujo parâmetro permite a verificação de mensagens atrasadas e duplicadas), *READY* (enviada quando o sistema está pronto para funcionar), *COMPTEST*, *SYSTEST*, *NORMAL*, *DEGRADED*, *EMERGENCY* e *SHUTDOWN* (informando o modo de operação

do sistema).

3 A Especificação em TLA

Utilizando uma estratégia de *comunicação de falhas*[1] e de *cálculo de nível*, as propriedades de segurança do *Boiler System* são descritas nesta seção usando a lógica TLA . Esta especificação foi desenvolvida em [1] e é utilizada aqui como ponto de partida para nossos estudos. Iniciamos com uma breve descrição de TLA.

3.1 Noções Básicas sobre TLA

A Lógica Temporal de Ações (TLA) é a combinação de duas lógicas: uma lógica de ações e uma lógica temporal simples.

Primeiramente, os conjuntos **Val** e **Var** são considerados. Assume-se que **Val** é um conjunto de valores que incluem boleanos **true** e **false**, números tais como 1, 2 e -3, sequência de caracteres como "abc" e conjuntos como o conjunto **Bool** de boleanos e **Nat** de naturais. **Var** representa o conjunto infinito de nomes de variáveis, tais como x e *boiler*.

Fórmulas atômicas em TLA são *predicados* e *ações*. Um predicado pode ser visto como uma função de *estados* em valores *booleanos*, onde um estado é uma atribuição de valores a variáveis. Uma ação é uma expressão booleana formada de variáveis, variáveis decoradas e valores. Uma ação representa uma relação entre o estado anterior (representado pelas variáveis não decoradas) e o estado posterior (representado pelas variáveis decoradas). Assim, $x' = x + 1$ é válida entre os dois estados se o valor de x no estado posterior é uma unidade maior que o valor de x no estado anterior.

A sintaxe de TLA é:

$$\mathrm{F} ::= \mathrm{P} \mid \Box A \mid \neg\ \mathrm{F} \mid F_1 \wedge F_2 \mid \Box\ \mathrm{F}$$

onde P denota predicados e A ações.

Sistemas são sempre representados em TLA por fórmulas do tipo $\mathrm{P} \wedge \Box A$, onde P representa uma condição inicial. Para expressar que uma propriedade F é satisfeita por um sistema *Sys*, escreve-se $Sys \Rightarrow F$.

3.2 Modelagem do Sistema em TLA

A especificação em TLA do *Boiler System* realizada por Bruns e Anderson em [1] usa consistência de dados para detectar falhas. O modelo do *Boiler System* em TLA captura os componentes físicos e o sistema de monitoração. O sistema de monitoração lê informações dos dispositivos e decide quando o sistema entra em **shutdown**. As bombas são ligadas pelo sistema de controle (que não faz parte do modelo). A comunicação entre os sistemas de instrumentação (responsável pelo envio de dados físicos), de monitoração e de controle, não é modelada.

As variáveis do Sistema L, L_c, *up* e *Safe* representam, respectivamente, o nível real do conteúdo do vasilhame, o valor calculado do nível, a variável que indica se o sistema está operante ou desativado e o limite de segurança do nível de água no vasilhame.

O Sistema do Boiler é descrito na forma padrão de uma especificação em TLA:

$$BSys_{TLA} \stackrel{\text{def}}{=} Init \wedge \Box Step$$

[1] Note que, *falhas* (do inglês, "*failure*"), neste contexto, dizem respeito a possíveis defeitos de dispositivos que, consequentemente, podem gerar *falhas* ("*fault*"), isto é, erros no sistema.

onde a condição inicial *Init* estabelece que, inicialmente, conhecemos o nível real do boiler que deve estar dentro dos limites de segurança e *Step* representa os componentes do Sistema.

