2 Fundamentos

Neste capítulo são apresentados os fundamentos que serviram de base para a elaboração e construção deste trabalho. Inicialmente, será apresentada uma visão geral dos tipos de ferramentas normalmente utilizadas durante o ciclo de vida de um software e suas principais características. Em seguida, apresentaremos um breve resumo do que é a Web Semântica e quais das suas tecnologias suportam a abordagem de extração apresentada.

2.1. Visão Geral das Ferramentas utilizadas no Ciclo de Vida de Desenvolvimento de Software

O ciclo de desenvolvimento de software é o processo pelo qual os sistemas de informação são criados ou modificados. Durante este processo, os envolvidos normalmente lidam com um grande conjunto de ferramentas que os auxiliam na execução de suas tarefas[9]. Dentre estas ferramentas, podemos destacar as IDEs (*Integrated Development Envoriment*) ou Ambiente de Desenvolvimento Integrado, as ferramentas de Construção, as ferramentas de Controle de Versão, as ferramentas para a automação de Testes, as ferramentas de Integração Contínua e os Sistemas de Acompanhamento de Defeitos e Demandas. Como mencionamos anteriormente, estas ferramentas armazenam grande parte dos artefatos que são gerados durante a vida de um projeto. Na Figura 1 podemos observar como as informações fluem entre estas ferramentas. Abaixo descrevemos suas principais características:

IDE (Integrated Development Envoriment)[10]: São softwares que facilitam atividades relacionadas com a codificação de programas de software. Normalmente, fornecem um editor de código-fonte, ferramentas para a elaboração

[9] https://en.wikipedia.org/wiki/Programming_tool
[10] https://en.wikipedia.org/wiki/Integrated_Development_Environment

e execução do build e formas de depurar a execução do código. Exemplos de IDEs são o Eclipse[11], o IntelliJ[12] e o Microsoft Visual Studio[13].

Ferramentas de Construção[14]: São ferramentas que auxiliam a criação de roteiros para automatizar algumas das tarefas que os desenvolvedores executam no decorrer do projeto, como a compilação do código-fonte, o empacotamento de executáveis gerados, a execução de testes e a implantação em ambientes de promoção (CLARK, 2004). Exemplos destes tipos de ferramentas são o Make[15], Apache Ant[16] e o Apache Maven.

