Arquitetura de software

Apesar do interesse pela arquitetura de software como campo de pesquisa, há pouco consenso entre os pesquisadores sobre o que exatamente deve ser incluído na definição de arquitetura. Em muitos casos, isso fez com que aspectos importantes do projeto arquitetônico fossem negligenciados por pesquisas anteriores. Este capítulo define uma terminologia autoconsistente para arquitetura de software com base em um exame das definições existentes na literatura e em meu próprio insight a respeito de arquiteturas de aplicativos baseadas em rede. Cada definição, destacada em uma caixa para facilitar a referência, é seguida por uma discussão de como ela é derivada ou se compara a pesquisas relacionadas.

1.1 Abstração em tempo de execução

Uma arquitetura de software é uma abstração dos elementos de tempo de execução de um sistema de software durante alguma fase de sua operação. Um sistema pode ser composto de vários níveis de abstração e muitas fases de operação, cada uma com sua própria arquitetura de software.

No coração da arquitetura de software está o princípio da abstração: ocultar alguns dos detalhes de um sistema por meio do encapsulamento, a fim de melhor identificar e sustentar suas propriedades [117]. Um sistema complexo conterá muitos níveis de abstração, cada um com sua própria arquitetura. Uma arquitetura representa uma abstração do comportamento do sistema naquele nível, de modo que os elementos arquitetônicos são delineados pelas interfaces abstratas que fornecem a outros elementos naquele nível [9]. Dentro de cada elemento pode ser encontrada uma outra arquitetura, definindo o sistema de subelementos que implementam o comportamento representado pela interface abstrata do elemento pai. Essa recursão de arquiteturas continua até os elementos mais básicos do sistema: aqueles que não podem ser decompostos em elementos menos abstratos.

Além dos níveis de arquitetura, um sistema de software geralmente terá várias fases operacionais, como inicialização, inicialização, processamento normal, reinicialização e desligamento. Cada fase operacional possui sua própria arquitetura. Por exemplo, um arquivo de configuração será tratado como um elemento de dados durante a fase de inicialização, mas não será considerado um elemento arquitetônico durante o processamento normal, pois nesse ponto as informações nele contidas já estarão distribuídas por todo o sistema. Pode, de fato, ter definido a arquitetura de processamento normal. Uma descrição geral de uma arquitetura de sistema deve ser capaz de descrever não apenas o comportamento operacional da arquitetura do sistema durante cada fase, mas também a arquitetura de transições entre fases.

Perry e Wolf [105] definem os elementos de processamento como "transformadores de dados", enquanto Shaw et al. [118] descreve os componentes como "o locus da computação e do estado." Isso é esclarecido em Shaw e Clements [122]: "Um componente é uma unidade de software que executa alguma função em tempo de execução. Os exemplos incluem programas, objetos, processos e filtros." Isso levanta uma distinção importante entre a arquitetura de software e o que é tipicamente referido como estrutura de software: a primeira é uma abstração do comportamento em tempo de execução de um sistema de software, enquanto a última é uma propriedade do código-fonte estático do software. Embora haja vantagens em ter a estrutura modular do código-fonte compatível com a decomposição do comportamento dentro de um sistema em execução, também há vantagens em ter componentes de software independentes sendo implementados usando partes do mesmo código (por exemplo, bibliotecas compartilhadas). Separamos a visão da arquitetura do software daquela do código-fonte para enfocar nas características de tempo de execução do software, independentemente da implementação de um determinado componente. Portanto, o projeto arquitetônico e o projeto estrutural do código-fonte, embora intimamente relacionados, são atividades de projeto separadas. Infelizmente, algumas descrições de arquitetura de software falham em fazer essa distinção (por exemplo, [9]).

1.2 Elementos

Uma arquitetura de software é definida por uma configuração de elementos arquitetônicos - componentes, conectores e dados - restritos em seus relacionamentos para atingir um conjunto desejado de propriedades arquitetônicas.

Um exame abrangente do escopo e da base intelectual da arquitetura de software pode ser encontrado em Perry e Wolf [105]. Eles apresentam um modelo que define uma arquitetura de software como um conjunto de elementos arquitetônicos que possuem uma forma particular, explicada por um conjunto de lógicas. Os elementos arquitetônicos incluem processamento, dados e elementos de conexão. A forma é definida pelas propriedades dos elementos e pelos relacionamentos entre os elementos - ou seja, as restrições dos elementos. O fundamento lógico fornece a base subjacente para a arquitetura, capturando a motivação para a escolha do estilo arquitetônico, a escolha dos elementos e a forma.