$$Init \stackrel{\text{def}}{=} L_c = L \wedge up \wedge L \subseteq Safe$$

O modelo do *Boiler System* possui cinco componentes: o comportamento do boiler, o comportamento em caso de *shutdown*, fusão de dados, consistência de dados e determinação do nível de água.

$$Step \stackrel{\text{def}}{=} Boiler \wedge Shutdown \wedge Fusion \wedge Cons \wedge Level$$

Aqui, apenas mostraremos a modelagem, através da especificação em TLA, do componente *Shutdown*. O modelo *Shutdown* determina o valor da variável *up* que assume **true** quando o nível do boiler estiver dentro dos limites de segurança:

$$Shutdown \triangleq up = L_c \subseteq Safe$$

Em [1], algumas propriedades matemáticas do sistema foram estabelecidas, indicando que o sistema é seguro de acordo com as hipóteses da modelagem. Dentre tais propriedades de segurança, a mais importante mostra que, enquanto a variável *up* tem o valor **true**, o nível de água do boiler permanece dentro de seu limite de segurança.
Assim, o seguinte teorema foi provado em [1]:

Teorema 1: $BSys_{TLA} \Rightarrow Safety$

onde a noção de segurança é formalizada por

$$Safety \stackrel{\text{def}}{=} \Box(up \Rightarrow L \subseteq Safe)$$

Alguns aspectos importantes não são considerados na especificação em TLA. Por exemplo, inconsistências além daquelas geradas por falhas dos dispositivos como erros de comunicação (falhas de protocolo entre subsistemas) ou em cálculos de variáveis (erros algébricos do programa) não foram consideradas. Além disso, o sistema pode assumir estados intermediários a medida que alguns dispositivos falham individualmente. Tais aspectos são considerados na especificação em Z descrita na próxima seção.

4 Transição de Estados

Analisando os pontos que não foram considerados pela especificação anterior em TLA, podemos imaginar o sistema modelado como uma sequência de transição de estados gerados por situações que comprometam a estabilidade do mesmo. Neste modelo, o tratamento dado às situações de falha varia de estado para estado. Assim, caso o sistema esteja funcionando *normalmente* e o dispositivo medidor do nível de água apresente alguma falha, o sistema assumirá o estado de *emergência*. Entretanto, caso esta falha aconteça quando o sistema estiver com outro dispositivo de medida (do vapor ou das bombas) com defeito, o funcionamento do sistema passa a ser inviável e, portanto, o sistema deve ser *desativado*. A Figura 2 mostra as possíveis transições do *Boiler System*.

Outro importante aspecto é que as transições de estado devem considerar que existem atrasos, perdas ou corrupção de mensagens no canal de comunicação entre os sistemas de instrumentação e monitoração/controle. Na verdade, o canal de comunicação entre tais sistemas apresenta um atraso inerente de transmissão referente a própria velocidade de transmissão e processamento. Desta forma, o sistema de monitoração/controle precisa estabelecer uma estimativa sobre este valor que considere a diferença entre a informação que chega e a informação real (veja Figura 2).

Quanto aos possíveis atrasos, existem três possibilidades:

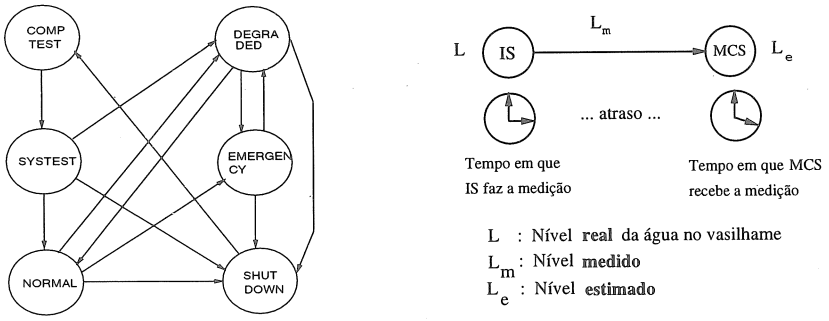

Figura 2: Transição de estados e comunicação entre subsistemas

1. *não existe falha no link*: o atraso é somente devido a velocidade de linha e processamento.
2. *mensagem chega com atraso ou é perdida*: neste caso MCS nao recebe nenhum valor e deve continuar estimando o crescimento/decrescimento através dos últimos valores recebidos.
3. *mensagem é corrompida*: neste caso MCS descarta tal mensagem e continua estimando o valor real.

