# Proposta de um Sistema para Compreensão de Aplicações Web

# Resumo

Desenvolver software para a web é sempre um desafio dada a diversidade de tecnologias e linguagens que se interligam neste tipo de aplicações. Assim, a tarefa de compreender este tipo de software é bastante complexa. A área de Compreensão de Programas (CP) já disponibiliza técnicas e ferramentas para aplicações ditas "tradicionais", no entanto para as Aplicações Web, ainda não existem ferramentas que transfiram o mesmo nível de conhecimento, em termos estáticos, dinâmicos e comportamentais. Este artigo tem como objectivo propor um sistema de visualização que mostre as características das aplicações Web, nomeadamente a geração dinâmica de páginas e relações de dependência entre ficheiros e páginas, permitindo deste modo compreender detalhadamente a aplicação em toda a sua extensão e profundidade com vista a efectivamente ajudar em qualquer tarefa de manutenção e reengenharia. A abordagem proposta é baseado em modelos cognitivos e técnicas de visualização de Software.



**Palavras chave:** aplicações web, compreensão de programas.


## 1. Introdução

Com o crescimento exponencial de aplicações projectadas para utilização através de um *browser*, muitas tecnologias foram desenvolvidas para melhorar o desempenho e funcionalidades das Aplicações *Web* (AW). A importância das ferramentas de compreensão de AW advém da capacidade de traduzir a sua estrutura/tecnologias/paradigmas em representações visuais, organizadas e facilmente exploráveis [Storey]. Em [XXX], foi feito um estudo que passou por caracterizar aplicações Web, nomeadamente os componentes/relacionamentos que dela fazem parte, bem como enumerar as características que em geral uma ferramenta de compreensão de programas deve ter para atingir esse objectivo, distinguindo-se o uso de vistas, facilidade na navegação, interactividade, contextualização, apresentação e animação. Neste artigo iremos enumerar os domínios de conhecimento essenciais ao entendimento de aplicações Web, baseados em modelos cognitivos e nas características deste tipo de programa e suas relações [Fracchia, Muller, Storey]. Seguindo estes modelos, juntamente com a aplicação de técnicas de visualização de software reconhecidas como auxiliares num sistema de compreensão de programas [Storey], parece haver um cruzamento de interesses que achamos fundamental ao entendimento deste tipo de aplicações com características próprias e mecanismos envolvidos. Seguidamente, iremos propor tipos de representações que satisfazem esses requisitos. Estas representações fazem parte de uma tese de mestrado [XXX] no entanto apenas algumas irão ser apresentadas. Assim, apresentaremos as que de alguma forma, abarcam os conceitos que queremos aqui transmitir. Á laia de conclusão, queremos também destacar que as representações foram analisadas por programadores Web, embora a analise tenha sido meramente informativa para os autores, pois não foi sistemática nem foram analisados formalmente os resultados dessa análise.

### Domínios de Conhecimento de uma Aplicação Web

Compreender uma AW, passa por compreender, tal como nas aplicações "Clássicas" a sua representação estática, dinâmica e comportamental [Holt and Ahmed], mas também passa por compreender a funcionalidade dos componentes que a constituem, as tecnologias associadas, linguagem e paradigma de programação. Assim, e de acordo com modelos cognitivos de

compreensão de programas, existem quatro domínios de conhecimento que é preciso explorar para retirar a informação que interessa representar. São eles:

- Análise do Domínio do Problema: A análise do domínio do problema tem em vista recuperar toda a informação possível acerca do problema tratado pela aplicação, quer através de documentação, quer através de entrevistas a utilizadores da aplicação, ou mesmo através de testes ao software. É portanto toda a informação relativa às regras de negócio da aplicação, as entidades que interagem e relações entre elas.
- Análise do Domínio do Programa: A análise do domínio do programa engloba, tanto o conhecimento do paradigma, da linguagem de programação e das tecnologias que fazem parte da aplicação e conhecimento do funcionamento dos componentes da aplicação.
- Análise da Representação Estática e Dinâmica: A análise da representação estática, envolve o estudo do fluxo de dados e de controlo, o conhecimento acerca das variáveis (tipos e dependências), das funções, objectos ou procedimentos, e dependências de ficheiros. Relativamente à análise dinâmica não se pode fazer a mesma abordagem que se segue tipicamente no âmbito das aplicações ditas "*Clássicas*", pois o fluxo de controlo da aplicação não depende exclusivamente da aplicação, mas também depende das acções de navegação do utilizador que podem alterar o fluxo da aplicação.
- Análise Comportamental: Esta representação relaciona a análise do domínio do problema como a análise da representação estática e dinâmica. Sabendo as soluções que existem para o problema que a aplicação se propõe resolver, queremos por um lado relacioná-la com o código que a implementa e visualizar todo o percurso de execução e, por outro lado, dada uma operação que a aplicação efectue, visualizar o seu percurso pelos componentes que a realizaram. Um exemplo de representações deste tipo de análise são os chamados *Use-Cases* de UML, que representam como a aplicação se comporta nas várias situações que podem ocorrer durante a sua execução, já usados noutras ferramentas cujo propósito é perceber o funcionamento da aplicação [Di Lucca]. Durante uma operação específica teremos todos os passos até que essa operação esteja finalizada.

Achamos que através do conhecimento destes quatro domínios de conhecimento, conseguimos abranger todo um conjunto de factores que embora externos ao programa, deles dependem a compreensão efectiva de uma AW.

## 2. Requisitos de um Sistema de Visualização para Compreensão de Aplicações Web

Os requisitos de um sistema de visualização obtêm-se do nosso ponto de vista, através de cruzamentos entre as características essenciais numa ferramenta de compreensão, e entre os domínios de conhecimento que as AW exigem que se conheçam. Embora as questões relacionadas com a visualização de software, sejam válidas para as aplicações clássicas, neste artigo iremos contextualiza-las no dominío da Web.

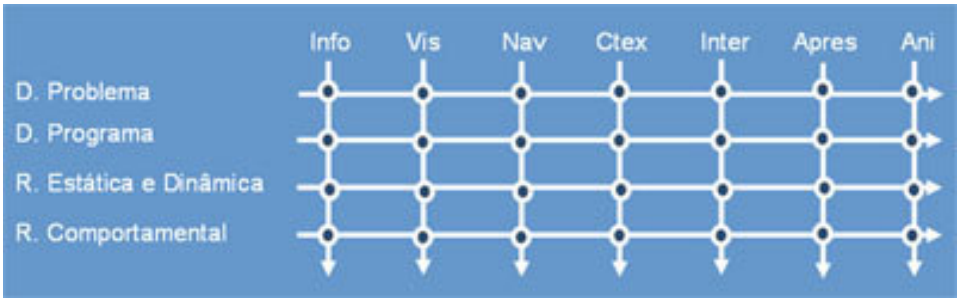


Figura 1- Requisitos de Ferramentas de Compreensão Web

A Figura 1 representa, através das suas colunas (Info - Informação, Vis - Vistas, Nav - Navegação, Ctex - Contextualização, Inter - Interactividade, Apres - Apresentação e Ani - Animação), as características que uma ferramenta de visualização deve englobar, e das linhas (Análise do Domínio do Problema, Análise do Domínio do Programa, Representação Estática e Dinâmica e Representação Comportamental) a informação que queremos analisar para compreender uma AW.