Minhas definições para arquitetura de software são uma versão elaborada daquelas do modelo de Perry e Wolf [105], exceto que excluo a lógica. Embora o fundamento lógico seja um aspecto importante da pesquisa da arquitetura de software e da descrição da arquitetura em particular, incluí-lo na definição da arquitetura de software implicaria que a documentação do projeto é parte do sistema de tempo de execução. A presença ou ausência de fundamento lógico pode influenciar a evolução de uma arquitetura, mas, uma vez constituída, a arquitetura independe de suas razões de ser. Os sistemas reflexivos [80] podem usar as características do desempenho passado para mudar o comportamento futuro, mas ao fazer isso eles estão substituindo uma arquitetura de nível inferior por outra arquitetura de nível inferior, em vez de englobar a lógica dentro dessas arquiteturas.

Como ilustração, considere o que acontece com uma construção se suas plantas e planos de projeto forem queimados. O edifício desaba imediatamente? Não, uma vez que as propriedades pelas quais as paredes sustentam o peso do telhado permanecem intactas. Uma arquitetura tem, por design, um conjunto de propriedades que permitem atender ou exceder os requisitos do sistema. O desconhecimento dessas propriedades pode levar a alterações posteriores que violam a arquitetura, assim como a substituição de uma parede de suporte por uma grande moldura de janela pode violar a estabilidade estrutural de um edifício. Portanto, em vez de justificativa, nossa definição de arquitetura de software inclui propriedades arquitetônicas. A justificativa explica essas propriedades, e a falta de justificativa pode resultar em decadência ou degradação gradual da arquitetura ao longo do tempo, mas a justificativa em si não faz parte da arquitetura.

Uma característica chave do modelo em Perry e Wolf [105] é a distinção dos vários tipos de elementos. Os elementos de processamento são aqueles que realizam transformações nos dados, os elementos de dados são aqueles que contêm as informações que são usadas e transformadas e os elementos de conexão são a cola que mantém as diferentes peças da arquitetura juntas. Eu uso os termos mais prevalentes de componentes e conectores para me referir a elementos de processamento e conexão, respectivamente.

Garlan e Shaw [53] descrevem a arquitetura de um sistema como uma coleção de componentes computacionais junto com uma descrição das interações entre esses componentes - os conectores. Este modelo é expandido em Shaw et al. [118]: A arquitetura de um sistema de software define esse sistema em termos de componentes e de interações entre esses componentes. Além de especificar a estrutura e a topologia do sistema, a arquitetura mostra a correspondência pretendida entre os requisitos do sistema e os elementos do sistema construído. Elaboração posterior desta definição pode ser encontrada em Shaw e Garlan [121].

O que é surpreendente sobre o Shaw et al. [118] modelo é que, ao invés de definir a arquitetura do software como existente dentro do software, é definir uma descrição da arquitetura do software como se essa fosse a arquitetura. No processo, a arquitetura de software como um todo é reduzida ao que é comumente encontrado na maioria dos diagramas de arquitetura informal: caixas (componentes) e linhas (conectores). Os elementos de dados, junto com muitos dos aspectos dinâmicos das arquiteturas de software reais, são ignorados. Esse modelo é incapaz de descrever adequadamente as arquiteturas de software baseadas em rede, uma vez que a natureza, a localização e o movimento dos elementos de dados dentro do sistema costumam ser os determinantes mais significativos do comportamento do sistema.

1.2.1 Componentes

Os componentes são o aspecto mais facilmente reconhecido da arquitetura de software. Os elementos de processamento de Perry e Wolf [105] são definidos como aqueles componentes que fornecem a transformação nos elementos de dados. Garlan e Shaw [53] descrevem componentes simplesmente como os elementos que executam a computação. Nossa definição tenta ser mais precisa ao fazer a distinção entre os componentes e o software dentro dos conectores.

1.2.2 Conectores

Um conector é um mecanismo abstrato que medeia a comunicação, coordenação ou cooperação entre componentes.