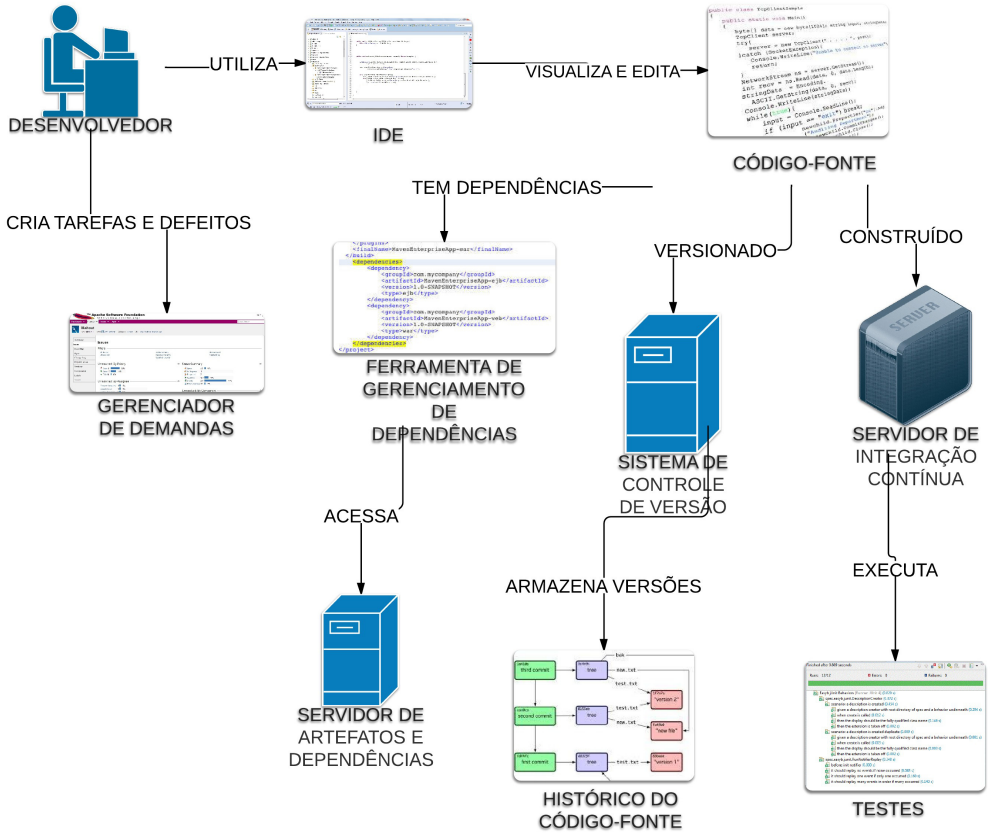

Figura 1 Exemplo do Fluxo de Informações entre ferramentas utilizadas no desenvolvimento de software

[11] http://www.eclipse.org/
[12] http://www.jetbrains.com/idea/
[13] http://www.visualstudio.com/
[14] https://en.wikipedia.org/wiki/Build_automation
[15] http://en.wikipedia.org/wiki/Make_(software)
[16] http://ant.apache.org/

Gerenciadores de Dependência[17]: São ferramentas que gerenciam as dependências externas e internas de um projeto. Elas são responsáveis por acessar repositórios de artefatos para recuperar os arquivos físicos que representam estas dependências e por verificar se uma determinada dependência é compatível com as outras encontradas no projeto. Exemplos destes tipos de ferramentas são o Apache Maven, o SBT[18] e o Apache Ivy.

Sistemas de Controle de Versão[19]: São sistemas que gerenciam as diversas versões do código-fonte desenvolvido e fornecem mecanismos para que os desenvolvedores trabalhem de forma paralela. Além disso, são utilizados para coordenar a liberação de um novo release ou a correção de algum código com problemas. Exemplos destas ferramentas são o CVS[20], Subversion[21] e o Git[22].

Ferramentas de Automação de Testes[23]: Estas ferramentas auxiliam os desenvolvedores na confecção de testes para verificar a corretude do código desenvolvido, o desempenho do sistema ou para testar a integração entre seus diversos componentes. A execução destes testes normalmente é feita através de roteiros criados com o auxílio de ferramentas de construção utilizadas no projeto. Alguns exemplos de ferramentas de automação de testes são o TestNG[24], o JUnit[25], o Arquillian[26] e o PaxExam[27].

Sistemas de Acompanhamento de Defeitos e Demandas[28]: São ferramentas utilizadas na comunicação entre os membros da equipe, pois permitem o cadastro das tarefas e auxiliam na organização do trabalho que deve ser realizado durante o desenvolvimento do sistema. Além disso, são utilizadas para registro dos defeitos

[17] http://en.wikipedia.org/wiki/List_of_software_package_management_systems#Application-level_package_managers
[18] http://www.scala-sbt.org/
[19] https://en.wikipedia.org/wiki/Revision_control
[20] http://www.nongnu.org/cvs/
[21] http://subversion.apache.org/
[22] http://git-scm.com/
[23] https://en.wikipedia.org/wiki/Test_automation
[24] http://testng.org/doc/index.html
[25] http://junit.org/
[26] http://arquillian.org/
[27] https://ops4j1.jira.com/wiki/display/PAXEXAM3/Pax+Exam
[28] https://en.wikipedia.org/wiki/Bug_tracking_system

encontrados. Exemplos de ferramentas são o Atlassian Jira[29], o Bugzilla[30] e o Trac[31].

Ferramentas de Integração Contínua[32]: Como mencionado anteriormente, a prática de Integração Contínua promove a integração frequente do código e construções automatizadas. Este tipo de ferramenta é responsável por automatizar a construção do sistema através da execução periódica de mecanismos que recuperam a última versão do código-fonte, executam os roteiros de construção e os testes automatizados. Exemplos de ferramentas são o Hudson[33], Jenkins[34] e o CruiseControl[35].

