Normas no Desenvolvimento de Ambientes Web Inclusivos e Flexíveis

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Frederico José Fortuna e aprovada pela Banca Examinadora.

Campinas, 18 de maio de 2010.

Profª. Drª. Maria Cecília Calani Baranauskas
(Orientadora)

Prof. Dr. Rodrigo Bonacin
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Miriam Cristina Alves – CRB8 / 5089

Fortuna, Frederico José
F779n Normas no desenvolvimento de ambientes Web inclusivos e flexíveis/Frederico José Fortuna-- Campinas, [S.P. : s.n.], 2010.

Orientadores : Maria Cecília Calani Baranauskas; Rodrigo Bonacin.
Dissertação (Mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Interfaces de usuário (Sistema de computador). 2. Interação humano-computador. 3. Semiótica e computação. 4. Sites da Web - Desenvolvimento.. I. Baranauskas, Maria Cecilia Calani, 1954-. II. Bonacin, Rodrigo. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Título em inglês: Norms in the development of inclusive and flexible Web environments.

Palavras-chave em inglês (Keywords): 1. User interfaces. 2. Human-computer interaction. 3. Semiotics and computing. 4. Web site development.

Área de concentração: Interação Humano-Computador

Titulação: Mestre em Ciência da Computação

Banca examinadora: Profa. Dra. Maria Cecília Calani Baranauskas (IC-UNICAMP)
Profa. Dra. Lúcia Filgueiras (Escola Politécnica – USP)
Prof. Dr. Rogério Drummond Burnier P. de Mello Filho (IC-UNICAMP)

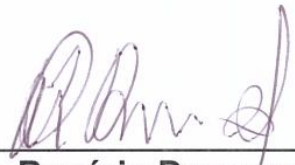
Data da defesa: 14/05/2010

Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 14 de maio de 2010, pela Banca examinadora composta pelos Professores Doutores:

Profª. Drª. Lucia Vilela Leite Filgueiras
Escola Politécnica / USP



Prof. Dr. Rogério Drummond Burnier Pessoa de Mello Filho
IC / UNICAMP

Profª. Drª. Maria Cecília Calani Baranauskas
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Normas no Desenvolvimento de Ambientes Web Inclusivos e Flexíveis

Frederico José Fortuna[1]
Maio de 2010

Banca Examinadora:

Profª Drª Maria Cecília Calani Baranauskas (Orientadora)
Instituto de Computação – UNICAMP

Profª Drª Lucia Filgueiras
Escola Politécnica - Universidade de São Paulo - USP

Prof. Dr. Rogério Drummond Burnier Pessoa de Mello Filho
Instituto de Computação – UNICAMP

Prof. Dr. Marcos Augusto Francisco Borges (suplente externo)
Faculdade de Tecnologia – UNICAMP

Profª Drª Ariadne Maria Brito Rizzoni Carvalho (suplente interno)
Instituto de Computação – UNICAMP

[1] Apoiado pela FAPESP, processo número 08/52261-4.

Resumo

De acordo com W3C, o valor social da Web está no fato de que ela possibilita a comunicação, o comércio, e oportunidades de troca de conhecimento. Estes benefícios deveriam estar disponíveis para todas as pessoas, com o hardware e versão do software que utilizam, sua infra-estrutura de rede, linguagem nativa, cultura, localização geográfica, habilidade física e conhecimento. Estes aspectos estão relacionados tanto a questões sociais quanto tecnológicas. Considerando a diversidade de usuários e a complexidade de situações possíveis de uso da Web, buscam-se soluções para interfaces mais flexíveis, que possibilitem sua adaptação a diferentes contextos de uso.

Este trabalho apresenta uma abordagem para solucionar o problema de como desenvolver interfaces de usuário flexíveis para sistemas Web, investigando como interfaces poderiam ser adaptadas a diferentes contextos de uso, considerando o conceito de normas da Semiótica Organizacional. Tal abordagem está representada em um framework, proposto neste trabalho, para apoiar designers e desenvolvedores na construção de interfaces flexíveis. Resultados obtidos na aplicação do framework em um sistema Web real, inserido no contexto da inclusão digital e acesso universal, são apresentados e discutidos nesta obra. Tais resultados são sugestivos da viabilidade da proposta e apontam para seu aprofundamento futuro.

Abstract

According to W3C, the social value of the Web is in the fact that it enables communications, business and knowledge sharing opportunities. These benefits should be available for every person regardless of the person's hardware, software, network infrastructure, native language, cultural aspects, geographical location, physical and mental abilities. These aspects are related both to social and technological issues. Considering the differences among users and the complexity of possible Web usage, solutions are sought for more flexible user interfaces that allow their adaptation to different use contexts.

This work presents an approach to solve the problem of developing flexible user interfaces for Web systems, investigating how interfaces can be adapted to different use contexts considering the concept of norms from Organizational Semiotics. This approach is represented by a framework, proposed on this work, that may help designers and developers to build flexible Web interfaces that may be adapted according to each use context. Results gathered when the framework was applied in a real Web system related to the context of universal access and digital inclusion are presented and discussed. Such results are suggestive of the proposal's viability and point to further improvements in future research.

Agradecimentos

Gostaria de agradecer, primeiramente, aos meus pais Ana Dalva e Vilmar por todo o apoio e carinho que me deram durante todo o tempo em que estive longe deles durante os estudos na universidade, uma vez que sem o apoio deles acredito que eu não conseguiria chegar aonde cheguei. Eu dedico este trabalho a toda a minha família, mas em especial aos meus pais.

Agradeço também a grande ajuda da minha orientadora, Cecília Baranauskas, e do meu co-orientador Rodrigo Bonacin, que me guiaram durante o desenvolvimento de todo esse trabalho e me deram conselhos que foram úteis não só na minha pesquisa como na minha vida. Agradeço também pela paciência dos meus orientadores ao me atenderem diversas vezes em reuniões nos seus locais de trabalho e nas diversas conversas por telefone!

Meus agradecimentos vão também para o meu irmão Vinicius, sua esposa Lorena, ao meu irmão Vitor e sua futura esposa Clarissa, por terem me ajudado de diversas formas durante meus estudos, mesmo quando a ajuda veio por meio de uma simples conversa pela internet devido à distância que nos separa. O incentivo, o carinho e a experiência deles certamente me ajudaram no desenvolvimento deste trabalho.

Estendo meus agradecimentos aos demais familiares por todo o incentivo que me deram durante o desenvolvimento deste trabalho, agradecendo em especial à minha querida tia Marlene, por todas às vezes que me deu apoio por telefone e ao me visitar em Campinas.

Gostaria de agradecer também os companheiros de pesquisa, integrantes do projeto e-Cidadania, pelas diversas dicas que foram bastante úteis na minha vida acadêmica e na vida pessoal.

Agradeço também aos meus amigos em geral pela companhia e incentivo durante todo o tempo em que estive em Campinas. Em especial, agradeço meus grandes amigos Pedrinho e Tati pelo suporte na vida acadêmica, nas conversas que tivemos sobre trabalhos, provas, relatórios e afins e agradeço pela companhia nas vezes que saímos ou viajamos juntos. Agradeço também aos amigos de "república" que conviveram comigo nos últimos seis anos. Apesar de algumas brincadeiras, considero todos grandes amigos.

Sou grato a todos os professores do Instituto de Computação que me ensinaram durante a graduação e o mestrado, pois o conhecimento passado por eles será importante na minha vida profissional. Em especial, agradeço aos professores Sindo, Rodolfo e Anido que considero não só professores, mas também bons amigos. Agradeço também aos funcionários do Instituto de Computação e do Núcleo de Informática Aplicado à Educação da Unicamp pela ajuda em situações do dia-a-dia na universidade.

À FAPESP, agradeço pelo apoio financeiro dado à minha pesquisa de mestrado, o que colaborou para que me dedicasse ainda mais à pesquisa e outras atividades acadêmicas.

Por fim, gostaria de agradecer a todas as pessoas que me ajudaram, direta ou indiretamente, na realização deste trabalho e agradeço a todas as pessoas que acreditaram no meu sucesso e no sucesso desta pesquisa. Eu gostaria muito de citar o nome de todas as pessoas a quem eu sou agradecido, mas a lista de nomes é muito extensa e não caberia nesta seção. A todos agradeço com um enorme "muito obrigado"!