Perry e Wolf [105] descrevem a conexão de elementos vagamente como a cola que mantém as várias peças da arquitetura juntas. Uma definição mais precisa é fornecida por Shaw e Clements [122]: Um conector é um mecanismo abstrato que medeia a comunicação, coordenação ou cooperação entre componentes. Os exemplos incluem representações compartilhadas, chamadas de procedimento remoto, protocolos de passagem de mensagens e fluxos de dados.

Talvez a melhor maneira de pensar sobre conectores seja compará-los com componentes. Os conectores permitem a comunicação entre os componentes, transferindo elementos de dados de uma interface para outra sem alterar os dados. Internamente, um conector pode consistir em um subsistema de componentes que transformam os dados para transferência, realizam a transferência e, em seguida, revertem a transformação para entrega. No entanto, a abstração comportamental externa capturada pela arquitetura ignora esses detalhes. Em contraste, um componente pode, mas nem sempre, transformar os dados de uma perspectiva externa

1.2.3 Dados

Um datum é um elemento de informação que é transferido de um componente, ou recebido por um componente, por meio de um conector.

Como observado acima, a presença de elementos de dados é a distinção mais significativa entre o modelo de arquitetura de software definido por Perry e Wolf [105] e o modelo usado por grande parte da pesquisa rotulada arquitetura de software [1, 5, 9, 53, 56 , 117-122, 128]. Boasson [24] critica a pesquisa atual de arquitetura de software por sua ênfase em estruturas de componentes e ferramentas de desenvolvimento de arquitetura, sugerindo que mais foco deve ser colocado na modelagem de arquitetura centrada em dados. Comentários semelhantes são feitos por Jackson [67].

**1.3 Configurations**

Uma configuração é a estrutura de relacionamentos arquitetônicos entre componentes, conectores e dados durante um período de tempo de execução do sistema.

Abowd et al. [1] definem a descrição arquitetônica como suporte à descrição de sistemas em termos de três classes sintáticas básicas: componentes, que são o locus da computação; conectores, que definem as interações entre os componentes; e configurações, que são coleções de componentes e conectores que interagem. Várias notações concretas de estilo específico podem ser usadas para representá-los visualmente, facilitar a descrição de computações e interações legais e restringir o conjunto de sistemas desejáveis.

Estritamente falando, pode-se pensar em uma configuração como sendo equivalente a um conjunto de restrições específicas na interação do componente. Por exemplo, Perry e Wolf [105] incluem topologia em sua definição de relacionamentos de forma arquitetônica. No entanto, separar a topologia ativa das restrições mais gerais permite que um arquiteto distinga mais facilmente a configuração ativa do domínio potencial de todas as configurações legítimas. A justificativa adicional para distinguir configurações dentro de linguagens de descrição arquitetônica é apresentada em Medvidovic e Taylor [86].

1.4 Propriedades

O conjunto de propriedades arquitetônicas de uma arquitetura de software inclui todas as propriedades que derivam da seleção e organização de componentes, conectores e dados dentro do sistema. Os exemplos incluem tanto as propriedades funcionais alcançadas pelo sistema quanto as propriedades não funcionais, como relativa facilidade de evolução, capacidade de reutilização de componentes, eficiência e extensibilidade dinâmica, muitas vezes referida como atributos de qualidade [9].

As propriedades são induzidas pelo conjunto de restrições dentro de uma arquitetura. As restrições são frequentemente motivadas pela aplicação de um princípio de engenharia de software [58] a um aspecto dos elementos arquitetônicos. Por exemplo, o estilo tubo e filtro uniforme obtém as qualidades de reutilização de componentes e configurabilidade do aplicativo, aplicando generalidade a suas interfaces de componentes - restringindo os componentes a um único tipo de interface. Portanto, a restrição arquitetônica é "interface de componente uniforme", motivada pelo princípio de generalidade, a fim de obter duas qualidades desejáveis ​​que se tornarão as propriedades arquitetônicas de componentes reutilizáveis ​​e configuráveis ​​quando esse estilo for instanciado em uma arquitetura. O objetivo do projeto arquitetônico é criar uma arquitetura com um conjunto de propriedades arquitetônicas que formam um superconjunto dos requisitos do sistema. A importância relativa das várias propriedades arquitetônicas depende da natureza do sistema pretendido. A Seção 2.3 examina as propriedades que são de interesse particular para arquiteturas de aplicativos baseadas em rede.