Aplicando os domínio do conhecimento à Web, iremos descrever que tipo de representações queremos ver representadas, numa ferramenta. Os parâmetros a testar o **domínio do problema**, são a representação dos conceitos/entidades, relações e regras de funcionamento (regras de negócio). A avaliação do **domínio do programa**, requer informação relativa aos paradigmas, linguagens e tecnologias associadas à aplicação. A **análise da representação estática e dinâmica** nestas aplicações deve ser efectuada em conjunto, e os critérios incidirão sobre as representações do fluxo de controlo e de dados, de páginas estáticas e dinâmicas e o seu mecanismo de geração dinâmica de páginas; Note-se que sobre uma página estática deverá ser feita uma análise estática e dinâmica, bem como sobre as páginas dinâmicas. Também se deve analisar se existem representações relativas à interacção entre as linguagens de programação usadas e entre ficheiros de código / páginas; Devem ser avaliadas as visualizações que representem relações entre componentes e de documentos gerados no Cliente com o ficheiro que o criou. Finalmente, a **avaliação da análise comportamental** será avaliar de que forma são representados comportamentos, i.e., para cada funcionalidade do sistema, determinar se podemos ver representados todos os percursos possíveis até que esta seja considerada como finalizada. Os parâmetros a avaliar nesta dimensão são relacionados com a representação dos comportamentos da aplicação relativamente às suas funcionalidades.

## 3. Proposta de Representações Visuais

Já se falou do que se deve mostrar numa ferramenta de CP para AW, no entanto, ainda não existem representações que ilustrem a informação que achamos ser essencial para as compreender. Nesta secção iremos propor algumas representações dos componentes de uma AW bem como do Mecanismo de Geração Dinâmica de Páginas.

### Domínio do Problema

A visualização desta dimensão do conhecimento é um passo importante para a construção de um modelo mental. A representação do domínio do problema deve passar pela visualização do dicionário de dados da aplicação, quando este existe. No caso de não ser possível, propõe-se a

visualização de informação extraída da base de dados da aplicação - os meta dados das tabelas. Também se propõe a apresentação de um mapa de conceitos, extraído a partir da base de dados, no entanto, o sistema deverá interagir com o utilizador [1] para que este possa introduzir manualmente os significados. O sistema deverá permitir, para cada campo de cada tabela, introduzir o respectivo significado - também pode agrupar conceitos e associar-lhes uma imagem. Para esta introdução de dados, propomos a apresentação de um ecrã como na Figura 2, em que para cada campo da tabela, o utilizador possa introduzir um texto informativo - um significado - possibilitando agrupar conceitos e associá-los com uma imagem, se assim pretender. Um mapa de conceitos mostrará o resultado desta associação.


Figura 2 - Introdução de Significados

## Domínio do Programa

O domínio do programa irá passar pela representação do paradigma, tecnologias e linguagens que implementam a aplicação. A nossa proposta para representar o **paradigma** passa por mostrar um conjunto de Clientes, com ligação a um Servidor que por sua vez se encontra relacionado com uma base de dados e um sistema de ficheiros. Para saber mais detalhadamente o que se passa em cada um destes componentes deve poder-se seleccionar cada um e através de navegação descer um nível de abstracção, detalhando por exemplo no caso do servidor, os ficheiros que compõem a aplicação, e ainda sobre estes ser possível descer para outro tipo de representações - em última instância até ao código. Ainda no caso do servidor, deve existir informação acerca do tipo de requisitos que a aplicação necessita que estejam instalados para que este execute a aplicação, bem como informação acerca do relacionamento deste com os seus ficheiros e ainda a sua relação com o servidor de base de dados - no âmbito da aplicação. A nossa proposta está exibida na Figura 3 em para cada um dos elementos representados devem existir etiquetas informativas sobre as suas funções e relacionamento com o servidor.

[1] Para a introdução dos significados considera-se que o utilizador seja o programador que desenvolveu a aplicação.