4.1 A Especificação em Z do *Boiler System*

Nesta subseção, apenas alguns trechos da especificação em Z são apresentados, com o objetivo de fornecer uma idéia geral de como os aspectos dicutidos acima foram descritos. Para uma verificação completa desta especificação veja [3].

Z é uma notação baseada em lógica de predicados e teoria de conjuntos para modelagem rigorosa e construtiva de sistemas. Uma especificação em Z é formada por vários tipos de declarações para introduzir nomes, expressões que denotam termos, predicados que expressam propriedades dos valores dos nomes declarados e esquemas que impõem uma estrutura sobre as declarações e predicados. Assumimos que o leitor seja familiarizado com esta notação, descrita detalhadamente em [10].

A especificação em Z descreve o comportamento do *Boiler System* baseada na transição de seis estados representados pelo tipo enumerado *STATE*.

Por convenção, neste trabalho, um esquema é referenciado em letras estilo *itálico* e inicia-se com letra maiúscula. As variáveis são escritas em letras minúsculas salvo aquelas cujos nomes são formados por uma única letra. Para evitar dúvidas quando se fala de uma transição de estados (representada por um esquema) e um estado específico, este último será escrito em **negrito**.

As constantes *lim-sup* e *lim-inf* representam os limites físicos do nível de água no vasilhame que não poderá ultrapassar seu limite superior nem obter um nível menor que seu limite inferior. S_p e P_p representam os limites físicos da taxa de vapor máxima e fluxo total de água bombeada, respectivamente. A constante K é o fluxo de água por bomba e np, o número total de bombas no sistema (ligadas ou não). As constantes v_l, v_s e v_p são introduzidas para representar as taxas de variação mínimas das medidas do nível, vapor e bomba, respectivamente, referentes a atrasos inerentes ao canal de comunicação. Também são introduzidas as constantes i_n, i_d, i_e e i_s, que indicam os índices de incerteza de medidas provenientes dos estados **normal**, **degraded**, **emergency** e **shutdown**, respectivamente. A contante *period*, não considerada na especificação anterior, indica o intervalo de

tempo referente a cada transmissão (igual a 5 segundos, de acordo com a especificação informal em [8]).

Algumas das mensagens trocadas pelo sistema de Intrumentação e Monitoração/Controle são modeladas pelo tipo enumerado *MESSAGE*. Outras mensagens como *WATERLEVEL*, *STMRATE* e *WATFLOW*, emitidas pelo sistema de intrumentação, são representadas pelas variáveis de entrada $L_m?$, $S_m?$ e $P_m?$, respectivamente. As mensagens *COMPTEST*, *SYSTEST*, *NORMALOPS*, *DEGRADED*, *EMERGENCY* e *SHUTDOWN*, emitidas pelo sistema de monitoração/controle e que informam o estado do *Boiler System*, foram omitidas deste modelo, pois o estado do sistema é representado pela variável *state* que assume um dos valores de *STATE*.

