1

Introdução

A necessidade de se conhecer o identificador de um objeto para solicitar seus serviços foi apontada por Shaw, em [SG96], como uma restrição na construção de sistemas orientados a objetos. Apesar disso, poucas linguagens, padrões de projetos ou arquiteturas de *software* focam este problema. Na realidade, soluções propostas no nível de linguagens de programação envolvem relaxamento de conceitos de orientação a objetos como encapsulamento ou visibilidade. No nível de padrões de projetos ou arquiteturas de *software*, as soluções exigem que programadores escrevam códigos bastante complexos para permitir referências genéricas (como recomendadas nos padrões “Composite” e “Chain of Responsability” [G+95]).

Em [Car93], propõe-se a utilização de pronomes para solucionar este problema. Com estes, o programador consegue especificar o envio de uma mensagem de um objeto para outro(s) sem que aquele conheça a identificação deste(s). A idéia é que o envio de uma mensagem poderia ser especificado para um pronome e, em tempo de execução, esta seria encaminhada para o objeto, ou grupo de objetos, associados a este pronome.

Acredita-se ser a redução da distância entre projeto e implementação uma necessidade para a simplificação da implementação final de sistemas complexos [CCO98]. A crescente diversidade de requisitos não funcionais, como distribuição e concorrência, com que projetistas de *software* têm que conviver enfatiza a relevância desta redução. Apesar da variedade de ferramentas CASE que suportam geração automática de código, a distância entre especificação de projeto e sua realização final na linguagem alvo não foi reduzida. Estas ferramentas normalmente diluem as especificações de projeto pelo código de tal forma que, muitas vezes, nem o próprio projetista é capaz de identificá-las.

A presença de pronomes na linguagem utilizada para a implementação de um projeto auxilia a redução da distância entre ambas, uma vez que poupam o código da manutenção de algumas relações entre objetos. Alguns padrões de

projeto [G+95], por exemplo, podem ser implementados com pronomes de forma tão direta que a identificação destes no código final torna-se imediata.

A contribuição central desta tese é o estudo da utilização de alguns pronomes já propostos e a definição de dois novos pronomes. A todos é dado um novo enfoque para facilitar a incorporação destes em uma linguagem de propósito geral.

Este trabalho teve origem na especificação da linguagem TOOL [Car93]. Nesta, foi definido o pronome OWNER para representar o objeto criador do objeto corrente da execução. Um compilador para TOOL foi desenvolvido e, já nesta linguagem, foi criada uma série de ferramentas de auxílio ao desenvolvimento, como gerador de relatórios, construtor visual e ambiente de programação. Na construção destas ferramentas, o pronome OWNER se mostrou de crucial importância, facilitando e agilizando implementações que, em uma linguagem tradicional, se mostrariam bastante custosas. A utilização do pronome se mostrou tão vantajosa, na garantia de independência e reuso, que muitas vezes foram detectados desvios na modelagem ideal para que este pudesse ser utilizado, deixando claro que mais relações mereciam ser exploradas através de pronomes.

Como uma extensão deste trabalho, pelo menos no que diz respeito a pronomes, a linguagem COOL [Coe92] foi proposta e, nesta, já se verificava a existência de pronomes como ALL, PARENT e ANY. PARENT, apesar de não possuir a mesma semântica do pronome OWNER proposto anteriormente, foi introduzido no seu lugar no intuito de minimizar os desvios de modelagem. ALL e ANY, por sua vez, foram introduzidos na tentativa de aumentar a abrangência da comunicação através de pronomes. COOL não chegou a ser utilizado na implementação de sistemas para que se pudesse ter certeza que a substituição de OWNER por PARENT foi bem sucedida. Por representarem relações diferentes, mas igualmente importantes, estes dois pronomes, além de ALL e ANY, são explorados neste trabalho. Aqui, OWNER foi denominado CREATOR e a todos foi dada uma nova implementação utilizando-se linguagens abertas. Além do novo tratamento para estes pronomes, são ainda propostos, neste trabalho, os pronomes SENDER e MAIN, para representarem, respectivamente, o objeto emissor de uma mensagem e o objeto principal do sistema. A utilização de linguagens abertas para a introdução de pronomes em uma linguagem ao invés da alteração direta de seu compilador, como era o enfoque dado em TOOL e COOL, tornou esta inclusão menos traumática. Com este tipo de implementação, a linguagem pode ser configurada com os pronomes de

interesse de uma determinada aplicação, não gerando sobrecarga pela manutenção de pronomes inúteis para ela.

