



UNIVERSIDADE DO ALGARVE

Engenharia da Programação

“Modelo de Transformação”

DOCENTE	DISCENTES	
Eng.^a Paula Ventura	17149	António dos Anjos
	17325	Gil Moreira

Modelo de Transformação

Introdução

O modelo de Transformação ou de Desenvolvimento Transformacional é um modelo que se baseia em métodos formais. Um método formal, em Engenharia da Programação, é um método de desenvolvimento de software através do qual se pode definir precisamente um sistema e desenvolver implementações garantidamente correctas em relação a esta definição.

Os métodos formais

Algumas das características dos Métodos Formais são:

- Especificação precisa do processo de desenvolvimento;
- Baseado em linguagens de especificação e programação com semântica bem definida;
- Processo de desenvolvimento incremental, que possibilita expressão em vários níveis de abstracção;
- Processo de desenvolvimento habitualmente baseado em regras de transformação.

O modelo Transformacional pode ser descrito como um modelo de desenvolvimento de software que parte de uma especificação formal de um problema e, através de sucessivas transformações, chega a um produto final. A especificação formal do problema, após a análise do mesmo, é um pré-requisito para este modelo de desenvolvimento. Torna-se por isso necessário fazer uma breve referência ao método de Especificação Formal e de seguida falaremos do Modelo Transformacional.

A especificação formal surgiu da necessidade de definir o problema numa linguagem precisa. Para uma especificação detalhada, o modo mais preciso é através de uma notação matemática. Existem várias notações, por exemplo, as baseadas em Modelos (e.g. Z model e VDM) e as Algébricas (e.g. Larch e OBJ).

Por meio de uma abordagem algébrica o sistema é descrito em termos de operações e relacionamentos, enquanto que na abordagem baseada em modelos, é construído um modelo do sistema usando entidades matemáticas como conjuntos e sequências.

De uma forma geral, as características do método de Especificação Formal são:

■ Descrição Matemática Precisa

- Não ambígua;
- Permite verificar propriedades (ex. consistência);
- Usualmente abstractas e concisas.

■ É a base para

- Documento contratual;
- Documentação do produto;
- De referência para etapas seguintes.

Como exemplo, vamos enumerar alguns dos passos para a definição formal de uma função através de pré e pós condições:

- 1) Estabelecer um intervalo de parâmetros de entrada aceites pela função;
- 2) Especificar qual o resultado garantido numa situação normal;
- 3) Determinar quais as alterações a que serão, ou não, sujeitos os parâmetros de entrada e especificá-las.
- 4) Combinar os resultados dos passos precedentes em Pré e Pós-Condições.

Prós e Contras da Especificação Formal

Prós

- Desenvolvimento de uma especificação formal favorece uma compreensão mais profunda dos requisitos, e reduz erros e omissões e provê uma base para um design elegante;
- Especificações formais são entidades matemáticas e podem ser analisadas matematicamente para provar que uma implementação está de acordo com a especificação.
- Permite que se construa ferramentas para o processamento automático das especificações formais;
- Podem ser utilizadas como guia para a fase de testes.

Contras:

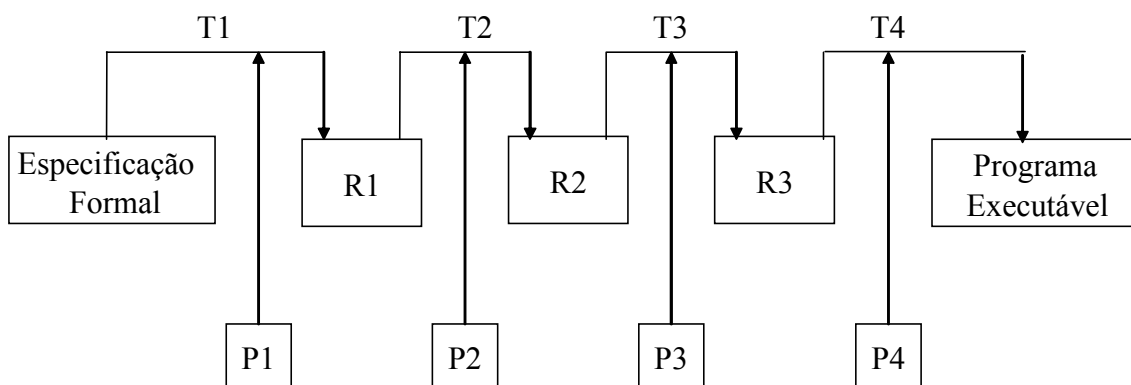
- Os gestores responsáveis pelo desenvolvimento são conservadores e por isso pouco abertos à adopção de técnicas onde as vantagens não são óbvias;
- Muitos engenheiros de software não foram treinados para o desenvolvimento através de especificação formal;
- Os clientes não estão familiarizados com técnicas de especificação formal;
- Algumas classes de software são difíceis de especificar formalmente com as técnicas existentes, particularmente os componentes interactivos e de processamento paralelo;
- Há um vasto desconhecimento da praticabilidade e aplicabilidade destas técnicas;
- Tem existido um ênfase maior no desenvolvimento de linguagens formais do que de métodos e ferramentas.

O Desenvolvimento Transformacional

Durante este tipo de desenvolvimento, a especificação formal é transformada através de uma série de passos onde cada transformação está suficientemente perto da descrição anterior, pelo que o esforço de verificação da transformação não é excessivo. Ou seja, é fácil provar que o programa cumpre os requisitos, desde que se prove que entre cada transformação eles se cumprem. Em sistemas de larga escala torna-se bastante difícil a realização de provas de verificação de correcção, no entanto, uma abordagem composta por uma sequência de passos mais pequenos, poderá ser mais eficaz.

Este processo não deixa de ser difícil, uma vez que, escolher que transformação a aplicar é uma tarefa complicada, e provar a correspondência entre as transformações não é algo trivial.

Pelos motivos atrás referidos, é improvável que uma abordagem puramente transformacional alguma vez seja utilizada em sistemas de larga escala, no entanto, a incorporação deste modelo noutros modelos, como p.ex. no modelo em espiral, dá-nos a oportunidade de melhorar o desenvolvimento de sistemas críticos tais como sistemas de controlo de aviões e sistemas médicos controlo.



Características gerais do Modelo Transformacional

- Permite a evolução do programa, coisa que p.ex. o modelo Cascata não permite;
- Fornece um método matemático para testar a correcção do programa;
- Dado P e uma Especificação= $\langle \text{Pré}, \text{Pós} \rangle, \dots$, pode-se verificar a validade de $\{\text{Pré}\} P \{\text{Pós}\}$;
- A abordagem transformacional é construtiva:
 - Dados os pares de Especificação= $\langle \text{Pré}, \text{Pós} \rangle$
 - Desenvolver o P que satisfaça todos os pares $\langle \text{Pré}, \text{Pós} \rangle$ da especificação;
 - Obter automaticamente o programa que satisfaça a especificação.
- Permite ao engenheiro de software a especificação, desenvolvimento e verificação do sistema de software pela aplicação de regras e notações matemáticas rígidas;
- Ambiguidades, incompletude e inconsistência podem ser descobertas e corrigidas facilmente através da análise matemática;
- Oferece a garantia de desenvolvimento de software sem defeitos;
- Porque o programador é obrigado a trabalhar dentro de limites bem definidos, evitam-se erros, e pode concentrar-se mais na parte que lhe compete.

Recentemente, técnicas desenvolvidas na Inteligência Artificial têm sido usadas para desenvolver um ambiente de suporte ao modelo transformacional. Estes ambientes, funcionam como assistentes automatizados, que mediam e auxiliam as actividades do Engenheiro de Software, tais como o registo dos passos intermédios do desenvolvimento, produção de informação necessária para a tomada de decisões e sugestão de estratégias.