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

Ajax – *Asynchronous JavaScript and XML*
AN – *Análise de Normas*
CMS – *Content Management System*
DOM – *Document Objetct Model*
e-Gov – *Governo Eletrônico*
FAN – *Flexibility via Ajax and Norms*
ICE – *Interface Configuration Environment*
IHC – *Interação Humano–Computador*
JEE – *Java Platform, Enterprise Edition*
JSON – *JavaScript Object Notation*
MEASUR – *Method for Eliciting, Analysing and Specifying User Requirements*
NAM – *Norm Analysis Method*
NBIC – *Norm-Based Interface Configurator*
OWL – *Web Ontology Language*
RDF – *Resource Description Framework*
RDFS – *Resource Description Framework Schema*
RSI – *Rede Social Inclusiva*
SO – *Semiótica Organizacional*
SOAP – *Simple Object Access Protocol*
SparQL – *SPARQL Protocol and RDF Query Language*
SWRL – *Semantic Web Rule Language*
WSDL – *Web Services Description Language*

Capítulo 1

Introdução

De acordo com *World Wide Web Consortium* (W3C, 2008), o valor social da Web está no fato de que ela possibilita a comunicação, o comércio, e oportunidades de troca de conhecimento. Ainda segundo a W3C, estes benefícios devem estar disponíveis para todas as pessoas, independente de seu hardware, software, infra-estrutura de rede, linguagem nativa, cultura, localização geográfica, habilidade física e mental. Estes aspectos estão relacionados tanto a questões sociais quanto tecnológicas.

Em decorrência da complexidade que esta diversidade de usuários e usos carrega, interfaces deveriam ser flexíveis possibilitando sua adaptação a cada contexto. Uma alternativa é oferecer aos usuários finais e outras partes interessadas como analistas de negócio, administradores públicos, membros de ONGs, entre outros, a oportunidade de ajustar aplicações Web a suas necessidades. Entretanto, estes ajustes vão além de simples escolhas de opções de visualização. Aspectos relacionados à semântica dos elementos da interface, a intenção de uso, e o contexto pragmático/social[2] devem ser considerados.

Considerando estes aspectos, quando alguém acessa uma pagina Web, é possível dizer que uma interação bidirecional, que envolve intenções, deveres, comprometimento, e responsabilidades, é estabelecida. Esta interação por sua vez está imersa em um contexto social complexo que contem leis, regras e cultura. Por exemplo, quando alguém acessa um site de governo eletrônico pode ter diferentes intenções, como visualizar as

[2] Por aspectos Sócio-Pragmáticos entende-se aqui intenções, comunicações, conversações, negociações, crenças, expectativas, funções, comprometimentos, contratos, leis, cultura, entre outros, baseados no nível pragmático e no nível social do *framework* semiótico (Stamper, 1993).

contas do governo, pagar taxas, marcar consulta, procurar emprego, entre outras. Se o cidadão marcar uma consulta médica, então uma relação de compromisso entre a entidade governamental e o cidadão é formada, onde ele estará comprometido a comparecer no local definido e a entidade de governo a prover a consulta. Entender esta relação é necessário para se aprimorar a qualidade dos sistemas, via interface de usuário.

De acordo com o IEEE *Standard Computer Dictionary*, flexibilidade é a facilidade com a qual um sistema ou componente pode ser modificado para uso em aplicações ou ambientes diferentes daqueles para os quais foi especificamente construído (IEEE, 1990). Atualmente interfaces Web não são, em geral, flexíveis. A maior parte delas é estática, porém há algumas exceções como as interfaces dos serviços iGoogle[3], Pageflakes[4], Netvibes[5] e MyYahoo![6]. Entretanto, a flexibilidade provida pelas interfaces desses serviços não é suficiente quando um contexto vasto e diversificado de usuários é considerado; as opções de ajuste fornecidas por esses serviços estão restritas a um conjunto de possibilidades que são as mesmas para todos os usuários. Além disso, estas interfaces não levam em conta aspectos sociais e não são capazes de inferir necessidades dos usuários.

Diante do fato de que interfaces devem ser flexíveis para atenderem os diferentes interesses e necessidades de usuários e diante dos exemplos de interfaces ajustáveis, porém limitadas, o problema abordado nesta dissertação envolve: como criar interfaces flexíveis, que levem em conta diferentes contextos de uso, para construir sistemas Web inclusivos?

Este trabalho apresenta uma abordagem para solucionar o problema de como desenvolver interfaces de usuário flexíveis para sistemas Web como um todo, investigando como interfaces poderiam ser adaptadas a diferentes contextos de uso, considerando aspectos semânticos, pragmáticos e sociais. Tal abordagem está representada em um framework, proposto neste trabalho, para apoiar designers e

[3] http://www.google.com/ig

[4] http://www.pageflakes.com/

[5] http://www.netvibes.com/

[6] http://my.yahoo.com/

desenvolvedores na construção de interfaces flexíveis que levam em conta o contexto de uso de cada usuário. O trabalho também apresenta a aplicação deste framework em sistema desenvolvido no contexto de inclusão e acesso universal.

Como referencial teórico-metodológico básico, para modelagem dos aspectos humanos da interação, é proposta a Semiótica Organizacional (SO) (Liu, 2000), com foco nos métodos da Análise de Normas, que provê a representação do uso intencional dos signos e do comportamento resultante dos agentes responsáveis em um contexto social. Em adição à SO, também é considerado neste trabalho o referencial relativo à construção de sistemas flexíveis para prover o acesso universal (Savidis and Stephanidis, 2004).

1.1 Objetivos

Este trabalho teve três objetivos principais: (1) estudar aspectos técnicos do design e desenvolvimento de ambientes Web inclusivos; (2) propor e desenvolver um framework que permitisse o desenvolvimento de interfaces flexíveis fundamentadas no conceito de normas, oriundo da Semiótica Organizacional; e (3) validar a solução fornecida pelo framework, integrado a um sistema com usuários finais.

Estudar aspectos técnicos do design e desenvolvimento de ambientes Web inclusivos foi necessário na proposta de um framework para permitir o desenvolvimento de interfaces Web flexíveis. O desenvolvimento do framework constituiu o principal objetivo deste trabalho, em conjunto com sua validação com usuários finais para verificar se o framework se comportaria como previsto.

1.2 Estrutura da dissertação

Esta dissertação está organizada em cinco capítulos, com a seguinte estrutura: no Capítulo 2 é apresentado o referencial teórico-metodológico utilizado na pesquisa, no Capítulo 3 é apresentado o framework proposto para facilitar o desenvolvimento de interfaces Web flexíveis, incluindo sua arquitetura, uma descrição de seu funcionamento e as tecnologias empregadas em seu desenvolvimento. Em seguida, no Capítulo 4, é descrita a forma como a solução do framework foi validada, sendo apresentados, entre

outras informações, os resultados obtidos e uma discussão sobre tais resultados e as vantagens e pontos fracos do framework. Por fim, o Capítulo 5 apresenta a conclusão deste trabalho, incluindo uma discussão sobre trabalhos futuros.

Capítulo 2

Caracterização do Problema e

Referencial Teórico-Metodológico

Neste capítulo são apresentados fundamentos usados para delinear o desenvolvimento do framework proposto nesta pesquisa. Na seção 2.1 é introduzido o conceito de interfaces flexíveis, incluindo uma visão geral sobre pesquisas nesse campo. A seção 2.2 apresenta conceitos de Ajax e de outras tecnologias Web utilizadas na solução proposta. Na seção 2.3 são apresentados o conceito de Normas e outros conceitos da Semiótica Organizacional que foram utilizados como base teórica para este projeto. Este capítulo é encerrado na seção 2.4, com a apresentação de uma solução conceitual, para o problema de como desenvolver interfaces Web flexíveis, com a utilização de Normas no desenvolvimento de interfaces Web. Além disso, nessa seção são apresentadas tentativas anteriores de se utilizar normas para prover interfaces flexíveis, sendo tais tentativas usadas como base para este trabalho.