1.5 Estilos

Um estilo arquitetônico é um conjunto coordenado de restrições arquitetônicas que restringe as funções / recursos dos elementos arquitetônicos e os relacionamentos permitidos entre esses elementos em qualquer arquitetura que esteja em conformidade com esse estilo.

Como uma arquitetura incorpora propriedades funcionais e não funcionais, pode ser difícil comparar diretamente arquiteturas para diferentes tipos de sistemas, ou até mesmo para o mesmo tipo de sistema definido em ambientes diferentes. Os estilos são um mecanismo para categorizar arquiteturas e definir suas características comuns [38]. Cada estilo fornece uma abstração para as interações de componentes, capturando a essência de um padrão de interação, ignorando os detalhes incidentais do resto da arquitetura [117].

Perry e Wolf [105] definem estilo arquitetônico como uma abstração de tipos de elementos e aspectos formais de várias arquiteturas específicas, talvez concentrando-se apenas em certos aspectos de uma arquitetura. Um estilo arquitetônico encapsula decisões importantes sobre os elementos arquitetônicos e enfatiza restrições importantes sobre os elementos e seus relacionamentos. Essa definição permite estilos que se concentram apenas nos conectores de uma arquitetura ou em aspectos específicos das interfaces do componente.

Em contraste, Garlan e Shaw [53], Garlan et al. [56], e Shaw e Clements [122], todos definem o estilo em termos de um padrão de interações entre os componentes digitados. Especificamente, um estilo arquitetônico determina o vocabulário de componentes e conectores que podem ser usados ​​em instâncias desse estilo, junto com um conjunto de restrições sobre como eles podem ser combinados [53]. Essa visão restrita de estilos arquitetônicos é um resultado direto de sua definição de arquitetura de software - pensar na arquitetura como uma descrição formal, ao invés de um sistema em execução, leva a abstrações baseadas apenas nos padrões compartilhados de diagramas de caixa e linha. Abowd et al. [1] vão além e definem isso explicitamente como ver a coleção de convenções que são usadas para interpretar uma classe de descrições arquitetônicas como definindo um estilo arquitetônico.

Novas arquiteturas podem ser definidas como instâncias de estilos específicos [38]. Uma vez que os estilos arquitetônicos podem abordar diferentes aspectos da arquitetura de software, uma determinada arquitetura pode ser composta de vários estilos. Da mesma forma, um estilo híbrido pode ser formado pela combinação de vários estilos básicos em um único estilo coordenado.

Infelizmente, usar o termo estilo para se referir a um conjunto coordenado de restrições geralmente leva à confusão. Esse uso difere substancialmente da etimologia do estilo, que enfatizaria a personalização do processo de design. Loerke [76] dedica um capítulo para denegrir a noção de que as preocupações estilísticas pessoais têm qualquer lugar no trabalho de um arquiteto profissional. Em vez disso, ele descreve os estilos como a visão dos críticos da arquitetura passada, em que a escolha de materiais disponíveis, a cultura da comunidade ou o ego do governante local eram responsáveis ​​pelo estilo arquitetônico, não o designer. Em outras palavras, Loerke vê a verdadeira fonte de estilo na arquitetura de construção tradicional como o conjunto de restrições aplicadas ao projeto, e obter ou copiar um estilo específico deve ser o menor dos objetivos do projetista. Visto que referir-se a um determinado conjunto de restrições como um estilo torna mais fácil comunicar as características das restrições comuns, usamos os estilos arquitetônicos como um método de abstração, em vez de um indicador de design personalizado.

1.6 Padrões e linguagens de padrões

Paralelamente à pesquisa de engenharia de software em estilos arquitetônicos, a comunidade de programação orientada a objetos tem explorado o uso de padrões de projeto e linguagens de padrões para descrever abstrações recorrentes no desenvolvimento de software baseado em objetos. Um padrão de projeto é definido como uma construção de sistema importante e recorrente. Uma linguagem de padrões é um sistema de padrões organizado em uma estrutura que orienta a aplicação dos padrões [70]. Ambos os conceitos são baseados nos escritos de Alexander et al. [3, 4] no que diz respeito à arquitetura de edifícios.