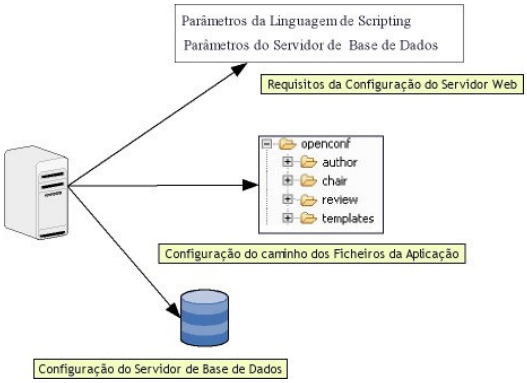


Figura 3 - Representação do Servidor

A representação da base de dados pode ser feita segundo vários tipos de diagramas, no entanto deve sempre verificar-se a possibilidade de se visualizarem apenas as tabelas que se pretende, e sempre com a possibilidade de ser descer em níveis de abstracção que permitam ir à estrutura de uma tabela, através de navegação. Relativamente aos ficheiros de uma aplicação, deverá ser possível visualizá-los numa estrutura em árvore com directorias e ficheiros e descer até ao código, se assim pretendermos. Seria útil a representação de tipos de ficheiros diferenciados com cores, distinguindo ficheiros com extensão *HTML, PHP, CSS, GIF e SQL.*

Actualmente, em que as **tecnologias** do lado do cliente estão em grande desenvolvimento, consideramos para já o uso de sessões e de *CSS*. Estas devem permitir, tal como nas outras representações, navegar para níveis mais detalhados, possibilitando seguir a visualização até aos ficheiros fonte. Parece útil detectar, no caso das *CSS*, se estas estão num ficheiro externo ou se estão inseridas na própria página Cliente, e de acordo com a situação, diferenciar a representação. Temos consciência que estas representações, apenas servem para alguém, cuja experiência em desenvolvimento *web* é nula, ou quase nula, pelo que são apenas uma ajuda na compreensão deste mecanismo, em que se apresentam os componentes envolvidos e as suas relações.

Relativamente à **linguagem**, a informação útil que propomos disponibilizar é a apresentação de um texto informativo geral, das principais características da linguagem, nomeadamente sobre o seu paradigma - lógico, funcional ou objectos.

## Análise Estática e Dinâmica

A representação dos componentes de uma AW passa por mostrar as páginas estáticas, dinâmicas, e suas relações. Acreditamos que se se compreender como são obtidas as páginas e de que forma se relacionam, é possível levantar hipóteses acerca dos dados que circulam e quando circulam. Desta forma achamos que se criam pistas, que segundo os autores dos modelos cognitivos, são um auxiliar bastante importante num sistema de compreensão.

### Página Estática

Uma página estática é considerada, como um ficheiro que contém apenas código interpretado pelo browser. Nas páginas estáticas, e para destacar a informação que interfere no fluxo da aplicação, estão representados os links e os campos dos formulários. Pelo que propomos cores e

formas diferentes para representar links com outras páginas, campos de formulários, links externos e links para emails.