2.1 Interfaces flexíveis

A definição de flexibilidade pelo IEEE *Standard Computer Dictionary* sugere que interfaces de usuário, flexíveis, devem ser facilmente modificadas para diferentes

contextos de uso. Entende-se que, para que se alcance esse requisito, fatores técnicos e sociais devem ser considerados no design do sistema e da interação.

De acordo com Savidis e Stephanidis (2004, p. 165), "*users are no longer only the traditional able-bodied, skilled and computer-literate professionals; product developers can no longer know who their target users will be; information is no longer relevant only to the business environment; and artifacts are no longer bound to the technological specifications of a pre-defined interaction platform"*. Neste cenário, interfaces não podem ser consideradas como artefatos estáticos construídos para usuários regulares e bem conhecidos. Além disso, o ambiente tecnológico e social está em constante evolução, sempre exigindo mudanças na interface de usuário. Assumindo que não existe utilidade prática em uma interface na qual os usuários não conseguem fazer o que eles querem, nós estamos diante de um novo desafio: prover interfaces úteis e fáceis de usar para usuários desconhecidos e em um ambiente em constante desenvolvimento.

Literatura, em Interação Humano-Computador (IHC), tem apontado diferentes alternativas para prover flexibilidade na interface do usuário. Dourish (1996) argumenta que não podemos desassociar disciplinas técnicas e sociais no design de sistemas. A relação entre elas é consideravelmente mais intricada do que sugere a visão tradicional, onde requisitos e restrições não têm conseqüências além da aplicação, e aspectos sociológicos não se aplicam dentro do sistema. Como conseqüência, para Dourish (1996), flexibilidade na infraestrutura e no uso do sistema estão fortemente relacionadas. A partir desta perspectiva, ele tem proposto um framework customizável para sistemas computacionais de trabalho cooperativo, permitindo aos programadores e usuários finais realizar *tailoring* em estruturas de ferramentas de acordo com as necessidades de aplicações e domínios.

Tailoring é um conceito relacionado à nossa noção de flexibilidade; esse conceito pode ser entendido como *a atividade de modificar uma aplicação de computador dentro do contexto de seu uso* (Kahler et al., 2000, p. 1). Pesquisas em sistemas no quais se pode realizar *tailoring* geralmente pressupõem que o usuário deva ficar responsável pela tarefa de realizar *tailoring* (Teege et al., 1999; Morch and Mehandjiev, 2000). Um desafio em *tailoring* é como prover, aos usuários, interfaces de alto nível nas quais mudanças

relevantes podem ser realizadas sem a necessidade de o usuário ter grande habilidade técnica ou experiência. Morch (1995) identifica três níveis para realização de *tailoring* por um usuário final: (1) Customização, que permite modificar a aparência de objetos de apresentação ou editar os valores de seus atributos por meio da seleção de opções de configuração pré-definidas (Morch, 1995, p. 44); (2) Integração, que permite a usuários adicionar novas funcionalidades a uma aplicação (Morch, 1995, p. 44); e (3) Extensão, que é uma abordagem a *tailoring* na qual a funcionalidade de uma aplicação é melhorada pela adição de novo código (Morch, 1995, p.47). A Figura 1 apresenta, de forma esquemática, a forma de uso destas três categorias. Pode-se observar que os *gaps* na Figura 1 representam os pontos em que os mecanismos de *tailoring* devem atuar para complementar a ação do usuário no processo.

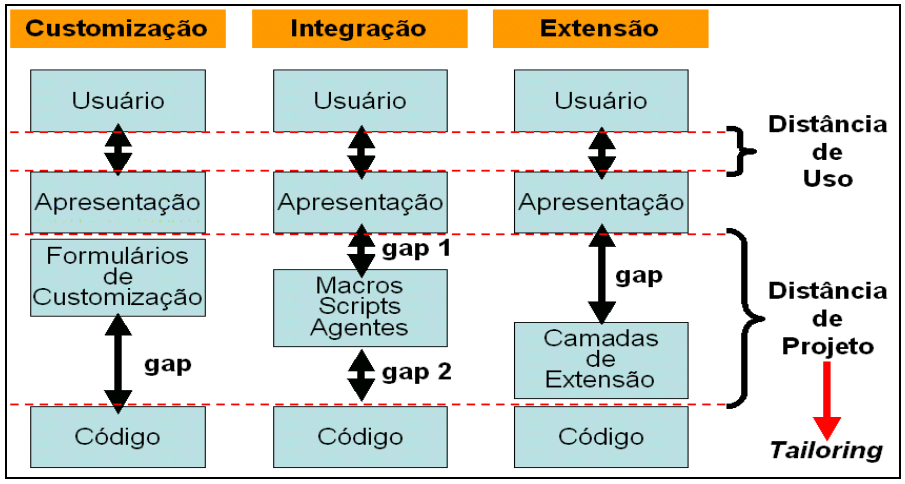


Figura 1: Esquema das categorias de Tailoring, adaptado de Morch (Morch, 1995) .

Entre outras abordagens, as aplicações baseadas em contexto (Moran and Dourish, 2001) coletam e utilizam informações em relação ao contexto no qual o fornecimento de serviços é adequado a cada pessoa, lugar, evento etc. Contexto se refere às situações físicas e sociais nas quais dispositivos computacionais estão incluídos. Fischer e outros (2004) têm proposto o conceito de "Meta-Design", uma visão na qual design, aprendizado e desenvolvimento fazem parte das práticas de trabalho diárias de usuários finais.

Neste trabalho, uma abordagem para a construção de sistemas flexíveis é explorada, em conjunto com técnicas e métodos da SO, para possibilitar a construção de interfaces

capazes de se ajustar de acordo com o contexto pragmático/social de uso, resultando na proposta de um framework para facilitar o desenvolvimento de interfaces flexíveis. Usando técnicas para a construção de interfaces flexíveis em conjunto com SO, usuários e especialistas no domínio podem, por exemplo, criar e manter normas que definem aspectos que regem o uso sistema.

2.2 Tecnologias Web utilizadas

Entre as tecnologias utilizadas no framework proposto, a principal é Ajax. Ajax, acrônimo em inglês de "JavaScript Assíncrono e XML", pode ser entendido como um novo estilo, que inclui outras tecnologias Web, para desenvolver aplicações Web. Ajax não é uma tecnologia nova, ela emergiu na indústria de software como uma forma de juntar algumas tecnologias existentes, a fim de fornecer "interfaces Web mais ricas".

O termo Ajax foi inventado por Garret (2005) e incorpora as seguintes tecnologias:

- Apresentação baseada em padrões, usando XHTML e CSS;
- Exibição e interação dinâmica usando o *Document Objetct Model* (DOM);
- Troca e manipulação de dados usando XML e XSLT;
- Recuperação assíncrona de dados usando XMLHttpRequest;
- JavaScript interligando tudo;

Ajax muda substancialmente a interação cliente-servidor. O clássico modelo síncrono no qual cada requisição de usuário é seguida por uma recarga de página é substituído por uma comunicação assíncrona independente na qual o usuário não precisa esperar em uma "página preta" pelo processamento do servidor.

Parte do processamento pode ser executada no lado do cliente. Esse aspecto minimiza a carga de trabalho do servidor e, principalmente, de comunicação. Muitas aplicações que eram inicialmente restritas às plataformas desktop, devido a limitações do desenvolvimento HTML padrão, podem ser transpostas para a Web com a utilização de recursos da tecnologia Ajax. Por exemplo, é possível realizar "*drag and drop*" com imagens, aplicar filtro nelas e mudar seus tamanhos sem recarregar a página. Sobre flexibilidade de interface, Ajax permite mudar funcionalidades, remover ou incluir

qualquer objeto de interface em uma página Web sem a necessidade de uma recarga de página.

Algumas das primeiras aplicações populares a utilizarem Ajax foram o Google Suggest, Google Maps e Gmail. Hoje em dia, Ajax é utilizado amplamente no desenvolvimento Web e muitas plataformas e ferramentas de desenvolvimento suportam Ajax. O *World Wide Web Consortium* (W3C) também está trabalhando, no momento, para padronizar algumas tecnologias fundamentais do Ajax, como a especificação do XMLHttpRequest (W3C, 2009).