O espaço de design de padrões inclui questões de implementação específicas para as técnicas de programação orientada a objetos, como herança de classe e composição de interface, bem como as questões de design de alto nível abordadas por estilos arquitetônicos [51]. Em alguns casos, as descrições do estilo arquitetônico foram reformuladas como padrões arquitetônicos [120]. No entanto, um benefício primário dos padrões é que eles podem descrever protocolos relativamente complexos de interações entre objetos como uma única abstração [91], incluindo, assim, tanto as restrições de comportamento quanto as especificações da implementação. Em geral, um padrão, ou linguagem de padrão, no caso de vários padrões integrados, pode ser considerado uma receita para implementar um conjunto desejado de interações entre objetos. Em outras palavras, um padrão define um processo para resolver um problema seguindo um caminho de opções de design e implementação [34].

Como os estilos de arquitetura de software, a pesquisa de padrões de software se desviou um pouco de sua origem na arquitetura de construção. Na verdade, a noção de padrões de Alexander centra-se não em arranjos recorrentes de elementos arquitetônicos, mas sim no padrão recorrente de eventos - atividade humana e emoção - que ocorrem dentro de um espaço, com a compreensão de que um padrão de eventos não pode ser separado o espaço onde ocorre [3]. A filosofia de design de Alexander é identificar padrões de vida que são comuns à cultura alvo e determinar quais restrições arquitetônicas são necessárias para diferenciar um determinado espaço de forma que permita que os padrões desejados ocorram naturalmente. Esses padrões existem em vários níveis de abstração e em todas as escalas.

Como um elemento no mundo, cada padrão é uma relação entre um determinado contexto, um certo sistema de forças que ocorre repetidamente naquele contexto e uma certa configuração espacial que permite que essas forças se resolvam. Como um elemento da linguagem, um padrão é uma instrução, que mostra como essa configuração espacial pode ser usada, uma e outra vez, para resolver o dado sistema de forças, onde quer que o contexto o torne relevante. O padrão é, em suma, ao mesmo tempo uma coisa que acontece no mundo e a regra que nos diz como criar essa coisa e quando devemos criá-la. É um processo e uma coisa; tanto uma descrição de uma coisa que está viva, quanto uma descrição do processo que irá gerar essa coisa. [3]

De muitas maneiras, os padrões de Alexander têm mais em comum com os estilos de arquitetura de software do que os padrões de projeto da pesquisa OOPL. Um estilo arquitetônico, como um conjunto coordenado de restrições, é aplicado a um espaço de projeto para induzir as propriedades arquitetônicas desejadas do sistema. Ao aplicar um estilo, um arquiteto diferencia o espaço de design de software na esperança de que o resultado corresponda melhor às forças inerentes ao aplicativo, levando assim a um comportamento do sistema que aprimora o padrão natural em vez de entrar em conflito com ele.

1.7 Visualizações

Um ponto de vista arquitetônico geralmente é específico do aplicativo e varia amplamente com base no domínio do aplicativo. ... vimos pontos de vista arquitetônicos que abordam uma variedade de questões, incluindo: questões temporais, abordagens de estado e controle, representação de dados, ciclo de vida da transação, salvaguardas de segurança e pico de demanda e degradação normal. Sem dúvida, existem muitos outros pontos de vista possíveis. [70]

Além das muitas arquiteturas dentro de um sistema e dos muitos estilos arquitetônicos dos quais as arquiteturas são compostas, também é possível visualizar uma arquitetura de muitas perspectivas diferentes. Perry e Wolf [105] descrevem três visões importantes na arquitetura de software: visão de processamento, dados e conexão. Uma visualização de processo enfatiza o fluxo de dados por meio dos componentes e alguns aspectos das conexões entre os componentes com relação aos dados. Uma visualização de dados enfatiza o fluxo de processamento, com menos ênfase nos conectores. Uma visualização de conexão enfatiza o relacionamento entre os componentes e o estado de comunicação.

Múltiplas visões arquitetônicas são comuns em estudos de caso de arquiteturas específicas [9]. Uma metodologia de projeto arquitetônico, o 4 + 1 View Model [74], organiza a descrição de uma arquitetura de software usando cinco visualizações simultâneas, cada uma das quais abordando um conjunto específico de interesses.

