1
Introdução

Nos últimos anos, a popularidade alcançada por Orientação a Objetos acabou por desvirtuar o significado do termo, a ponto de transformá-lo em lugar-comum. De um lado, empresas o utilizam como adjetivo dos mais diversos produtos; de outro, consumidores o percebem como sinal de superioridade e qualidade, mesmo quando seu uso traz mais dúvidas que certezas sobre a natureza do produto. Hoje, podemos encontrar – além das linguagens de programação tradicionais – bases de dados, interfaces de usuário, metodologias e até circuitos integrados (OOPIC) que se auto-intitulam orientados a objeto.

Este fenômeno é efeito colateral de uma pretensa revolução no modo como software é construído, na qual Orientação a Objetos seria a chave para um mundo novo, onde *reusabilidade* é a grande estrela. A revolução não veio, e a chamada "Crise do Software" (Cox, 1990) continua: o desenvolvimento de programas ainda é um processo arcano, e o reuso, a exceção da regra. Muitas vezes, exige um esforço tal que se justifica a "reinvenção da roda", sendo descartado o reuso em favor de uma implementação customizada.

Parte do problema advém do fato de que as linguagens de programação OO esperam alcançar o reuso através da *programação*, ao invés da *composição*. Nestas linguagens, o reuso é conseqüência da programação explícita de novos objetos, e não da composição funcional de objetos existentes (Nierstrasz, 1993). Por este motivo, somente uma mudança de foco – uma que privilegie *componentes de software* – pode ter esperanças de levar o reuso a níveis mais razoáveis. A idéia por trás de componentes de software é o estabelecimento de uma série de normas e padrões, de tal forma que a conformidade de um componente a um determinado padrão permita sua troca por outro, equivalente; ou ainda, que dois componentes possam ser facilmente conectados (Szyperski, 1997).

Porém, o paradigma da engenharia de software em vigor está mais preocupado com a especificação de *processos de desenvolvimento* que com a criação de padrões que possibilitem o nascimento de uma cultura voltada para

componentes. (Cox, 1990) descreve isso com lucidez, ao comparar o paradigma atual ao de um artesão, capaz de criar, uma a uma, partes que se encaixem perfeitamente para produzir a peça final. O artesão é um mestre conhecedor do processo de fabricação, mas é ignorante sobre padrões que permitam a fabricação de partes por terceiros, o que inviabiliza a produção de peças em larga escala.

Nesta dissertação, abordamos a construção de um programa de maneira idêntica ao problema da *síntese* de um circuito elétrico: dado um conjunto de entradas, como interligar componentes de forma a produzir o resultado esperado? A analogia com circuitos elétricos vai além da simples conexão de componentes, na medida em que tentamos incorporar vários de seus conceitos a este novo modelo de computação. O resultado é um modelo de componentes onde o reuso é favorecido, e onde características como concorrência e escalabilidade emergem de modo natural.