$STATE ::= Comptest \mid Systest \mid Normal \mid Degraded \mid Emergency \mid Shutdown$
$MESSAGE ::= Ready \mid Watlevok \mid Stmratok \mid Pumpok \mid Watlevus \mid Stmratus \mid Pumpus$
$SITUATION := OPEN \mid CLOSED \qquad INDICATOR := ON \mid OFF$

$| \; Safe : \; (\mathbb{R} \times \mathbb{R}) \quad | \; K : \; \mathbb{R} \quad | \; S_p, P_p : \; \mathbb{R} \quad | \; v_l, v_s, v_p : \; \mathbb{R}$

$$\begin{array}{|l} i_n, i_d, i_e, i_s : \; \mathbb{R} \\ \hline i_n < i_d < i_e < i_s \\ i_n > 0 \end{array} \qquad \begin{array}{|l} np : \; \mathbb{N}^+ \\ \hline np = 4 \end{array} \qquad \begin{array}{|l} period : \; \mathbb{N}^+ \\ \hline period = 5 \end{array}$$

O estado do *Boiler System* é representado pelo esquema *BSys-Real* onde L, S, e P são respectivamente os valores reais do nível de água no vasilhame, da taxa de vapor e do fluxo de água bombeada. Assume-se que estes valores são constantemente alterados durante o funcionamento do sistema, embora as operações responsáveis por estas alterações não estejam representadas na especificação em Z.

L_c, S_c e P_c representam os valores do nível de água, taxa de vapor e fluxo de bombas calculados pelo sistema de instrumentação. O cálculo de medidas é considerado correto e por isso é definido em termos de valores reais do *Boiler System*. As variáveis L_e, S_e e P_e correspondem aos respectivos valores estimados calculados pelo sistema de monitoração/controle. L_{old}, S_{old} e P_{old} representam os últimos valores medidos do nível, fluxo e vapor, respectivamente, uma que vez pode haver perda de mensagens. O componente de estado *syncron* funciona como um contador de mensagens e através deste componente é estabelecida a sincronização entre os processos comunicantes.

Os componentes de estado *pumps*, *valve* e *up* indicam se as bombas estão ligadas, o escape está aberto e se o sistema está em funcionamento, respectivamente.

BSys-Real

$$\begin{array}{|l} L, S, P, L_c, S_c, P_c : \mathbb{R} \\ L_{old}, S_{old}, P_{old}, L_e, S_e, P_e : \mathbb{R} \\ pumps : Seq \; INDICATOR \\ syncron : \mathbb{N}^+ \\ valve : SITUATION \\ up : INDICATOR \\ state : STATE \\ \hline \sharp(pumps) = np \wedge (S_c \leq S_p) \wedge (P \leq P_p) \wedge (S_c \leq S_p) \wedge (L_c = L + (P - S)) \wedge (P_c = K * np) \end{array}$$

A função *FE* é definida para auxiliar no cálculo do intervalo estimado de uma medida recebida por MCS que sempre utiliza uma taxa de variação proveniente de atraso na comunicação. *plt* calcula o período decorrente desde a última transmissão.

$$\begin{array}{|l} FE : \; (\mathbb{R} \times \mathbb{R}) \rightarrow (\mathbb{R} \times \mathbb{R}) \\ \hline FE = \lambda \, m : \mathbb{R}, \; v : \mathbb{R}.(m - v, m + v) \end{array} \qquad \begin{array}{|l} plt : \; \mathbb{N}^+ \rightarrow \mathbb{N}^+ \\ \hline plt = \lambda \, s : \mathbb{N}^+.(s - syncron * period) \end{array}$$

$$- \subseteq_I - : \quad (\mathbb{R} \times \mathbb{R}) \leftrightarrow (\mathbb{R} \times \mathbb{R})$$
$$(a, b) \subseteq_I (c, d) = (a < b) \land (a \geq c) \land (b \leq d)$$

O esquema *Estimate-Level* especifica o controle do cálculo estimado do nível da água baseado nas informações de medidas de dispositivos, taxas de variação e verificações sobre as transmissões. Considerando tais transmissões, quatro casos são possíveis:

1. a mensagem esperada em um intervalo de transmissão é recebida ($syncron = sync?$ e $transmit? =$ **true**) com atrasos inerentes a velocidade de linha e processamento e, portanto, as taxas de variação v_l, v_s e v_p e o estado em que o sistema se encontra são considerados no cálculo do valor estimado;