1.8 Trabalho Relacionado

Incluo aqui apenas as áreas de pesquisa que definem a arquitetura de software ou descrevem estilos de arquitetura de software. Outras áreas de pesquisa de arquitetura de software incluem técnicas de análise arquitetônica, recuperação e reengenharia de arquitetura, ferramentas e ambientes para projeto arquitetônico, refinamento de arquitetura da especificação à implementação e estudos de caso de arquiteturas de software implantadas [55]. Trabalhos relacionados nas áreas de classificação de estilo, paradigmas de processos distribuídos e middleware são discutidos no Capítulo 3.

1.8.1 Metodologias de Projeto

A maioria das pesquisas iniciais sobre arquitetura de software concentrou-se em metodologias de design. Por exemplo, o projeto orientado a objetos [25] defende uma maneira de estruturar problemas que leva naturalmente a uma arquitetura baseada em objetos (ou, mais precisamente, não leva naturalmente a qualquer outra forma de arquitetura). Uma das primeiras metodologias de projeto a enfatizar o projeto no nível arquitetônico é o Jackson System Development [30]. O JSD estrutura intencionalmente a análise de um problema de modo que leve a um estilo de arquitetura que combina tubo e filtro (fluxo de dados) e restrições de controle de processo. Essas metodologias de design tendem a produzir apenas um estilo de arquitetura.

1.8.2 Manuais para projeto, padrões de projeto e linguagens de padrões

Shaw [117] defende o desenvolvimento de manuais de arquitetura nas mesmas linhas das disciplinas tradicionais de engenharia. A comunidade de programação orientada a objetos assumiu a liderança na produção de catálogos de padrões de projeto, como exemplificado pelo livro "Gang of Four" [51] e os ensaios editados por Coplien e Schmidt [33]. Os padrões de projeto de software tendem a ser mais orientados para o problema do que os estilos arquitetônicos. Shaw [120] apresenta oito exemplos de padrões arquitetônicos baseados nos estilos arquitetônicos descritos em [53], incluindo informações sobre os tipos de problemas mais adequados para cada arquitetura. Buschmann et al. [28] fornecem um exame abrangente dos padrões arquitetônicos comuns ao desenvolvimento baseado em objeto. Ambas as referências são puramente descritivas e não fazem nenhuma tentativa de comparar ou ilustrar as diferenças entre os padrões arquitetônicos. Tepfenhart e Cusick [129] usam um mapa bidimensional para diferenciar entre taxonomias de domínio, modelos de domínio, estilos arquitetônicos, frameworks, kits, padrões de projeto e aplicações. Na topologia, os padrões de design são estruturas de design predefinidas usadas como blocos de construção para uma arquitetura de software, enquanto os estilos de arquitetura são conjuntos de características operacionais que identificam uma família arquitetônica independente do domínio do aplicativo. No entanto, eles falham em definir a própria arquitetura.

1.8.3 Modelos de referência e arquiteturas de software específicas de domínio (DSSA)

Os modelos de referência são desenvolvidos para fornecer estruturas conceituais para descrever arquiteturas e mostrar como os componentes estão relacionados entre si [117]. A Object Management Architecture (OMA), desenvolvida pelo OMG [96] como um modelo de referência para arquiteturas de objetos distribuídos intermediários, especifica como os objetos são definidos e criados, como os aplicativos clientes chamam os objetos e como os objetos podem ser compartilhados e reutilizados. A ênfase está no gerenciamento de objetos distribuídos, em vez de na interação eficiente do aplicativo.

Hayes-Roth et al. [62] definem a arquitetura de software de domínio específico (DSSA) como compreendendo: a) uma arquitetura de referência, que descreve uma estrutura computacional geral para um domínio significativo de aplicações, b) uma biblioteca de componentes, que contém pedaços reutilizáveis ​​de experiência de domínio, e c ) um método de configuração de aplicativo para selecionar e configurar componentes dentro da arquitetura para atender a requisitos específicos de aplicativo. Tracz [130] fornece uma visão geral do DSSA.

Os projetos de DSSA foram bem-sucedidos na transferência de decisões de arquitetura para sistemas em execução, restringindo o espaço de desenvolvimento de software a um estilo de arquitetura específico que corresponda aos requisitos de domínio [88]. Os exemplos incluem ADAGE [10] para aviônicos, AIS [62] para sistemas inteligentes adaptativos e MetaH [132] para sistemas de orientação, navegação e controle de mísseis. O DSSA enfatiza a reutilização de componentes dentro de um domínio arquitetônico comum, em vez de selecionar um estilo arquitetônico específico para cada sistema.