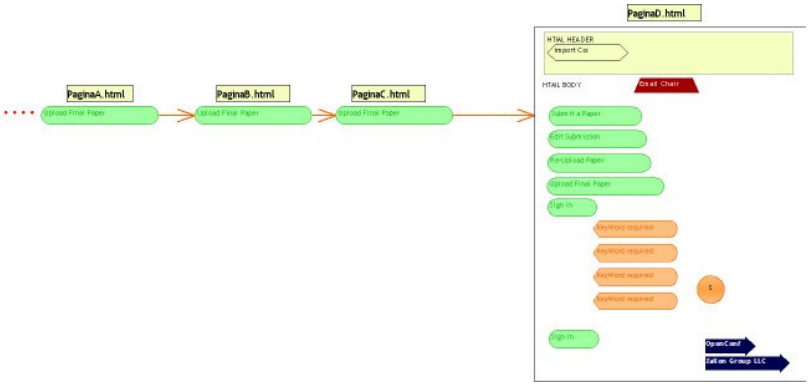


Figura 4 - Caminho entre duas Páginas Estáticas: Vista 1

Também consideramos ser útil a divisão da página em *header* e *body* para reflectir a estrutura de uma página *HTML*. Esta representação respeita a forma da página, pois sendo o *HTML* uma linguagem essencialmente de formatação de informação, importa respeitar a sua forma para um melhor entendimento. Propõe-se que seja possível navegar no sistema, seguindo os *links* apresentados. A página destino do *link* deve ser, então, mostrada; a original deve diminuir de tamanho ficando apenas visível o *link* seleccionado. Desta forma teremos uma representação natural do caminho despoletado pelo *link* em estudo. Existe porém uma condicionante espacial, pois é tecnicamente impossível representar infinitamente esta cadeia de antecessores; pelo que se propõe a representação de no máximo quatro níveis. Esta representação também é uma maneira de apresentar um dos tipos de relações que existem entre páginas Cliente, a relação "*liga a*". A representação dos antecessores permite o levantamento de hipóteses acerca do fluxo de dados e de controlo da aplicação. A sequência de passos até uma página, bem como a observação das ligações alternativas, são pistas que ajudam a compreender o fluxo e a informação que circula até à página destino [XXX].

Todas as páginas estáticas possuem relações que necessitam ser analisadas para a sua compreensão. Em resumo teremos as relações entre páginas Cliente são: *"depende de", "liga a", "é gerado por"* e *"é acedida por"* , e a nossa proposta para a sua representação está a Figura 6. É importante que a partir desta representação se possa passar directamente para a análise de cada um dos elementos nela representados, i.e., se quisermos analisar mais detalhadamente o ficheiro que a gerou, deve ser possível a selecção da página. A forma de passagem de uma representação para a outra (Navegação) não pode ser descuidada, pelo que se pode aplicar por exemplo a técnica de navegação *"pan and zoom"* aplicada noutras ferramentas, nomeadamente na ferramenta SHRimP [Lintern Michaud and Storey].