2. no lugar da mensagem esperada, no mesmo período de transmissão é recebida uma mensagem antiga ($sync? < syncron$) que é descartada e, portanto, o cálculo estimado é baseado em vários aspectos: última medida recebida, o tempo de espera equivalente a um período de transmissão (considerando que a mensagem anterior foi devidamente recebida), atraso decorrente do canal de comunicação e o estado atual do sistema;

3. uma mensagem posterior a mensagem esperada é recebida ($sync > syncron$) e, portanto, além do atraso decorrente do canal e o estado atual do sistema, considera-se o tempo de espera no cálculo estimado da medida;

4. nenhuma mensagem chega e, portanto, o cálculo estimado é idêntico ao item 2.

Estimate-Level

$$L_m?, i! : \mathbb{R}$$
$$sync? : \mathbb{N}^*$$
$$transmit? : \mathbb{B}$$
$$transmit? \Rightarrow$$
$$((syncron = sync? \land L_e = FE(L_m?, v_l + i!) \land L'_{old} = L_e \land syncron' = syncron + 1) \lor$$
$$(syncron < sync? \land L_e = FE(L_{old}, plt(sync?) * (v_l + i!)) \land syncron' = sync? + 1) \lor$$
$$(syncron > sync? \land L_e = FE(L_{old}, period * (v_l + i!)) \land syncron' = syncron))$$
$$\neg transmit? \Rightarrow (L_e = FE(L_{old}, period * (v_l + i!)) \land syncron' = syncron)$$

O esquema *RLimit-Ok* verifica se os limites para o nível de água estão sendo respeitados pelo nível estimado. O esquema *BSys-Real-Off* representa o estado em que o sistema não está em funcionamento, ou seja, sistema desligado, bombas desligadas e o escape aberto.

RLimit-Ok

$$L_c : \mathbb{R}$$
$$L_c \subseteq_I Safe$$

BSys-Real-Off

$$\Delta Bsys$$
$$ran\ pumps' = \{OFF\} \land valve' = OPEN \land up' = OFF$$

O estado inicial do sistema é representado pelo esquema *Comptest*, onde o sistema está desligado e os valores zerados.

Comptest

$$Bsys\text{-}Off$$
$$msg! : MESSAGE$$
$$L'_c = 0 \land L'_e = 0 \land L'_{old} = 0 \land S'_c = 0 \land S'_e = 0 \land S'_{old} = 0 \land P'_c = 0 \land P'_e = 0 \land P'_{old} = 0 \land$$
$$syncron = 0 \land state' = Comptest \land msg! = Ready$$

Para ilustrar como as transições da Figura 2 são especificadas, mostramos a seguir a modelagem da transição do estado **normal** para o estado **shutdown**, caso em que os limites físicos do nível de água são ultrapassados.

Normal-To-Shutdown

$$\begin{array}{|l} BSys\text{-}Real\text{-}Off \\ Estimate\text{-}Level \\ \neg RLimit\text{-}Ok \\ \hline i! = i_s \wedge state = Normal \wedge state' = Shutdown \\ \hline \end{array}$$

Em [3], também foram provadas algumas propriedades importantes sobre esta especificação como determinismo e ausência de *deadlock* sobre as transições de estado. Entretanto, é preciso mostrar que a especificação descrita em TLA e a especificação em Z não são *conflitantes*, ou seja, que não existe contradição lógica entre as duas descrições com relação às provas de consistência realizadas em cada uma delas separadamente. Isto será desenvolvido na próxima seção.

5 Integração entre as Especificações do *Boiler System*

Para se provar propriedades de ou estabelecer relações entre as especificações envolvendo Z e TLA é necessário definir uma relação matemática rigorosa entre os formalismos.