1.8.4 Linguagens de descrição de arquitetura (ADL)

A maior parte dos trabalhos publicados recentes sobre arquiteturas de software está na área de linguagens de descrição de arquitetura (ADL). Uma ADL é, de acordo com Medvidovic e Taylor [86], uma linguagem que fornece recursos para a especificação e modelagem explícita da arquitetura conceitual de um sistema de software, incluindo, no mínimo: componentes, interfaces de componentes, conectores e configurações arquitetônicas.

Darwin é uma linguagem declarativa que pretende ser uma notação de propósito geral para especificar a estrutura de sistemas compostos por diversos componentes usando diversos mecanismos de interação [81]. As qualidades interessantes de Darwin são que ele permite a especificação de arquiteturas distribuídas e arquiteturas compostas dinamicamente [82].

UniCon [118] é uma linguagem e um conjunto de ferramentas associado para compor uma arquitetura a partir de um conjunto restrito de exemplos de componentes e conectores. Wright [5] fornece uma base formal para especificar as interações entre os componentes arquitetônicos, especificando os tipos de conectores por seus protocolos de interação.

Assim como as metodologias de projeto, ADLs frequentemente introduzem suposições de arquitetura específicas que podem impactar sua capacidade de descrever alguns estilos de arquitetura e podem entrar em conflito com as suposições do middleware existente [38]. Em alguns casos, uma ADL é projetada especificamente para um único estilo arquitetônico, melhorando assim sua capacidade de descrição e análise especializadas ao custo da generalidade. Por exemplo, C2SADEL [88] é um ADL projetado especificamente para descrever arquiteturas desenvolvidas no estilo C2 [128]. Em contraste, ACME [57] é uma ADL que tenta ser o mais genérica possível, mas com a desvantagem de não suportar a análise específica de estilo e a construção de aplicativos reais; em vez disso, seu foco está no intercâmbio entre ferramentas de análise.

1.8.5 Modelos de arquitetura formais

Abowd et al. [1] afirmam que os estilos arquitetônicos podem ser descritos formalmente em termos de um pequeno conjunto de mapeamentos do domínio sintático de descrições arquitetônicas (diagramas de caixa e linha) para o domínio semântico do significado arquitetônico. No entanto, isso pressupõe que a arquitetura é a descrição, em vez de uma abstração de um sistema em execução. Inverardi e Wolf [65] usam o formalismo Chemical Abstract Machine (CHAM) para modelar elementos de arquitetura de software como produtos químicos cujas reações são controladas por regras explicitamente estabelecidas. Ele especifica o comportamento dos componentes de acordo com a maneira como eles transformam os elementos de dados disponíveis e usa regras de composição para propagar as transformações individuais em um resultado geral do sistema. Embora este seja um modelo interessante, não está claro como o CHAM poderia ser usado para descrever qualquer forma de arquitetura cujo propósito vai além de transformar um fluxo de dados. Rapide [78] é uma linguagem de simulação concorrente baseada em eventos projetada especificamente para definir e simular arquiteturas de sistema. O simulador produz um conjunto parcialmente ordenado de eventos que podem ser analisados ​​quanto à conformidade com as restrições arquitetônicas na interconexão. Le Métayer [75] apresenta um formalismo para a definição de arquiteturas em termos de gráficos e gramáticas de grafos.

1.9 Resumo

Este capítulo examinou os antecedentes desta dissertação. Introduzir e formalizar um conjunto consistente de terminologia para conceitos de arquitetura de software é necessário para evitar a confusão entre arquitetura e descrição de arquitetura que é comum na literatura, particularmente porque muitas das pesquisas anteriores sobre arquitetura excluem dados como um elemento arquitetônico importante. Concluí com um levantamento de outras pesquisas relacionadas à arquitetura de software e estilos arquitetônicos.

Os próximos dois capítulos continuam nossa discussão sobre o material de fundo, concentrando-se em arquiteturas de aplicativos baseadas em rede e descrevendo como os estilos podem ser usados ​​para orientar seu projeto arquitetônico, seguido por uma pesquisa de estilos arquitetônicos comuns usando uma metodologia de classificação que destaca as propriedades arquitetônicas induzidas quando os estilos são aplicados a uma arquitetura para hipermídia baseada em rede.