O Ajax também propõe o uso da tecnologia DOM. De acordo com o *World Wide Web Consortium* (W3C, 2004), DOM é uma interface independente de plataforma e linguagem que permite a programas e *scripts* acessarem e modificarem dinamicamente o conteúdo, estrutura e estilo de documentos na Web.

JSON, sigla em inglês para "*JavaScript Object Notation*", é mais uma tecnologia que foi utilizada neste projeto. JSON é um formato para troca de dados entre aplicativos. Esse formato foi baseado em JavaScript, porém é um formato de texto independente de linguagem ou plataforma. Esse formato é fácil de ser lido e escrito por desenvolvedores, o que facilita seu uso. Além dessa facilidade, JSON tem uma boa eficiência devido ao fato de ser um formato leve, simples, para troca de dados.

O framework proposto neste trabalho também acessa serviços remotos utilizando SOAP, sigla para a expressão em inglês *"Simple Object Access Protocol*". Segundo o W3C (2000), "*SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is a XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses*".

2.3 Semiótica Organizacional e Normas

A Semiótica Organizacional (SO) entende uma organização como um sistema de signos e objetiva estudá-la utilizando conceitos e técnicas baseados na Semiótica desenvolvida por Peirce (1931-1958) e Morris (1938) entre outros. SO entende que todo comportamento

organizado é afetado pela comunicação e interpretação dos signos pelas pessoas, de maneira individual ou em grupos. O MEASUR - *Methods for Eliciting, Analyzing and Specifying User's Requirements* (Stamper, 1993) é um conjunto de métodos baseados em conceitos da SO para lidar com signos e seus significados (Semântica), intenções (Pragmática), e conseqüências sociais de seu uso (Social). Neste trabalho são utilizados principalmente conceitos de dois métodos do MEASUR: Análise Semântica e, em especial, conceitos da Análise de Normas.

O método de Análise Semântica (AS) ajuda os analistas e usuários na elicitação e representação de requisitos de uma maneira precisa e formal, na qual é possível descrever uma visão dos agentes do domínio e seus padrões de comportamento chamados *affordances*. *Affordance* é um conceito introduzido por Gibson (1968) que pode ser utilizado para expressar invariantes de repertórios de comportamento de um organismo possíveis pela estrutura do organismo combinada com a de seu ambiente.

O conceito de *affordance* também é utilizado em IHC (Interação Humano-Computador); segundo Norman (1999) ele próprio deveria ter utilizado o termo "*perceived affordance*" (ou *affordance* percebida) porque no design de interfaces estamos muito mais interessados sobre o quê o usuário percebe de determinado objeto. Em interfaces gráficas os designers têm algum controle sobre os "*perceived affordances*"; eles devem garantir, por exemplo, que o usuário perceba o significado associado ao ato de pressionar em uma determina região da tela (Norman, 1990).

No MEASUR a noção que Gibson introduziu foi estendida por Stamper para incluir invariantes que são percebidas no mundo social. Estas invariantes, no mundo social, são chamadas de normas sociais por Stamper (1996, p. 374): "Gibson deu maior enfoque à percepção do mundo físico; entretanto, a noção de *affordance* pode ser generalizada para incluir invariantes que nós percebemos no mundo social. Se alguém tem uma patente de direitos autorais então deveríamos supor um comportamento invariante das pessoas em direção a respeitar o seu trabalho. Um copo tem várias invariantes sociais que são válidas, por exemplo, para nos permitir beber algo de uma maneira aceitável em uma companhia refinada e ele pode ter também a invariante de posse, assim como a patente pode...".

Na Análise Semântica representações de *affordances* são combinadas com outros conceitos como dependência ontológica, agentes, parte-todo, generalização,

determinantes, entre outros, para formar um diagrama de ontologia. Conforme destacado por Liu (2000), este método produz um modelo estável da organização, pouco suscetível a mudanças no contexto organizacional. Este modelo deverá ser utilizado para representar a parte da interface menos sucessível a mudanças.

O método de Análise de Normas (AN) tem o foco nas normas sociais, culturais e organizacionais que governam as ações dos agentes. Normas definem as responsabilidades dos agentes engajados em alguma tarefa, ou as condições sob as quais certas ações podem ou não ser realizadas por um agente.

No nível pragmático a AN descreve o relacionamento entre o uso intencional dos signos para comunicação entre agentes e o comportamento resultante dos responsáveis no contexto social. No nível social as normas expressam crenças, expectativas, compromissos, contratos, leis, culturas e também o negócio. Normas correspondem no nível social à idéia de *affordance* no nível individual (Liu, 2000). As normas são resultantes de ações em uma sociedade e ao mesmo tempo controlam e ordenam as ações dos agentes em um contexto social.

A Análise de Normas é um método utilizado para discutir aspectos dos níveis pragmático e social da organização. Normas são mais suscetíveis às mudanças no contexto organizacional e em intenções dos agentes em sociedade.

2.4 Normas e interfaces flexíveis: solução conceitual

Do ponto de vista conceitual, o framework proposto é baseado no Método de Análise de Normas (em inglês, NAM) do MEASUR. Como dito anteriormente, normas descrevem as relações entre um uso intencional de signos e o comportamento resultante de agentes responsáveis em um contexto social. Normas também descrevem crenças, expectativas, compromissos, contratos, leis, cultura, assim como negócios.

"Normas podem ser representadas em todos os tipos de signos, seja em documentos, comunicação oral ou condutas, com o objetivo de preservar, difundir e segui-las. Entretanto, não se pode sempre colocar as mãos convenientemente em uma norma, como se pode pegar um documento que carrega informações através de uma organização. Uma norma é mais como um campo de força que faz os membros de uma comunidade

tenderem a se comportarem ou pensarem em determinada maneira." (Stamper et al. 2000, p. 15)

Além da descrição das responsabilidades dos agentes no contexto organizacional, Análise de Normas pode também ser usada para analisar as responsabilidades de manter, adaptar e personalizar as funcionalidades do sistema.

Normas podem ser representadas pelo uso de linguagem natural ou Lógica Deôntica nos últimos estágios de modelagem. O seguinte formato é adequado para especificar normas comportamentais (Liu, 2000):

<Norma>::= sempre <condição>se <estado> então <agente>é<D>executar <ação>

Onde <D> é um operador deôntico que especifica se a ação é obrigatória, permitida ou proibida. As normas não são necessariamente obedecidas por todos os agentes em todas as circunstâncias; elas são convenções sociais (leis, papéis e convenções informais) que devem ser obedecidas. Por exemplo: uma norma especifica que os agentes são obrigados a pagar uma taxa; se um agente não tem dinheiro ele não irá pagar, mas geralmente há implicações quando um agente não obedece às normas.

Bonacin et al. (2009) apresentam os fundamentos, um framework, e um conjunto de ferramentas para prover serviço personalizado, utilizando simulação de normas. Parte de seu framework é utilizada nesta proposta; o framework em Bonacin et al. é limitado por utilizar soluções "*thin client*", limitadas, uma vez que uma recarga de página deveria ser feita para cada mudança na interface, além de tratar somente eventos síncronos. A parte de seu framework que é utilizada neste trabalho é a ferramenta NBIC/ICE, uma ferramenta de gerenciamento de normas desenvolvida por Bonacin (2004, 2007). NBIC (*Norm Based Interface Configurator*), sigla em inglês para "Configurador de Interface Baseado em Norma", é uma ferramenta que armazena e gerencia normas. ICE (*Interface Configuration Environment*), sigla em inglês para "Ambiente de Configuração de Interface", é uma ferramenta que realiza inferências, utilizando a máquina de inferência Jess, sobre normas armazenadas no NBIC e cria planos de ação, que descrevem ajustes que devem ser feitos na interface do sistema e como esses ajustes devem ser realizados.

Este trabalho expande as possibilidades do framework proposto por Bonacin et al., do ponto de vista tecnológico e prático, utilizando a tecnologia Ajax. Além disso, é provida uma forma mais fácil e produtiva de desenvolver interfaces flexíveis. Neste trabalho, normas podem ser avaliadas durante a interação (tempo real) do usuário com o sistema, e as avaliações podem refletir em mudanças imediatas na interface.