A experiência adquirida com TOOL, levou a especificação da linguagem DDL [Car97] para descrição textual de sistemas orientados a objetos. Sua principal proposta era minimizar descontinuidades entre projetos e códigos. Baseada nos conceitos desta linguagem, uma ferramenta CASE para construção de sistemas orientados a objetos, denominada 2GOOD [CCO98], foi desenvolvida. A garantia de que sistemas especificados na ferramenta eram totalmente orientados a objetos vinha do fato destes precisarem ser mapeados para construções DDL, que não admitiam construções que não o fossem.

Quando o desenvolvimento desta tese se iniciou, seu objetivo era introduzir pronomes em DDL. Naquele momento, ainda não se tinha definido quais pronomes seriam relevantes, mas tinha-se, da experiência acumulada pelo projeto TOOL, a certeza que seriam mais de um e que o conjunto escolhido deveria abranger uma gama grande de construções e relações entre objetos. A inexistência de um compilador para DDL e a pouca difusão que esta linguagem possuía fez com que a utilização de DDL como linguagem hospedeira dos pronomes fosse abandonada e um foco mais genérico fosse dado ao trabalho. Os rígidos conceitos de orientação a objetos que esta linguagem impunha foram, no entanto, respeitados.

Considerada como uma solução para problemas de reuso de *software*, a metodologia orientada a objetos passou a ser fortemente questionada quando começou a ser utilizada em sistemas grandes e complexos. Estes sistemas precisavam de novas e mais complexas abstrações, levando à criação de relações mais frágeis que herança e agregação. A criação explícita destas relações transformou sistemas complexos em verdadeiras “teias” de associações, aumentando consideravelmente o acoplamento entre classes e dificultando o reuso. Por causa disto, embora orientação a objetos continue a ser extensamente utilizada, esta utilização é feita em um nível bem abaixo de todo seu potencial. Na prática, reuso de classe não é apontado como um dos objetivos finais de um projeto, principalmente quando este se dá entre classes de diferentes sistemas. Encapsulamento, regra básica de orientação a objetos para a criação de sistemas adaptáveis, é também constantemente violado.

A redução da necessidade de associações, sem quebra de encapsulamento e sem que isso impossibilite a comunicação entre objetos, é a diretriz deste trabalho. Para alcançá-la, pronomes são utilizados para substituir algumas associações entre objetos e, com isso, permitir que o reuso possa ser

melhor explorado, tornando mais próximo o ideal de utilizar orientação a objetos em todo seu potencial.

A utilização de linguagens abertas na implementação de pronomes torna a incorporação destes em uma linguagem menos traumática, uma vez que o compilador desta não precisa ser alterado, possibilitando a criação de linguagens configuráveis por aplicação. Mesmo com estas, a definição de um novo pronome requer um conhecimento aprofundado de como pronomes devem ser implementados. Por outro lado, novos pronomes podem ser constantemente propostos para auxiliar um tipo específico de aplicação. Apesar de não ser o objetivo principal desta tese, um *framework* é aqui proposto para facilitar a tarefa de implementação de novos pronomes. Este *framework* é ainda instanciado para a obtenção de dois novos pronomes: CLOCK, para representar um relógio físico ou lógico do sistema, e SIBLINGS, para representar o conjunto de objetos que possuem o mesmo pai do objeto corrente da execução.

Os demais capítulos desta tese estão organizados da seguinte forma:

No Capítulo 2, os conceitos de comunicação baseada em pronomes são apresentados; aplicações que teriam suas implementações facilitadas com a utilização destes são exemplificadas; trabalhos de alguma forma relacionados a este são comparados; e é feito um resumo teórico de linguagens abertas a fim de justificar sua utilização na introdução de pronomes em uma linguagem.

Nos Capítulos 3 a 8, um conjunto de pronomes tidos como facilitadores na implementação de padrões de projetos e arquiteturas de software amplamente conhecidos é proposto. Neste conjunto, foram incluídos os pronomes: SENDER, CREATOR, PARENT, MAIN, ALL e ANY. Nesta ordem, eles são explorados e suas implementações detalhadas. Nestes capítulos, são ainda comparadas as implementações tradicionais das aplicações apresentadas no Capítulo 2, com as implementações utilizando estes pronomes.

No Capítulo 9, o *framework* é apresentado e instanciado para a obtenção dos pronomes CLOCK e SIBLINGS.

No Capítulo 10, o trabalho é concluído com uma análise dos resultados obtidos e um levantamento de possíveis futuros trabalhos.