Um possível enfoque é fazer uma tradução da especificação em Z para uma especificação em TLA que possua o "mesmo significado". O problema deste enfoque é definir precisamente o que quer dizer "mesmo significado" quando considera-se formalismos diferentes. Uma solução seria estabelecer uma relação matemática entre as semânticas de Z e TLA, que permitisse mostrar equivalência de significados das especificações nestes formalismos. Entretanto, este problema, embora relevante, é extremamente complexo e fora do escopo deste trabalho.

O que se propõe aqui é produzir uma tradução sistemática de estruturas da linguagem Z para estruturas de TLA. A corretude semântica desta tradução é realizada intuitivamente com base na preservação de significados entre os dois formalismos, de forma semelhante ao enfoque utilizado para a tradução da linguagem MooZ para Eiffel em [7].

5.1 Tradução de Z para TLA

Para realizar a tradução de Z para TLA, uma função de tradução sintática é definida, denominada por τ, que leva um subconjunto de frases Z em TLA. Para uma melhor visualização de como uma especificação em Z é traduzida para TLA, a função τ é definida em duas etapas: tradução de tipos de dados e tradução de estruturas sintáticas.

A tradução dos tipos de Z para TLA é realizada considerando cada tipo em Z um subconjunto do conjunto **Val** de valores.

Uma especificação em Z é composta por uma seqüência linear de parágrafos que podem ser um dos seguintes objetos sintáticos: esquema, descrição axiomática, restrição, abreviação, tipo básico e tipo algébrico. Desta forma, uma especificação em Z é descrita como segue:

Especificação	::=	Parágrafo ... Parágrafo	
Parágrafo	::=	Esquema	Esquema
	|	Nome-Esquema $\triangleq$ Esquema-Exp	Esquema horizontal
	|	Descr-Axiom	Descr. axiomática

Predicado	Restrição
Ident == Expressão	Abreviação
[Ident,...,Ident]	Tipo básico
Ident ::= Ident |...| Ident	Tipo algébrico

Em [3], a função de tradução τ é definida para cada um dos parágrafos a partir de uma sintaxe simplificada, contendo unicamente as estruturas de Z consideradas no trabalho citado. Mostramos a seguir a definição de τ para algumas destas estruturas sintáticas.
Por exemplo, a função de tradução τ para um esquema é definida por:

τ *(Esquema)* $= Nome\text{-}Esquema \triangleq \tau$ *(Esquema-Def)*
τ *(Esquema-Def)* $= \tau$*(Declaração)* $\wedge \tau$*(Predicado)*

Na tradução de declarações e predicados, considera-se que os símbolos lógicos e relacionais de Z e TLA tem o mesmo nome. Assim, um predicado em Z do tipo $A \wedge B$ seria traduzido para $\tau(A) \wedge \tau(B)$ onde o símbolo $\wedge$ da tradução é a conjunção em TLA. Desta forma, temos por exemplo:

τ *($\neg$ Predicado)* $= \neg \tau$ *(Predicado)*
τ *(Predicado* $\wedge$ *Predicado)* $= \tau$ *(Predicado)* $\wedge \tau$ *(Predicado)*
...

Seguindo o mesmo raciocínio, a função de tradução τ é definida para as outras estruturas de Z, bem como cálculo de esquemas.

5.2 Consistência entre as Especificações do *Boiler System*

Considerando a especificação em TLA do *Boiler System*, o Teorema 1 definido na Seção 3.2 mostra que o sistema satisfaz uma importante propriedade de segurança referente ao nível de água do *Boiler System*:

$$BSys_{TLA} \Rightarrow Safety$$

Tal propriedade também deve ser preservada pela especificação em Z, quando as duas especificações do *Boiler System* são utilizadas conjuntamente. Para isso, a especificação em Z é traduzida para TLA utilizando a função de tradução τ comentada na seção anterior. Apenas parte da tradução é apresentada a seguir.