2.2. Tecnologias da Web Semântica

O termo "Web Semântica" foi cunhado em 2001 por Tim Berners-Lee, o inventor da World Wide Web, para designar uma "web de dados que podem ser processados diretamente e indiretamente por máquinas" (BERNERS-LEE, HENDLER, LASSILA, 2001). Esta nova Web é necessária pelo fato da Web tradicional não ter sido concebida para ser compreendida por computadores, mas sim por pessoas. Por isso, a Web Semântica visa estender a estrutura tradicional das páginas web para inserir metadados sobre as próprias páginas e sobre como elas estão relacionadas com outras entidades. Estes metadados fornecem informações estruturadas e permitem que agentes automatizados possam navegar pela Web e executar tarefas que normalmente só os seres humanos seriam capazes de realizar por conta das limitações da web tradicional (SEGARAN, EVANS, TAYLOR, 2009). Um exemplo seria pesquisar todos os voos que saem do Aeroporto conhecido popularmente como Galeão antes das 19 horas da véspera de Natal de 2013.

[29] https://www.atlassian.com/software/jira
[30] http://www.bugzilla.org/
[31] http://trac.edgewall.org/
[32] https://en.wikipedia.org/wiki/Continuous_integration
[33] http://hudson-ci.org/
[34] http://jenkins-ci.org/
[35] http://cruisecontrol.sourceforge.net/

Com o intuito de fornecer mecanismos para estruturar as informações presentes em fontes não estruturadas, a Web Semântica faz uso de formatos e tecnologias que compõem sua base. Dentre estas, podemos destacar:

- Resource Description Framework (RDF)
- RDF Schema (RDFS)
- SPARQL
- Web Ontology Language (OWL)

2.2.1. Resource Description Framework (RDF)

O Resource Description Framework (RDF)[36] é uma especificação da W3C que foi originalmente pensada para funcionar como um modelo de dados para metadados. Este modelo baseia-se na ideia de asserções do tipo sujeito-predicado-objeto. O primeiro elemento desta tripla denota um recurso, isto é, qualquer entidade que pode ser identificada, nomeada ou localizada. O predicado denota traços e aspectos do sujeito da tripla, ou seja, o tipo de relacionamento entre o sujeito e o objeto. Já o terceiro termo, o objeto, é a valoração desta característica e pode apresentar-se como um valor literal ou como o identificador de outro recurso.

O modelo de dados RDF possui diversas similaridades com outras formas de representação, como o modelo ER (Entidades e Relacionamentos) para modelar banco de dados relacionais ou o diagrama de classes da UML para modelar sistemas orientados a objetos[37]. Ambos fornecem meios para representar relacionamentos entre uma entidade e suas propriedades, e entre uma entidade e outra entidade. Abaixo um exemplo da sentença "O aluno Carlos cursa a disciplina INF0000" utilizando as notações das três modelagens:

[36] http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
[37] http://en.wikipedia.org/wiki/Resource_Description_Framework#Overvie0077

