# 8
# Trabalhos futuros

Nesta seção, detalhamos algumas das linhas de pesquisa a serem investigadas. Algumas das idéias aqui propostas surgem como necessidade óbvia após uma análise do trabalho apresentado, e serão visitadas prioritariamente. Outras surgiram casualmente da imaginação do autor, e são descompromissadas por natureza. Ao leitor é deixada a tarefa de separá-las.

## 8.1.Um Modelo de objetos para circuitos de objetos

Embora ao longo deste trabalho muito tenha sido falado a respeito do papel de objetos no universo de circuitos, não houve a preocupação, neste primeiro momento, de construir um modelo próprio daquilo que se entende por objeto. Pelo contrário, nos baseamos vagamente na definição de objetos no contexto da modelagem OO tradicional, como "entidade que possui estado e comportamento". Esta abordagem permitiu transmitir mais facilmente a idéia central do trabalho – a de objetos como transmissores de informação em circuitos – mas não é suficiente para uma apresentação formal da teoria.

Fica então clara a necessidade da criação de um modelo de objetos próprio, *a la* datatrons de Visual Circuit Board (*objectrons*?). Com uma definição formal de objetos, tornamos o modelo de circuitos de objetos mais homogêneo, restringindo o universo de representações de uma determinada computação. Em outras palavras, tornamos circuitos de objetos menos ambíguos. O desafio, porém, está em reduzir a ambigüidade sem prejuízo para a simplicidade e a elegância. Neste sentido, algumas questões se colocam de pronto:

- Uma vez que dispositivos já implementam funções – isto é, possuem comportamento – é interessante que objetos também o tenham? Um possível caminho é o de objetos como armazenadores de informação pura, ou seja, possuidores de estado, mas não de comportamento.
- O estado de um objeto pode ser alterado? A alteração do estado de um objeto é um efeito colateral e tende a tornar o modelo mais complexo.

Caso não seja permitido, um novo estado só seria alcançado com a criação de um novo objeto.

- Que objetos podem ser referenciados por um objeto? São permitidas referências cíclicas?
- Quais as classes de objetos fundamentais que devem ser incorporadas ao modelo de circuitos de objetos?
- Qual o suporte à criação de novas classes de objetos? Qual o papel de herança e polimorfismo, se é que possuem algum, e como são alcançadas?
- Qual a longevidade de um objeto? Onde é criado, e onde é destruído?

## 8.2.Catálogos de dispositivos

À medida que a quantidade de dispositivos aumenta, fica evidente a necessidade de algum mecanismo de classificação e organização. Sem este mecanismo, a simples tarefa de encontrar um dispositivo para determinado uso pode se tornar impraticável. Surge então o conceito de *catálogos de dispositivos*, para os mais variados propósitos. Um catálogo é uma estrutura hierárquica contendo especificações de dispositivos de alguma forma relacionados entre si – seja por semelhança de funcionalidade, domínio de aplicação, ou algum outro critério.

Catálogos são de alguma forma similares aos *pacotes* de classes da linguagem Java, ou aos *namespaces* de C#. Uma classe pertence a um único pacote ou namespace, mas este é um conceito que merece ser revisto no caso de catálogos, pois qualquer mecanismo de categorização deve prever a possibilidade de inclusão de uma entidade em mais de uma categoria. Em outras palavras, deve ser possível incluir um dispositivo em mais de um catálogo.

## 8.3.Cartas temporais

A propriedade de autonomia permite a criação de dispositivos cujo funcionamento interno é ditado unicamente pela passagem do tempo, e não por sinais externos recebidos por algum de seus barramentos. Num circuito com dispositivos desta natureza, a sincronia só pode ser alcançada se o comportamento do dispositivo ao longo do tempo for conhecido. Isto pode ser alcançado através

de *cartas temporais*: documentos contendo especificações do funcionamento de um dispositivo segundo uma série de eventos periódicos.

## 8.4.Linhas de tempo

Indo um pouco mais além, podemos extrapolar o conceito de tempo como uma dimensão única, tal como ocorre num circuito elétrico real. A conveniência de trabalhar no mundo das abstrações nos permite redefinir o conceito de tempo, que passa então a ser uma propriedade *interna* do dispositivo.