Desta forma, o modelo descrito em Z das transições de estado considerando falhas na comunicação é representado em TLA por

$$BSys_{\text{Z-TLA}}\text{-}Real \triangleq Init_{\text{Z-TLA}}\text{-}Real \wedge \Box Predicates_{\text{Z-TLA}}\text{-}Real$$

onde, $Init_{\text{Z-TLA}}$-*Real* representa as condições iniciais do sistema (em Z representadas pelo esquema que especifica o estado inicial *Comptest*) e as traduções de todas as partes de declaração (de esquemas e descrições axiomáticas), $Predicates_{\text{Z-TLA}}$-*Real* representa todos os predicados dos esquemas que modelam as transições de estado da Figura 2, incluindo o predicado invariante, definido no esquema que especifica o estado do sistema, e a parte de predicados definidas nas descrições axiomáticas. Para facilitar a leitura das fórmulas, $Init_{\text{Z-TLA}}$-*Real* e $Predicates_{\text{Z-TLA}}$-*Real* são subdivididas em fórmulas menores.
A fórmula $Predicates_{\text{Z-TLA}}$-*Real* é definida como segue

$Predicates_{\text{Z-TLA}}\text{-}Real \stackrel{\text{def}}{=} (np = 4) \wedge (period = 5) \wedge ... \wedge Inv_{Real} \wedge Transitions_{Real}$

$Transitions_{Real} \stackrel{\text{def}}{=} Comptest\text{-}To\text{-}Systest \vee Systest\text{-}To\text{-}Normal \vee ... \vee Shutdown\text{-}To\text{-}Systest$

onde, por exemplo, considerando o esquema *RLimit-Ok* definido por

$RLimit\text{-}Ok \stackrel{\text{def}}{=} L_c \subseteq Safe$

tem-se a transição *Normal-To-Shutdown* traduzida para a fórmula a seguir

$$\begin{array}{lll} Normal\text{-}To\text{-}Shutdown & \stackrel{\text{def}}{=} & \neg\ RLimit\text{-}Ok \\ & \wedge & BSys\text{-}Off \\ & \wedge & i! = i_s \\ & \wedge & state = Normal \\ & \wedge & state' = Shutdown \end{array}$$

Uma vez traduzida a descrição do *Boiler System BSys-Real* para a fórmula em TLA $BSys_{\text{Z-TLA}}$-*Real* é necessário estabelecer que $BSys_{\text{Z-TLA}}$-*Real* satisfaz as condições de segurança da especificação em TLA. Desta forma, seria preciso provar que:

$$BSys_{\text{Z-TLA}}\text{-}Real \Rightarrow Safety \qquad (1)$$

No entanto, isto não pode ser realizado diretamente pois as variáveis em $BSys_{\text{Z-TLA}}$-*Real* e *Safety* podem não ser as mesmas, terem tipos diferentes ou simplesmente denotar objetos diferentes apesar de terem o mesmo nome. Em particular, a variável *up* existe nas duas fórmulas mas com tipos diferentes:

- *up* em $BSys_{\text{Z-TLA}}$-*Real*, obtido através da função de tradução τ a partir da variável da especificação em Z, assume os valores em $\{ON, OFF\}$,
- *up* em *Safety*, é o *up* da especificação em TLA, que é uma variável boleana, ou seja, assume valores em $\{\mathbf{true}, \mathbf{false}\}$.

É claro que existe uma bijeção entre os dois conjuntos de valores para *up*. Desta forma, pode-se facilmente obter uma fórmula em TLA equivalente a (1), que define o teorema a seguir:

Teorema 2: $BSys_{\text{Z-TLA}}\text{-}Real \Rightarrow \Box(up = ON \Rightarrow Lim\text{-}Ok)$

Para facilitar a demonstração da prova do Teorema 2, o seguinte lema é definido

Lema 1:

$Transitions \Rightarrow (up = ON \Rightarrow Limit\text{-}Ok)$

Em [3], o Lema 1 é provado facilmente através do método de tableau [9]. Assim, utilizando o Lema 1 e a regra de inferência TLA STL4 descrita como segue

