6

Conclusões

Neste trabalho, desenvolvemos uma infra-estrutura para dar apoio à execução de componentes de *software*, composta por três serviços principais. Esses serviços permitem o acesso a nós da rede para a criação de contêineres (Nó de Execução), a carga e execução de diferentes componentes em uma mesma área de memória (Contêiner de Componentes), e a instalação de implementações de componentes para que se tornem disponíveis no ambiente distribuído (Repositório de Componentes), entre outras funcionalidades. O conjunto de tais facilidades representa técnicas de implantação e execução que se baseiam em trabalhos representativos do estado da arte da área, mas ainda se diferenciando, por exemplo, através de funcionalidades não muito abordadas, como a de suspensão de contêineres, e por permitir a carga de diferentes tipos de componentes num mesmo processo.

Os serviços da infra-estrutura de execução se baseiam em um conjunto de operações bem reduzido, o que facilita seu aprendizado. Para isso, abordamos apenas as características que consideramos essenciais a uma infra-estrutura para a execução de componentes de *software*. Devido a estas características, acreditamos ter atingido o objetivo de mantê-la mais simples e objetiva que os outros trabalhos mencionados. A possibilidade de instalação, execução e configuração de componentes em tempo real permite a criação de muitos serviços interessantes em nível mais alto, que certamente se tornarão alvo de estudos posteriores. Atualmente, o sistema já é utilizado como ferramenta de ensino em disciplinas para o estudo e experimentação de técnicas relacionadas a componentes de *software*.

As avaliações apresentadas no capítulo 5 mostram que o uso da arquitetura por si só implica em um custo relativamente baixo em termos de desempenho. O uso de memória não é diferente, agregando pouco ao uso mínimo exigido pelo ORB. No entanto, o uso das facilidades de interceptação tem um custo mais alto, mesmo com apenas o interceptador CORBA instalado, sem

nenhum interceptador OpenBus estando em uso. Mas, como descrevemos, as maiores vantagens estão na aplicação do paradigma de programação orientada a componentes e no uso da infra-estrutura de execução, que diminuem sensivelmente o trabalho necessário para o desenvolvimento e manutenção de aplicações distribuídas, como no exemplo da aplicação de testes distribuídos.

6.1

Trabalhos Futuros

O sistema OpenBus ainda pode contar com diversas melhorias e novas funcionalidades, que podem ser adicionadas à sua base. Por exemplo, a implementação de repositório provida é extremamente básica. Diversos estudos como [9, 50, 2] demonstram técnicas mais eficazes e interessantes, como a especificação de recursos de *hardware* e *software* necessários. Uma possível integração seria com o projeto LuaRocks [51]. Uma implementação melhorada do repositório poderia transformar implementações recebidas de componentes Lua em LuaRocks e utilizar seus metadados ao colaborar com outros repositórios ou contêineres, para instalar dependências automaticamente.

Alguns serviços antigos do projeto OpenBus ainda não utilizam o modelo SCS completo, e também não utilizam os serviços apresentados no Capítulo 4. Particularmente, o Serviço de Sessão poderia ser modificado para se encarregar da carga completa de sessões. Dada uma descrição onde estejam explicitados os componentes necessários, suas propriedades e suas dependências, o serviço se encarregaria de realizar todo o procedimento necessário para que entrassem em funcionamento, realizando operações como:

1. Criar ou reutilizar um contêiner;
2. Criar ou reutilizar um repositório;
3. Instalar componentes, se necessário;
4. Carregar componentes especificados e suas dependências;
5. Realizar conexões;
6. Inicializar os componentes principais de uma aplicação.

Tal funcionalidade poderia ser parametrizada para obter diferentes resultados, facilitando a implantação. O Serviço de Controle de Acesso pode ser

melhor aproveitado, pois ainda não é aplicada nenhuma política de controle de acesso aos serviços da infra-estrutura de execução. Atualmente, se um componente tem uma credencial válida no barramento, tem permissão para utilizar todos os serviços de execução. Tais políticas são necessárias para se aumentar a segurança e controle dentro dos barramentos.

Interfaces gráficas e via linha de comando para usuários, que permitam o uso dos serviços e a composição visual de aplicações sem a necessidade de escrita de código, também são frentes de trabalho interessantes. Independente do tipo de interface, é importante também fornecer futuramente uma ferramenta para o acompanhamento da execução de algoritmos. Podemos nos aproveitar de ferramentas existentes, como o projeto SGA.

Em nossas implementações, nos focamos em implementações Lua, mas certas funcionalidades exigem implementações próprias para cada linguagem. Um bom exemplo é o contêiner, e atualmente não existe uma implementação Java com todas as funcionalidades que descrevemos, como por exemplo a faceta de suspensão. Estas funcionalidades devem ser equiparadas entre as diferentes implementações de contêiner. Ainda em relação ao contêiner, pretendemos futuramente prover facilidades para a carga de componentes locais, não necessariamente pertencentes às tecnologias CORBA e SCS, como descrito na Seção 4.2.

Este trabalho abre portas para diversos trabalhos futuros, principalmente em relação a qualidade de serviço e gerenciamento automático de recursos do sistema. Em nosso grupo de trabalho, outras pessoas já estão trabalhando em alguns projetos desses tipos, baseados nas implementações Lua e Java. Por exemplo, um trabalho em andamento realiza a coleta de métricas de desempenho no Nó de Execução e Contêiner de Componentes. Um outro trabalho, também ainda em desenvolvimento, utiliza as métricas do primeiro para realizar detecção de falhas e prover um escalonamento para tarefas de aplicações, escolhendo automaticamente os nós mais apropriados para a execução de cada tarefa. Estas áreas são amplas e ainda podem gerar diversos outros trabalhos.