Esta pesquisa propõe simular normas e deduzir, em tempo real, o comportamento esperado de usuários e organizações durante a interação do usuário. Por exemplo, se uma pessoa acessa um portal de governo eletrônico (e-Gov) e é obrigada a pagar uma taxa, então esta pessoa estará provavelmente interessada em acessar informações sobre o processo de pagamento. Logo, a interface do sistema pode ser imediatamente adaptada, apresentando, com destaque, informações sobre o processo de pagamento. Dessa forma, a interface pode ser modificada de acordo com necessidades e preferências do usuário.

Normas são suscetíveis a mudanças no contexto organizacional e na representação de intenções de agentes na sociedade. Na abordagem proposta neste trabalho, especialistas de domínio, designers e usuários mantém as especificações das normas de acordo com as mudanças em seu contexto pragmático-social (Stamper, 1973).

Capítulo 3

FAN: Flexibilidade via Ajax e Normas

Neste capítulo é apresentado o framework[7] proposto para facilitar o desenvolvimento de interfaces Web flexíveis. O framework em questão se chama FAN, sigla para "*Flexibilidade via Ajax e Normas*" ou, em Inglês, "*Flexibility via Ajax and Norms*". Primeiramente são discutidos os requisitos do FAN e as tecnologias utilizadas em desenvolvimento; em seguida é apresentada a arquitetura do framework e seu funcionamento. Este capítulo é encerrado com as funcionalidades do framework.

3.1 Requisitos do framework

Nas etapas iniciais do desenvolvimento do framework, antes de especificar sua arquitetura, foram levantados os requisitos que o framework deveria atender. Tais requisitos foram definidos a partir de levantamentos com o público alvo do projeto e-Cidadania (Baranauskas, 2007) e a partir da experiência de outros projetos anteriores. Estes requisitos foram cruciais nas decisões de quais tecnologias seriam utilizadas em todo o processo. Seguindo a proposta inicial, o framework deveria prover meios de modificar interfaces Web em tempo real, sem necessitar de recarga de página, sendo tais modificações orientadas por normas modeladas no sistema.

O framework deveria atender, portanto, a uma série de requisitos conforme listado a seguir. É apresentado, em seguida, um detalhamento de cada requisito. O framework deveria ser capaz de:

[7] O termo *framework* é definido pelo dicionário Cambridge como "*1. a supporting structure around which something can be built; 2. a system of rules, ideas or beliefs that is used to plan or decide something*". Aqui o termo já aparece instanciado no contexto da computação.

1. Identificar e capturar eventos na interface (ex: clique de mouse sobre ajuste de fonte na interface);
2. Gerar e gravar, em arquivos, fatos relacionados aos eventos capturados (ex: usuário prefere fonte em tamanho maior) e carregar, dos arquivos, esses fatos;
3. Comunicar-se com a ferramenta NBIC/ICE, responsável por determinar ajustes que devem ser feitos na interface com base em inferências feitas ao analisar normas e fatos[8], enviando, para essa ferramenta, fatos gerados pelo framework e recebendo os dados resultantes do processamento realizado pela ferramenta;
4. Analisar os dados enviados pela ferramenta NBIC/ICE e modificar a interface Web do sistema de acordo com as orientações contidas nesses dados;
5. Acessar elementos da interface Web do sistema e modificar seus atributos, independente dos tipos dos atributos (ex: atributos de estilo ou de comportamento), remover elementos de uma página e também ser capaz de executar, na interface, código Javascript contido nos dados enviados pela ferramenta NBIC/ICE;
6. Permitir desenvolvedores ou designers de sistemas Web agregarem facilmente novas capacidades, no que diz respeito à flexibilidade, em suas aplicações através de um framework de desenvolvimento.

O requisito de identificar e capturar eventos na interface decorre do fato de que todo o processo de ajuste de interface deve ser iniciado a partir de ações (eventos) do usuário. O designer da interface do sistema é encarregado de determinar quais eventos, provocados pelo usuário na interface, devem disparar o mecanismo de ajuste, uma vez que o processo não precisa ser iniciado necessariamente a qualquer evento. Eventos de interface podem ser entendidos como ações que o usuário realiza na interface por meio de mouse, teclado ou outro dispositivo de entrada. Podem ser considerados, como exemplos de eventos: um usuário entrar no sistema, clicar em um determinado elemento na tela, passar o mouse por cima de um botão, selecionar uma imagem, entre outras possibilidades.

[8] Fatos podem ser entendidos, neste trabalho, como assertivas a respeito do contexto de uso do sistema

O requisito seguinte, de gerar, gravar e carregar, em arquivos, fatos relacionados aos eventos capturados surgiu a partir de dois problemas. O primeiro problema é que a ferramenta auxiliar, usada para gerenciar normas do sistema, necessita receber fatos, relacionados aos eventos na interface, como parâmetro de entrada para ela inferir mudanças que devem ser feitas na interface com base nas normas de sistemas e fatos recebidos.

Logo, o framework deve ser capaz de gerar fatos relacionados aos eventos de interface que iniciem o mecanismo de ajuste. Um fato é composto de um conjunto de predicados, sendo cada predicado composto de um *affordance* e um conjunto de elementos. Cada elemento é constituído de uma *string* e um valor booleano.

O segundo problema é que ajustes feitos na interface precisavam ser gravados de alguma forma para que o usuário não tivesse que ajustar a interface toda vez que ele entrasse no sistema ou, ainda, quando mudasse de página ou apenas executasse uma atualização de página. Isto é, os ajustes precisavam ser mantidos. Logo, foi decidido gravar, em arquivos, todos os fatos de usuários que disparem o mecanismo de ajuste de interface. Toda vez que uma página for carregada por um determinado usuário, o framework deve conseguir carregar, a partir dos arquivos gravados, os fatos relacionados ao usuário em questão e à página atual, de forma que processo de ajuste seja disparado, automaticamente, para esses fatos carregados a partir de arquivos. Dessa forma, a interface será ajustada, durante seu carregamento, às necessidades ou gosto do usuário de acordo com os ajustes que esse usuário realizou nessa página anteriormente. Vale destacar que os fatos relacionados a usuários permitem que um determinado ajuste possa valer para todas as paginas do sistema, ou para um conjunto de páginas com determinada característica.

O framework deve ser capaz de se comunicar com uma ferramenta auxiliar, que realiza inferências, ao analisar normas e fatos relacionados a eventos na interface do sistema, para determinar ajustes que devem ser feitos na interface. Esse requisito decorre da decisão de se utilizar uma ferramenta já existente para realizar a tarefa de gerenciar normas e inferir ajustes a serem feitos na interface, a partir de fatos relacionados a eventos de interface, já que o framework não seria responsável por realizar tal tarefa. A ferramenta proposta deve, portanto, ser capaz de enviar fatos, gerados previamente, para

essa ferramenta auxiliar e receber os dados enviados pela ferramenta de gerenciamento de normas. O framework deve analisar os dados recebidos, o que caracteriza o quarto requisito, e realizar ajustes na interface de acordo com esses dados, caso ajustes sejam realmente necessários. A análise da informação enviada pela ferramenta auxiliar de gerenciamento de normas é importante porque permite ao framework saber se a interface do sistema deve ser ajustada e como o ajuste deve ser feito.

Caso o framework tenha que realizar modificações diretamente na interface Web, ele deve conseguir acessar elementos de uma página e modificar seus atributos, independentemente dos tipos dos atributos, e também deve ser capaz de remover elementos da interface e executar, nela, código JavaScript contido nos dados enviados pela ferramenta de gerenciamento de normas. Permitir a execução de código JavaScript é importante, pois o designer da interface do sistema pode desejar que um determinado código JavaScript seja executado toda vez que ocorrer um certo evento na interface do sistema. Essas ações em conjunto determinam uma forma de se realizar modificações em uma página Web, o que caracteriza o próximo requisito do framework, relacionado à ação de modificar a interface.

O requisito de "permitir desenvolvedores ou designers de sistemas Web agregarem facilmente novas capacidades" decorre da importância do framework prover, a desenvolvedores ou designers, a capacidade deles agregarem, à interface, novas funcionalidades ou opções de ajuste sem haver necessidade de modificar o código do núcleo (*core*) do sistema principal. Por meio de normas e regras [9], designers ou desenvolvedores podem, por exemplo, inserir novas funcionalidades na interface ao executar códigos JavaScript, contidos nas regras, não havendo assim a necessidade de reprogramação do sistema.