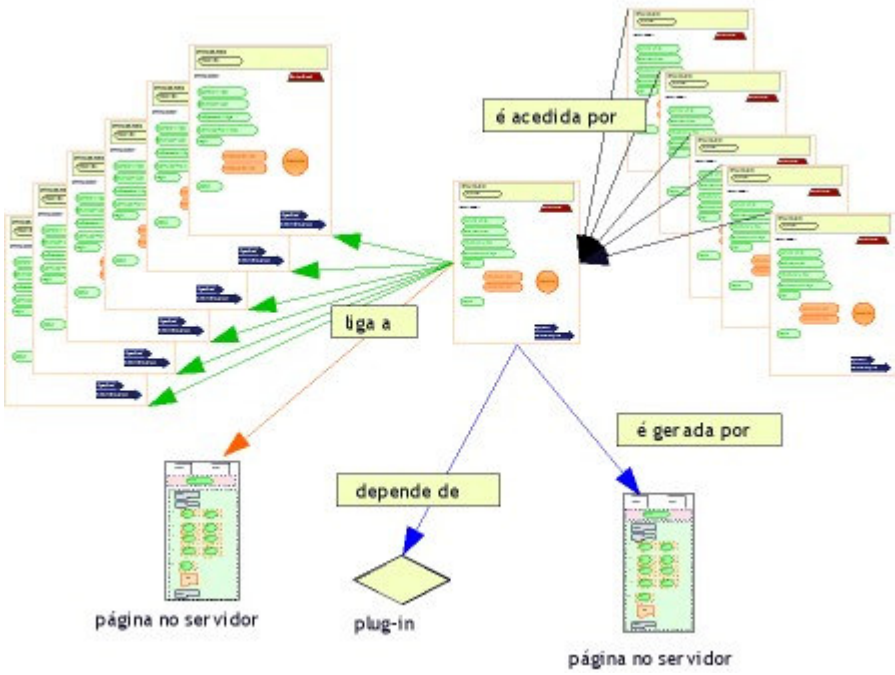


Figura 5 - Página Estática: Relações – Vista 1

Página Dinâmica

As páginas dinâmicas podem ser de dois tipos, *Tipo 2* (T2) – código interpretado pelo browser e código a ser processado no Servidor - e *Tipo 3* (T3) – código a ser processado no Servidor. A representação destas páginas obedece a critérios que permitem distinguir qualquer linha de código passível de ser enviado ao cliente, distinguindo-se também a forma como é produzido conteúdo para o cliente (se através de código processado no Servidor ou através de código interpretado no Cliente). Os blocos HTML, serão destacados dos blocos de código processado no Servidor, bem como os pedaços de código que pura e simplesmente processam código mas não enviam directamente código ao Cliente. A representação desta página está na incluída na Figura 5.

Mecanismo de Geração Dinâmica de Páginas

Para além das representações dos componentes de uma AW, existem dois conceitos que entendemos ser importantes representar. Por um lado queremos mostrar aquilo de que necessita uma página dinâmica para ser processada e onde vai buscar os conteúdos; por outro, de que forma processa a informação para gerar páginas diferentes. Comecemos por representar todas as páginas Cliente que podem ser geradas por uma página dinâmica. Uma forma de o fazer está representada na Figura 8 onde estão representadas todas as páginas geradas por uma determinada página dinâmica sendo o arco que as liga a representação de uma das relações que este tipo de página possui: a relação *"gera"*. Se se quiser ver com mais detalhe essa relação, ao seleccionar uma página é visualizada uma imagem da mesma página aumentada e as condições que precisam de ser verificadas para esta ser enviada ao Cliente.

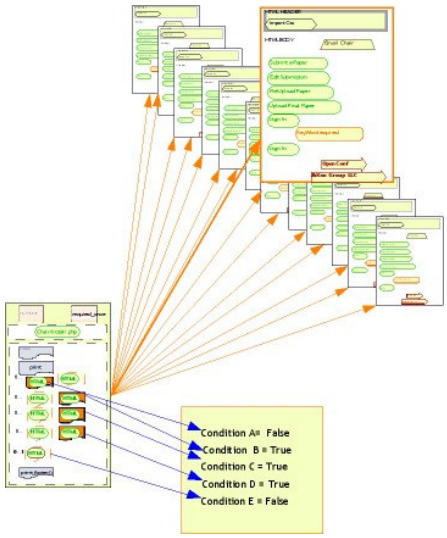


Figura 6 - Página Dinâmica: Relação

Para mostrar a forma como são gerados os conteúdos das páginas Cliente, é importante perceber o seu processamento no lado do servidor - perceber quais as condições que precisam ser satisfeitas. A Figura 6 pretende representar essa informação, mostrando os ficheiros que importa e relações com bases de dados. Ficheiros que nunca são directamente chamados pelo Cliente mas que são chamados por outros --- considerados como ficheiros do *T3* --- têm fundo avermelhado. A Figura 7 representa as dependências da página dinâmica, no Servidor, para a geração de determinada página no Cliente; é útil para detectarmos quais os ficheiros implicados na geração de uma página. Para se perceber que condições precisam ser satisfeitas para obter determinada página no Cliente, a Figura 7 é uma possível outra forma de representar a mesma informação, no entanto, foca objectivamente uma página estática e permite sabermos - através da visualização das condições que são precisas ser satisfeitas - como foi possível obter determinada página estática.