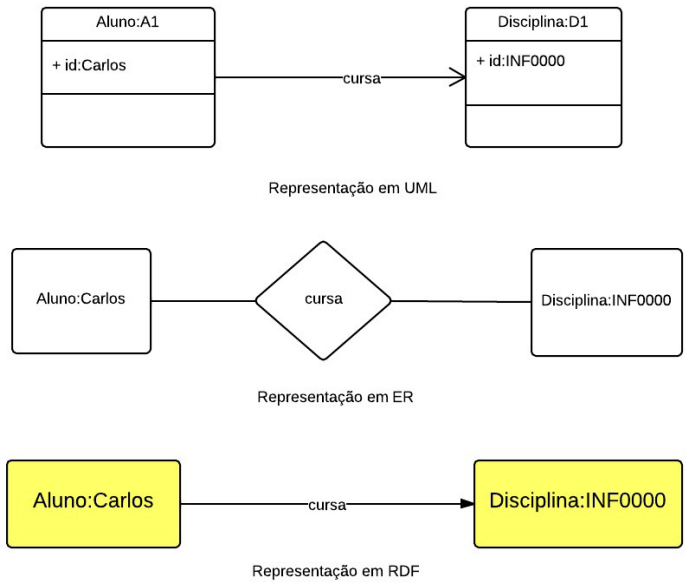

Figura 2 Exemplos de Representação de uma mesma informação nas três modelagens

Devido à simplicidade e flexibilidade de seu modelo de representação, RDF passou a ser utilizado na descrição de informações sobre recursos na Web. Para isto, cada recurso dentro de uma tripla RDF é identificado através de um URI (Universal Resource Identifier) ou um URL (Universal Resource Locator)[38], por exemplo, o endereço de um web site ou um endereço de e-mail são URLs válidas para representear um recurso. Isto permite que sejam ditos fatos sobre qualquer entidade na Web, incluindo coisas que podem não ser diretamente recuperadas, como pessoas ou objetos físicos. Como resultado, além de descrever coisas como páginas Web, RDF pode descrever carros, negócios, pessoas, notícias e eventos, desde que estes possuam um identificador ou localizador na web. Além disso, como o valor de uma propriedade em uma tripla pode ser a URI ou URL de outro recurso, um conjunto de triplas RDF pode ser compreendido como um multigrafo de recursos com suas propriedades e valores. Portanto, RDF pode ser considerado um dos blocos principais da Web Semântica, no sentido em que permite tornar a Web Tradicional em algo compreensível por máquinas através da identificação e caracterização das entidades presentes na mesma.

[38] http://en.wikipedia.org/wiki/Uniform_resource_locator

Além de definir como deve ser estruturado um modelo RDF válido, a sua especificação define como um conjunto de triplas RDF pode ser serializado, isto é, armazenado. A primeira notação criada para isto foi uma sintaxe baseada em XML conhecida como RDF/XML[39]. Após este, outros formatos foram desenvolvidos, mas cada qual com um objetivo e sintaxe distintos, sendo que alguns foram especificamente criados para facilitar a leitura e manipulação humana, como a Notation3 (N3)[40] ou a notação Turtle[41]. Abaixo um exemplo de representação em RDF/XML das informações de uma pessoa chamada Eric Miller:

```
    <?xml version="1.0"?>
    <rdf:RDF   xmlns:rdf="http://www.w3.org/1999/02/22-
rdf-syntax-ns#"
    xmlns:contact="http://www.w3.org/2000/10/swap/pim/c
ontact#">
    <contact:Person
rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
    </contact:Person>
    </rdf:RDF>
```

Figura 3 Exemplo da Notação RDF/XML representando as informações de pessoa Eric Miller

2.2.2. RDF Schema (RDFS) [42]

Como dito na seção anterior, as informações sobre uma determinada entidade, quando modelada em RDF, são descritas por meio de triplas. Entretanto, o RDF por si só não fornece mecanismos que permitam definir o domínio do conhecimento ao qual pertence um determinado conjunto de triplas. Esta

[39] http://www.w3.org/TR/REC-rdf-syntax/
[40] http://www.w3.org/TeamSubmission/n3/
[41] http://www.w3.org/TR/turtle/
[42] http://www.w3.org/TR/rdf-schema/

informação passou a ser representada através de extensões do RDF fornecidas pela Linguagem de Descrição de Vocabulários RDF ou RDF Schema (RDFS). O RDFS não fornece um vocabulário de classes sobre um domínio específico, ao invés disso, disponibiliza os meios necessários para descrever estas classes e propriedades, e para indicar quais classes e propriedades espera-se que sejam utilizadas em conjunto. Um exemplo disso é a propriedade "foaf:name" ter a restrição de somente poder ser utilizada para descrever o nome de entidade do tipo "foaf:Agent". Em outras palavras, o RDFS fornece um sistema de tipos para RDF, similar em alguns aspectos aos sistemas de tipos presentes em linguagens de programação orientadas a objetos, como Java. Por exemplo, o RDFS permite que os recursos sejam definidos como instâncias de uma ou mais classes, além de possibilitar que as classes sejam organizadas hierarquicamente; por exemplo, a classe ex:Cachorro pode ser definida como subclasse de ex:Mamifero que é subclasse de ex:Animal, significando que qualquer recursos que é da classe ex:Cachorro também é, implicitamente, da classe ex:Animal. Abaixo alguns elementos definidos no RDFS:

Classe	**Descrição**
rdfs:Resource	É a classe de todas as coisas descritas em RDF.
rdfs:Class	É a classe que define que um recurso pertence a um determinado tipo. Sendo que rdfs:Class é uma definição recursiva, pois rdfs:Class é do tipo rdfs:Class.
rdfs:Property	Classe que é o supertipo de todos os recursos que definem propriedades.
Propriedade	**Descrição**
rdfs:range	Esta propriedade é utilizada para exprimir que um recurso R deve ser do tipo C, caso o recurso esteja associado com uma propriedade P que tenha seu

	valor rdfs:range igual à C.
rdfs:domain	Esta propriedade é utilizada para exprimir que um recurso O deve ser do tipo D, caso o recurso esteja associado na forma de objeto com uma propriedade P que tenha seu valor rdfs:domain igual à D.
rdf:type	Propriedade que define que um recurso é uma instância de uma determinada classe RDF.
rdfs:subClassOf	Propriedade que permite definir uma hierarquia de classes.

Tabela 1 Principais elementos do RDFS

2.2.3. SPARQL

SPARQL[43] é o acrônimo recursivo para SPARQL Protocol and RDF Query Language. Esta recomendação da W3C define uma linguagem de consulta para dados em RDF. Uma típica consulta nesta linguagem consiste de padrões de triplas, conjunções, disjunções e padrões opcionais. Atualmente existem quatro diferentes tipos de consultas SPARQL:

- **SELECT Query:** utilizada para extrair informações em seu estado natural a partir de um ponto de acesso SPARQL. Os resultados são retornados de forma tabular.
- **CONSTRUCT Query:** utilizada para extrair informações de um ponto de acesso e transformá-los em RDF válido.
- **ASK Query:** utilizada para prover um resultado do tipo verdadeiro ou falso para uma consulta executada em um ponto de acesso.
- **DESCRIBE Query:** utilizada para extrair um grafo RDF de exemplo a partir de um ponto de acesso SPARQL. O conteúdo desta informação é deixado a critério do ponto de acesso utilizado.

[43] http://www.w3.org/TR/sparql11-query/

Abaixo apresentamos uma consulta SPARQL do tipo SELECT para retornar o título do livro identificado pelo URI http://example.org/book/book1.

```
SELECT ?title
WHERE
{
<http://example.org/book/book1>
<http://purl.org/dc/elements/1.1/title>
?title .
}
```

Figura 4 Exemplo de uma consulta SPARQL do tipo SELECT

Além de uma linguagem, esta especificação define um protocolo para a execução de consultas em serviços de processamento SPARQL. Este protocolo define o padrão de comunicação entre uma aplicação cliente e um ponto de acesso, ou seja, qual conjunto de operações suportadas, como elas devem ser acessadas e qual o conjunto de códigos de retornos que podem ser apresentados para uma determinada operação. Em sua versão mais atual, este protocolo define dois tipos principais de operações: *query* e *upate*. O primeiro tipo é utilizado para enviar uma consulta SPARQL para um determinado ponto de acesso e depois receber seu resultado. Já a segunda operação é utilizada quando é necessário executar uma operação de atualização sobre os dados de um ponto de acesso SPARQL. Pelo fato do protocolo SPARQL ter sido construído para utilizar o protocolo HTTP como base, qualquer uma destas duas operações deve seguir este padrão, o que inclui a utilização de códigos de sucesso e falha conforme definidos neste padrão, como por exemplo, o código 400 para indicar que a consulta fornecida não é uma consulta SPARQL válida.