É importante ressaltar que o framework é quase totalmente independente do sistema onde ele é utilizado, uma vez que o seu uso não implica em alterações no núcleo do sistema. O framework é, basicamente, um conjunto de códigos JavaScript que são agregados às páginas.

[9] Uma regra pode ser entendida como uma instância de uma norma no meio técnico.

3.2 A Utilização de Tecnologias Web

No desenvolvimento do framework foram utilizadas algumas tecnologias poderosas que são bastante utilizadas atualmente no desenvolvimento Web: Ajax (JavaScript e XML), PHP, DOM, JSON e *webservices* via SOAP.

Como o foco principal do framework é realizar ajustes em interfaces Web, geralmente construídas com HTML e CSS, sem a necessidade de novo carregamento (*reload*) de página, decidiu-se utilizar Ajax como tecnologia principal no desenvolvimento do framework, uma vez que Ajax permite desenvolvedores modificarem uma página Web em tempo real, sem a necessidade de carregamento de uma nova página, utilizando DOM para acessar, modificar, inserir ou remover elementos de uma página Web. Ajax permite também a realização de requisições HTTP assíncronas, o que possibilita que uma página seja ajustada enquanto o usuário navega nela, sem bloqueá-lo, isto é, o usuário pode navegar em uma página enquanto ajustes são processados. Outro ponto favorável a utilização de Ajax na maior parte do FAN é que todo o código JavaScript do framework é executado na máquina do cliente (*client-side*), evitando assim sobrecarregar o servidor Web.

A linguagem PHP foi utilizada no desenvolvimento de um módulo do framework que é responsável por manter, em arquivos XML, registros dos ajustes feitos na interface por cada usuário em cada página do sistema, isto é, o módulo é responsável por gravar, carregar e excluir, em arquivos, registros dos ajustes realizados na interface. A linguagem PHP foi escolhida devido a três fatores principais:

- O poder da linguagem PHP e a facilidade de utilizá-la para manipular dados em XML;
- Os ajustes de interface, feitos no lado do cliente, deveriam ser gravados em um servidor;
- O framework seria utilizado em um sistema construído com PHP; logo utilizar essa linguagem facilitaria a integração do framework com o sistema, embora ele possa ser utilizado em um servidor a parte em aplicações desenvolvidas em outras linguagens;

O código do framework desenvolvido em PHP roda em um servidor dedicado e é chamado, por código JavaScript, de maneira assíncrona. Como a chamada é assíncrona, o código JavaScript, rodando no lado do cliente, não bloqueia a navegação do usuário enquanto aguarda o fim do processamento do código PHP rodando no lado do servidor (*server-side*).

A tecnologia DOM foi utilizada tanto nos módulos desenvolvidos em JavaScript quanto em PHP. Essa tecnologia foi utilizada com JavaScript para facilitar o acesso e manipulação dos elementos da interface Web a serem ajustados no lado do cliente e DOM foi utilizado também com PHP, no lado do servidor, para facilitar o acesso e manipulação dos registros de ajustes de interface gravados em arquivos XML.

No desenvolvimento do framework foi utilizado JSON na comunicação entre partes do framework desenvolvidas em PHP e partes construídas com JavaScript. Devido às vantagens do JSON, descritas na seção 2.2, resolveu-se utilizar essa tecnologia na comunicação entre dois módulos do framework FAN, que serão apresentados na seção 3.3.

O framework utiliza serviços Web de um sistema auxiliar, responsável por gerenciar normas de sistemas, durante o processo de ajuste da interface. O framework acessa esses serviços remotos utilizando SOAP. A tecnologia SOAP foi utilizada devido à sua eficiência e por ser uma tecnologia independente de plataforma e linguagem, o que é importante tendo em vista que o framework deve se comunicar com um sistema externo desenvolvido em uma plataforma diferente da plataforma do framework.

3.3 Arquitetura

Nesta seção é apresentada e explicada arquitetura do framework FAN, descrevendo todos os seus componentes (módulos) e são apresentados também outros elementos que são essenciais para o funcionamento do framework.

O framework proposto provê funcionalidades poderosas, embora sua arquitetura, apresentada na Figura 2, não possua muitos componentes. A camada superior na Figura 2 representa a interface de um sistema Web. A camada central apresenta a arquitetura do framework e seus módulos e a camada inferior apresenta o sistema NBIC/ICE, sistema

responsável por gerenciar normas e realizar inferências sobre elas para definir possíveis mudanças na interface do sistema (camada superior). Mais adiante são explicadas em detalhes as funções de cada módulo do framework.

O FAN se comunica, via *webservices* utilizando SOAP (Newcomer, 2004), com o ambiente ICE (*Interface Configuration Environment*), que por sua vez se comunica com o sistema de gerenciamento de normas NBIC (*Norm Based Interface Configurator*), desenvolvido por Bonacin (2004, 2007) utilizando a tecnologia JEE (*Java Platform, Enterprise Edition*). O framework adapta a interface Web do sistema a cada contexto de uso do mesmo, baseando-se no Plano de Ação[10] gerado pelo ICE para cada contexto de uso do sistema. Por meio da análise do Plano de Ação gerado pelo ICE, o framework tem meios de modificar adequadamente a interface do usuário, pois tal Plano de Ação contempla diversas características do contexto de uso de determinado usuário. Dessa forma o *framework* é informado, entre outras coisas, sobre quais objetos da interface devem ser exibidos e os atributos e comportamento desejado para tais objetos.

[10] Entende-se por Plano de Ação um conjunto de ações a serem realizadas a partir da interpretação das normas pertinentes ao contexto do usuário.