STL4. $\frac{F \Rightarrow G}{\Box F \Rightarrow \Box G}$

a prova do Teorema 2 é demonstrada a seguir:

Prova do Teorema 2:

$Transitions$	$\Rightarrow$	$(up = ON \Rightarrow Limit\text{-}Ok)$	pelo Lema 1
$\Box Transitions$	$\Rightarrow$	$\Box(up \Rightarrow Limit\text{-}Ok)$	pela regra STL4
$\Box Predicates \wedge Init_Z$	$\Rightarrow$	$\Box(up \Rightarrow Limit\text{-}Ok)$	lógica proposicional
$BSys_Z$	$\Rightarrow$	$\Box(up \Rightarrow Limit\text{-}Ok)$	pela definição de *BSys-Real*

□

Desta forma, podemos afirmar que a especificação em Z garante propriedades de segurança da especificação em TLA, e portanto, quando utilizadas em conjunto, as especificações não são contraditórias, como queríamos demonstrar.

6 Considerações Finais

O objetivo básico deste artigo é mostrar que a utilização de múltiplos formalismos permite especificar requisitos iniciais de sistemas complexos.

A partir de uma descrição informal de um sistema de caldeiras (*Boiler System*), descrevemos dois modelos distintos nos formalismos TLA e Z. Mostramos que a especificação em TLA, tomada isoladamente, deixa de considerar aspectos importantes do sistema como transição de estados e comunicação entre subsistemas controladores. Com a especificação em Z, aumentamos o conhecimento sobre o sistema e, consequentemente, a precisão dos requisitos nesta fase de desenvolvimento. Desta forma, desenvolvemos uma especificação do *Boiler System* mais completa e, com as provas de consistência entre as duas especificações, mais confiável.

Portanto, comprovamos, teoricamente (visto que não chegamos a uma implementação do *Boiler System*), a viabilidade e adequação da utilização de múltiplos formalismos na análise de requisitos de sistemas.

Alguns aspectos como o custo de desenvolvimento formal e o uso de ferramentas de apoio não são discutidos aqui, embora reconhecemos a importância de tais considerações.

Referências

[1] Glenn Bruns and Stuart Anderson. Using Data Consistency Assumptions to Show System Safety, 1993. Laboratory for Foundations of Computer Science, Edinburgh.

[2] R. de Lemos, A. Saeed, and T. Anderson. A Train as a Case Study for the Requirements Analysis of Safety-Critical Systems. *Computer Journal*, 35(1):30–39, 1992.

[3] Simone C. dos Santos. Especificação de sistemas críticos: um enfoque baseado na utilização consistente de múltiplos métodos formais. Master's thesis, Universidade Federal de Pernambuco, November 1995.

[4] Simone C. dos Santos e Fabio Q. B. da Silva. Um Enfoque Multiformalismos para Especificação de Sistemas de Segurança Crítica. *IX Simpósio Brasileiro de Engenharia de Software*, 1995.

[5] Ismar N. Kaufman. A Aplicação de Especificações Formais no Projeto Lógico de Software. Master's thesis, Universidade Federal de Pernambuco, Recife - Brasil, 1992. Departamento de Informática.

[6] Leslie Lamport. The Temporal Logic of Actions. Technical report, Digital Systems Research Center, 1991.

[7] Virgínia Adélia de Oliveira Cordeiro. De MooZ para Eiffel - Uma Abordagem Rigorosa para o Desenvolvimento de Sistemas. Master's thesis, Universidade Federal de Pernambuco, Recife, Pernambuco, Brasil, 1994.

[8] Institute For Risk Research. Specification for a software program for a boiler water content monitor and control system. Waterloo, Canadá, 1992.

[9] R. M. Smullyan. *First-Order Logic.* Springer-Verlag, Berlim, 1968.

[10] J. M. Spivey. *The Z Notation - A Reference Manual.* Prentice Hall, 1992.