2.2.4.Web Ontology Language (OWL) [44]

A Web Ontology Language ou OWL é uma família de linguagem de representação do conhecimento para a autoria de ontologias. Os dados descritos por uma ontologia em OWL são interpretados como um conjunto de indivíduos e de asserções sobre suas propriedades, as quais podem relacionar estes indivíduos entre si. Ou seja, neste contexto, uma ontologia pode ser compreendida como um método formal de representação do conhecimento que permite definir um conjunto axiomas sobre um determinado domínio.

Todo conhecimento representado em OWL utiliza um pressuposto conhecido como "*open world assumption*" [45], onde se uma afirmação não pode ser provada como verdadeira, ela é assumida como indefinida. Este tipo de tratamento é o oposto de linguagens como SQL e Prolog[46], onde a informação desconhecida é sempre assumida como falsa. Atualmente existem três variantes da linguagem OWL, cada uma com um nível diferente de expressividade lógica. O primeiro nível é representado pelo OWL Lite, o segundo pelo OWL DL e o terceiro pelo OWL Full. Cada uma destas sublinguagens é uma extensão sintática de sua predecessora, por este motivo qualquer ontologia válida escrita em uma sublinguagem de mais baixo nível também pode ser escrita e será válida em uma sublinguagem de mais alto nível, sendo que o contrário não é válido. Abaixo vamos descrever sucintamente cada um destes tipos:

OWL Lite: foi originalmente criada para suportar a criação de hierarquias de classificação e para suportar alguns tipos de restrições no modelo, como cardinalidade.

OWL DL: é fundamentada na lógica descritiva e foi criada para prover o máximo de expressividade enquanto mantém completeza computacional, decidibilidade e funcionamento prático nos algoritmos de raciocínio. Diferente de OWL Lite, OWL DL inclui todos os construtos possíveis de OWL, mas com a restrição de que estes só podem ser utilizados sob determinadas condições.

[44] http://www.w3.org/TR/owl2-primer/
[45] http://semanticweb.com/introduction-to-open-world-assumption-vs-closed-world-assumption_b33688
[46] http://en.wikipedia.org/wiki/Prolog

OWL Full: é baseada em semânticas diferente da OWL Lite ou OWL DL e foi criada para manter a compatibilidade com RDFS em algum de seus pontos, como por exemplo, a possibilidade uma classe em RDFS poder ser tratada simultaneamente como o conjunto de indivíduos daquele tipo ou como um indivíduo propriamente dito, ou seja, o próprio recurso que representa a classe. O fato de OWL Full permitir este tipo de asserção, que incrementa o significado pré-definido de um vocabulário, faz com que qualquer ontologia escrita nesta linguagem seja indecidível, ou seja, nenhum raciocinador é capaz de realizar o raciocínio completo da mesma.

Abaixo apresentamos os principais elementos de uma ontologia escrita em OWL:

Instâncias: Uma instância é um objeto que corresponde a um Indivíduo na lógica descritiva.

Classes: Uma classe é uma coleção de objetos e corresponde a um conceito na lógica descritiva. Uma classe pode conter um ou mais indivíduos ou instâncias. Sendo que uma instância pode estar associada a múltiplas classes ou a nenhuma classe. Outra característica é que uma classe pode ser subclasse de outras classes, fazendo com que ela herde as características das mesmas.

Propriedades: Uma propriedade é uma relação binária que especifica características da instância de uma classe. Pode tanto ligar uma instância com um valor literal, no caso de ser do tipo *datatype property*, ou então ligar uma instância a outra, quando é uma *object property*. Propriedades também podem ter diversas classificações lógicas, como transitividade, simetria, propriedade inversa e propriedade funcional, além de ser possível definir restrições para os tipos das instâncias as quais ela se associa.

Datatype Properties: são propriedades que representam relações entre instâncias de classes e literais RDF ou tipos de dados do esquema XML.

Object properties: representam relações entre instâncias de duas classes.

Operadores: OWL suporta a aplicação de vários tipos de operações de conjuntos sobre classes, como união, interseção e complemento. Além disso, também é possível definir classes a partir da enumeração de seus indivíduos ou a partir da disjunção de seus membros.