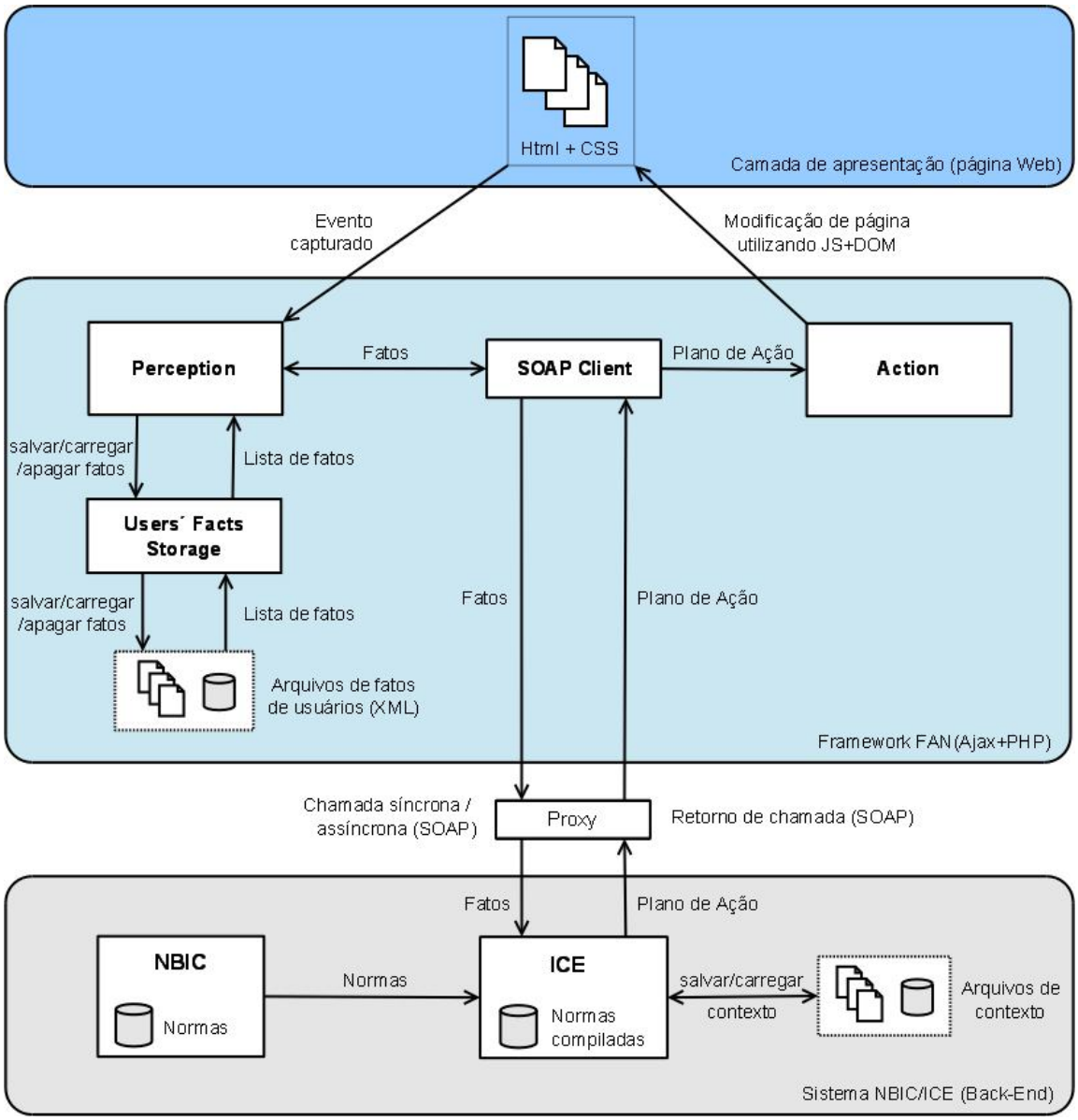


Figura 2: Arquitetura do framework FAN.

O framework FAN é composto basicamente de quatro módulos, que podem ser vistos na Figura 2: o módulo de percepção (*Perception Module*), o módulo de armazenamento de fatos de usuários (*Users' Facts Storage*), o cliente SOAP (*SOAP Client*) e o módulo de ação (*Action Module*).

Em relação às tecnologias utilizadas no framework, os módulos *Perception*, *SOAP Client* e *Action* foram desenvolvidos em Ajax. O módulo Users' Facts Storage foi desenvolvido em Ajax e PHP. Todo o código JavaScript roda na máquina do cliente (*client side*) e o código PHP é executado em um servidor com a finalidade de armazenar

arquivos gerados pelo módulo *Users's Facts Storage*. Além dessas duas tecnologias, foi utilizada, no módulo *SOAP Client*, a tecnologia SOAP para troca de informações entre o framework e o sistema NBIC/ICE e foi utilizado, no módulo *Users's Facts Storage*, XML, que foi aplicado na estrutura dos arquivos manipulados pelo módulo. Além disso, foi aplicada, nesse mesmo módulo, a tecnologia JSON na comunicação entre a parte do módulo desenvolvida em JavaScript e a parte desenvolvida em PHP.

Um Proxy foi utilizado na comunicação entre o framework, mais precisamente entre o módulo SOAP Client, e o sistema NBIC/ICE com o objetivo de contornar problemas de segurança relacionado à JavaScript. Alguns browsers não permitem a execução, em uma página, de código JavaScript armazenado em um servidor diferente do servidor que hospeda a página; com isso ocorriam problemas quando o módulo *SOAP Client* tentava acessar serviços Web do sistema NBIC/ICE, tentando acessar um arquivo com extensão *wsdl* (*Web Services Description Language*) que guarda informações dos serviços Web do NBIC/ICE, uma vez que o framework e o sistema NBIC/ICE estavam rodando em servidores distintos.

A seguir, é apresentada uma descrição detalhada de cada módulo do framework FAN, informando as funções de cada componente e como é feita a comunicação entre eles.

3.3.1 Módulo Perception

O módulo Perception recebe este nome porque sua função principal é "perceber" certos eventos na interface do sistema e iniciar o processo de ajuste de interface de acordo com informações sobre os eventos ocorridos na interface. Este módulo está ligado diretamente à interface do sistema e é acionado na ocorrência de certos eventos na interface. O designer é responsável por escolher quais eventos devem disparar o mecanismo de ajuste, uma vez que as opções de ajuste e os momentos em que a interface deve ser ajustada variam de acordo com cada sistema. A fim de ajudar a esclarecer o conceito de evento, são citados, a seguir, alguns exemplos de eventos:

- Usuário realizou *login* no sistema;
- Usuário clicou em um determinado elemento da página (ex: botão, imagem, link etc);

- Usuário passou o mouse por cima de uma imagem;
- Uma determinada página foi carregada pelo usuário;
- Usuário aumentou o tamanho da fonte de uma página dez vezes em uma mesma sessão;
- Usuário mudou o contraste da página ao realizar *login* no sistema cinco vezes consecutivas;

Este módulo também tem a função de criar fatos relacionados a eventos, pois a ferramenta ICE necessita receber fatos, relacionados a eventos na interface, como parâmetro para realizar inferências sobre normas para determinar ajustes na interface.

O módulo Perception tem também a função de requisitar ao módulo SOAP Client o envio do fato gerado para a ferramenta ICE, pois o Perception não tem acesso direto a funções do ICE. Outra função do Perception: este módulo envia requisições ao módulo Users' Facts Storage para que o mesmo guarde os fatos em arquivos específicos, associados aos usuários que geraram os eventos e associados às páginas em que os eventos ocorreram, pois diferentes páginas podem ter ajustes diferentes para diferentes usuários.

Além das funções já citadas, o componente Perception tem outras duas funções importantes, ligadas a capacidade do framework de guardar os ajustes, feitos por usuários, na interface: o módulo tem a função de requisitar ao módulo Users' Facts Storage a lista de fatos gravados em um arquivo associado ao usuário corrente e à página que ele está acessando, toda vez que uma página nova é carregada, e o módulo tem, por fim, a função de enviar para a o sistema ICE, um a um, os fatos retornados pelo módulo Users' Facts Storage. Esses fatos são enviados ao ICE toda vez que uma página é carregada para que os ajustes feitos previamente pelo usuário na página em questão sejam refeitos, de forma que a página tenha o mesmo comportamento e a mesma aparência que tinha quando o usuário a acessou ou modificou pela última vez. Já os fatos relacionados a usuários permitem que um determinado ajuste possa valer a todas as paginas do sistema, ou a um conjunto de páginas com determinada característica.

3.3.2 Módulo SOAP Client

O módulo SOAP Client tem a função de realizar a comunicação entre o framework FAN e a ferramenta ICE, por meio de requisições SOAP. Como citado anteriormente, SOAP é um protocolo leve, eficiente, para troca de dados entre sistemas.

Como o framework deve acessar serviços Web do sistema ICE utilizando SOAP foi decidido criar um módulo para ser responsável por acessar tais serviços Web, e esse módulo é o SOAP Client. Basicamente esse módulo recebe requisições do framework e as encaminha para a ferramenta ICE. Ele recebe os dados retornados pelo ICE e os envia para os módulos responsáveis por processar os dados recebidos. As requisições feitas ao SOAP Client podem ser síncronas ou assíncronas. Algumas requisições são síncronas, pois elas devem encerrar antes que outras requisições possam ser feitas.

A maior parte das requisições feitas ao SOAP Client é realizada pelo módulo Perception. O Perception envia, nas requisições, fatos relacionados a eventos na interface, que devem ser enviados para a ferramenta ICE para que ela "infira", com base em normas do sistema, ajustes que devem ser feitos na interface. Parte dessas requisições é síncrona e parte é assíncrona. As requisições feitas toda vez que uma página é carregada, a fim de realizar, na página, os ajustes gravados anteriormente pelo usuário, são realizadas sincronamente, pois por uma razão de consistência, a página deve estar ajustada conforme o usuário a acessou ou modificou no último acesso antes que ele possa interagir nessa página. As requisições feitas pelo módulo Perception ao SOAP Client para cada ajuste realizado durante a interação do usuário em uma página, isto é, enquanto ele navega em uma página, são assíncronas para não atrapalhar a navegação do usuário e por não haver problema de sincronismo dessas requisições com outras.

O código do módulo SOAP Client foi desenvolvido com base no código de um cliente SOAP, para JavaScript, desenvolvido por Matteo Casati (JavaScript SOAP Client, 2006). Este código teve que ser adaptado para que se comunicasse adequadamente com o framework FAN e com a ferramenta ICE. O código foi ainda expandido para conseguir atender a todos os requisitos de comunicação entre o framework e o ICE, principalmente no que diz respeito ao tratamento de tipos complexos de dados e exceções em SOAP.