Para todos os efeitos, o *tempo* é uma grandeza que dita a velocidade na qual a computação de um determinado dispositivo é realizada, seja ele atômico ou composto. Neste último caso, o que ocorre é que a velocidade de computação de cada um dos dispositivos internos é afetada pela linha de tempo do dispositivo pai, de modo recursivo.

Ao adotar esta estratégia, o design de sistemas onde determinadas computações são executadas mais rapidamente que outras é feito de forma elegante, pois o tempo possui uma hierarquia, a saber, a mesma dos dispositivos que compõem o circuito. A elegância advém do fato de que é *sempre* possível ajustar o ritmo da passagem de tempo de um dispositivo composto sem afetar o funcionamento das partes que o compõem. Para um dispositivo filho, a mudança de tempo no pai é imperceptível, na medida em que afeta a ele próprio e a todos os outros filhos de forma homogênea.

## 8.5.Esquemas

Num circuito elétrico, um *esquema* é uma representação diagramática dos componentes do circuito e suas ligações, contendo informações de layout (disposição dos elementos na placa) e anotações que auxiliam a identificar partes e entender o funcionamento. Sua utilidade como artefato de documentação do sistema é óbvia; logo, seria uma injustiça que circuitos de objetos não possuíssem também uma linguagem para construção de esquemas.

Nasce então a idéia de desenvolver um modelo de esquema de circuitos de objetos utilizando a linguagem XML, que permita relacionar os elementos de um circuito, bem como decorá-los com rótulos e todo tipo de anotações útil.

É interessante notar que esquemas também têm um papel no que diz respeito à introspecção, pois somente com um meta-modelo desta natureza pode-se prover aos dispositivos meios para obter e manipular informações sobre o circuito a que pertencem.

## 8.6.Checagem de tipos

Classes de objetos são especificações de *tipos*, e linguagens de programação OO utilizam informações sobre a estrutura das classes para fazer validações no código de um programa, tanto em tempo de compilação quanto de execução, conhecido como *checagem de tipo*. A checagem ocorre em um número de situações, tais como chamada de um método, conversão explícita ou *downcasting*, etc.

No âmbito de circuitos de objetos, a checagem de tipos pode ser feita da seguinte forma: ao "anotar" a porta de um barramento com a designação de um dado tipo, especifica-se que o potencial do ramo da porta deve, em qualquer instante, ser compatível com este tipo (sendo o potencial nulo $\xi$ o único compatível com todos os tipos). Então, é possível detectar erros na lógica do circuito ainda na etapa de construção, por exemplo, ao conectar portas associadas a tipos incompatíveis. Nada impede, entretanto, que várias portas tipadas sejam conectadas, desde que respeitada a compatibilidade, ou seja, desde que o potencial do ramo seja compatível com todas as portas tipadas.

Possivelmente, o conceito de circuitos elétricos que mais se aproxima da idéia de tipo num circuito de objetos é o de *impedância* (resistência dinâmica), visto que uma porta tipada oferece resistência à passagem de objetos cujos tipos são incompatíveis com o tipo da porta.

## 8.7.Ferramentas de desenvolvimento

O apelo visual por trás de circuitos de objetos motiva a criação de ambientes de desenvolvimento visuais, com várias funcionalidades, tais como:

- Navegação amigável: *zoom in*, *zoom out*, visão estrutural da árvore de dispositivos, etc.
- Busca de dispositivos baseada em critérios específicos: nome, catálogo(s) a que pertence, função implementada, etc.

- Visualização do fluxo de objetos num circuito em execução.
- Gerenciamento de catálogos: inclusão, remoção e edição de dispositivos.

## 8.8.JOCA e agentes de software

A arquitetura JOCA faz uso limitado do conceito de agentes de software como forma de implementar o aspecto autônomo de um dispositivo atômico. No estágio atual da arquitetura, o trabalho de programação de um dispositivo simples ainda é grande, e não há muito espaço para reutilização. A idéia é expandir o uso de agentes de software na especificação de dispositivos atômicos, construindo um amplo modelo de sistemas multi-agentes que dêem ao designer mais flexibilidade e simplicidade.

## 8.9.Outros trabalhos

- Design de mecanismos para suporte à introspecção e à evolução dinâmica.
- Estudo de técnicas da Engenharia de Software aplicadas a circuitos de objetos, tais como *design patterns* e contratos.
- Estudo de técnicas de design de circuitos elétricos aplicadas a circuitos de objetos, tais como tolerância à falhas e redundância.