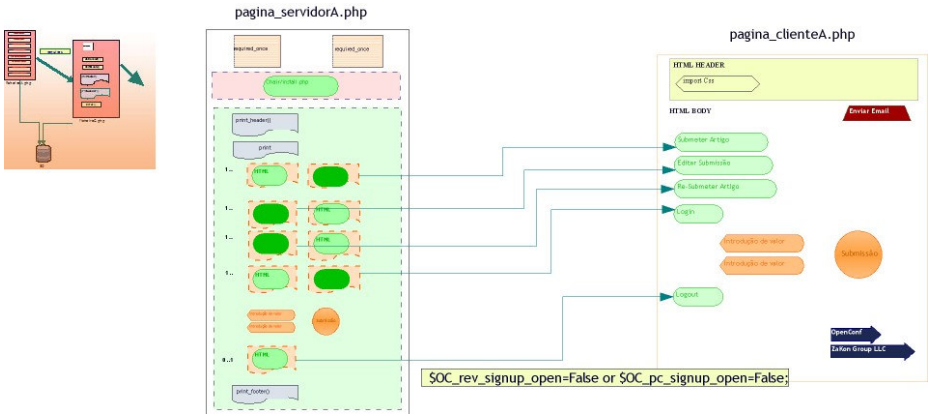


Figura 7 - Condições para Geração de Página Cliente

Análise Comportamental

As representação que iremos considerar, na análise comportamental das AW, são as adoptadas nas aplicações *WARE [Di Lucca]* e *PBS[Hassan and Holt]* - descritas no artigo [XXX].

Pressupomos por isso, que para representar as funcionalidades da aplicação se podem usar Diagramas de Classe. Para representar o percurso da aplicação relativamente a uma dada funcionalidade podem usar-se Diagramas de Sequência em UML.

## 4. Conclusão

Compreender aplicações, requer por um lado o conhecimento de factores externos que podem influenciar o seu funcionamento, mas o essencial reside na análise ao código. Para isso é preciso encontrar formas de o representar logicamente, de modo a proporcionar ao utilizador abstracções que o ajudem a, mais facilmente, captar e transformar internamente em representações mentais, construindo desta forma modelos da aplicação. Neste artigo propusemos uma forma de representar logicamente algumas das características das AW, evidenciando as suas relações e lógica de funcionamento. Por um lado, representámos as páginas no Cliente, destacando os elementos que dinamizam a aplicação. Por outro, representámos as páginas no Servidor, através dos elementos responsáveis pela geração de páginas, destacando as partes do ficheiro que podem ou não ser enviadas ao Cliente. Encontrámos também uma forma de representar todos os possíveis fragmentos de código enviados ao Cliente – pela página no Servidor - e ao mesmo tempo conseguimos informar sobre as condições que precisam ser satisfeitas para que cada fragmento seja enviado. Para além destas representações, do nosso ponto de vista essenciais na compreensão de AW, completámos a análise com os tipos de características que um sistema de visualização para compreensão de AW deve proporcionar, como o uso de vistas, interactividade, navegação, contextualização, apresentação e animação.

## 5. Referências


Storey, Margaret-Anne , "Theories, methods and tools in program comprehension: Past, present and Future", *IWPC '05: Proceddings of the International Workshop on Program Comprehension*, , 181-191. Washington, DC, USA, 2005. IEEE Computer Society.

Richard C. Holt Ahmed E. Hassan. *Towards a better understanding of Web applications*, WSE '01 Proceddings of the 3rd Internations Workshop on Web Site Evolution, 2001. IEEE Computer Society

XXX

Giuseppe Antonio Di Lucca, Anna Rita Fasolino, and Porfirio Tramontana, "*Reverse Engineering web applications: the ware approach*", Journal of Software Maintenance and Evolution, 16(1-2):71-101, 2004.

Rob Lintern, Jeff Michaud, Margaret-Anne Storey, and Xiaomin Wu. *Plugging-in visualization: experiences integrating a visualization tool with eclipse*. In SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization, pages 47–ff, San Diego, California, 2003. ACM Press.

A. Hassan and R. Holt. *A visual architectural approach to maintaining web applications*, 2003.

M.-A. D. Storey, F. D. Fracchia, and H. A. Mueller. *Cognitive design elements to support the construction of a mental model during software visu-alization*. In WPC '97: Proceedings of the 5th International Workshop on Program Comprehension (WPC '97), page 17, Washington, DC, USA, 1997.IEEE Computer Society.