3.3.3 Módulo Action

Este módulo possui duas funções principais: analisar as informações contidas no Plano de Ação, recebido como parâmetro de entrada, e realizar ajustes na interface do sistema de acordo com as informações contidas no Plano de Ação.

Um Plano de Ação é um arquivo XML que contem informações sobre ajustes que devem ser feitos na interface e como eles devem ser feitos. Um Plano de Ação contém ID's de objetos da interface a serem ajustados. Dependendo dos tipos de ajustes, um Plano de Ação pode também conter atributos, dos objetos que devem ser modificados, com seus respectivos valores, operações com DOM, a serem feitas com objetos da interface, ou ainda um Plano de Ação pode conter também código JavaScript a ser executado na interface. Considerando as informações que podem estar contidas em um Plano de Ação, os ajustes de interface podem ser dos seguintes tipos:

1. Modificação de um atributo de um elemento;
2. Operação do DOM usando um só elemento;
3. Operação do DOM usando dois elementos;
4. Execução de código JavaScript livre, incluindo chamadas a operações JavaScript da aplicação;

O módulo Action precisa analisar um Plano de Ação para saber o tipo de cada ajuste a ser realizado na interface e saber quais objetos de interface estão envolvidos em cada operação de ajuste. Durante a análise do Plano de Ação, a cada ajuste encontrado o módulo Action verifica o tipo do ajuste. Se o ajuste for do tipo 1, 2 ou 3, listados acima, o módulo acessa, via DOM, os elementos da interface envolvidos na operação de ajuste, que estão listados no Plano de Ação, e modifica seus atributos de acordo com os atributos e valores encontrados no Plano de Ação, se o ajuste for do tipo 1, ou executa operações do DOM com esses elementos se o ajuste for do tipo 2 ou 3. Se o ajuste encontrado no Plano de Ação for do tipo 4, o Action simplesmente executa, na interface, o código JavaScript contido, no ajuste em questão, no Plano de Ação. Esse código JavaScript é livre, podendo portanto ser o que designer de interface quiser. Esse código pode, por exemplo, chamar funções JavaScript já contidas em uma página ou pode executar operações novas, diferentes, não contidas originalmente na página. É importante ressaltar

que cabe aos designers/desenvolvedores da aplicação hospedeira definir caso a caso qual o tipo de ajuste mais adequado a seus requisitos e defini-los via a ferramenta NBIC.

3.3.4 Módulo Users' Facts Storage

O módulo *Users' Facts Storage* possui três funções fundamentais: gravar, ler e apagar fatos em arquivos XML. Toda vez que o módulo *Perception* gera um fato ele envia esse fato para o módulo *Users' Facts Storage*, que o grava em um arquivo XML, que é associado ao usuário que gerou um evento na interface e associado à página em que foi gerado tal evento. Esse arquivo é importante, pois toda vez que uma página do sistema for carregada por um usuário o módulo *Perception* irá requisitar ao módulo *Users' Facts Storage* os fatos, gravados em arquivos, associados a esse usuário e à página que ele está visitando, a fim de iniciar, para cada fato gravado, todo o processo de ajuste de página realizado pelo framework para alterar a interface do sistema de forma que ela fique com a mesma aparência e o mesmo comportamento que tinha na última vez que o usuário acessou ou modificou a página. Quando um ajuste não for mais necessário, isto é, quando o usuário remover o ajuste feito anteriormente em um objeto da interface, deixando-o com a aparência e comportamento originais (padrões) do site, o fato relacionado a tal ajuste deve ser apagado do arquivo associado. Desta forma, na próxima vez que o usuário recarregar a página do objeto em questão, ele continuará com sua aparência original, isto é, não será ajustado. Logo, este módulo deve ser capaz de apagar fatos gravados em arquivos.

Os arquivos XML gerados pelo módulo Users's Facts Storage seguem um padrão bem definido em relação às suas estruturas e nomes. Como cada arquivo deve guardar uma lista de fatos, relacionados a um determinado usuário, e cada fato é constituído por um conjunto de predicados, sendo cada predicado constituído por um *affordance* e um conjunto de elementos, foi determinada a estrutura de cada arquivo XML, como é mostrado na Figura 3 a seguir.

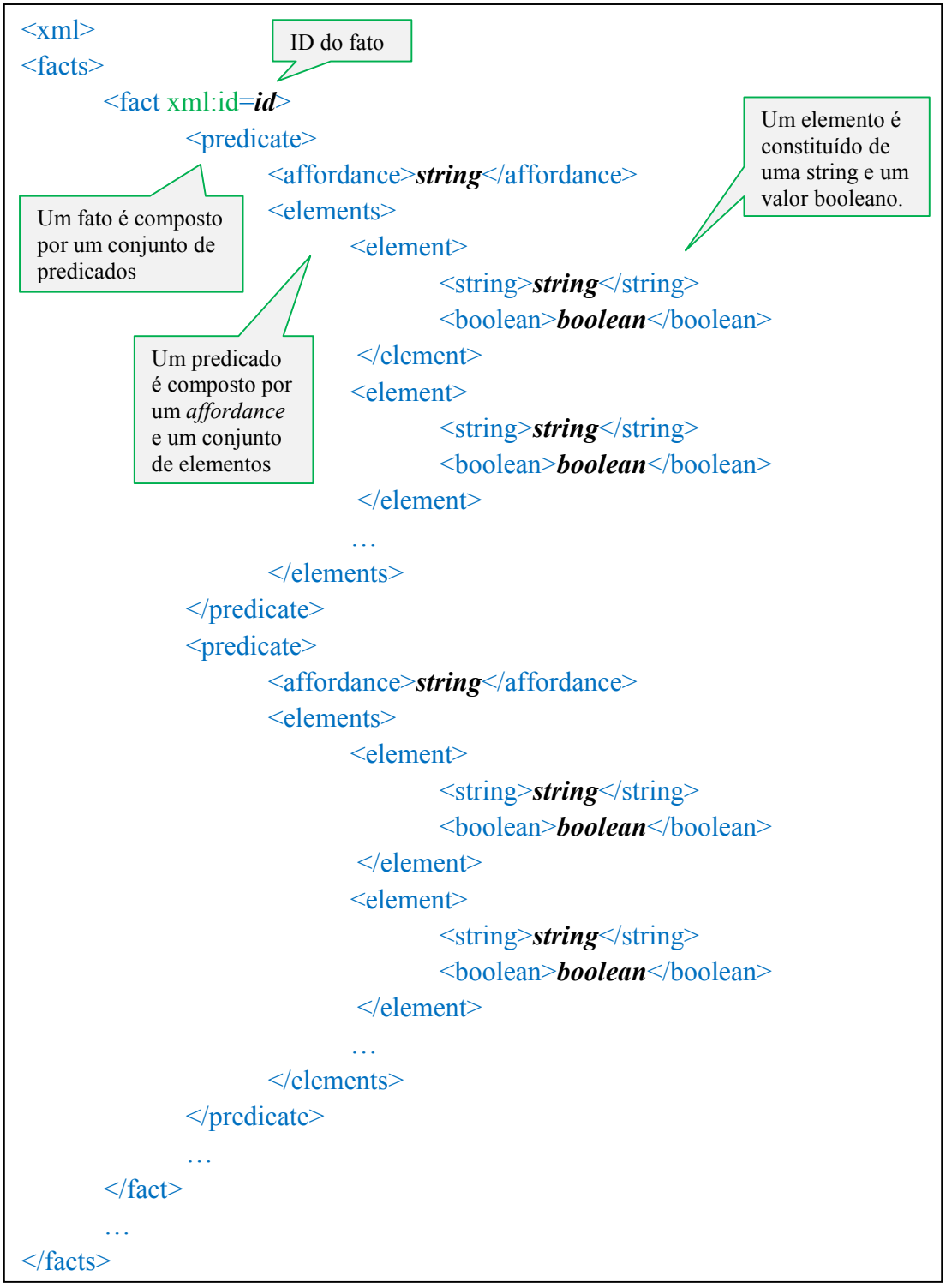


Figura 3: Estrutura de um arquivo XML que guarda fatos.

O nome de cada arquivo deve ser associado a cada usuário e a cada página do sistema, então o nome de cada arquivo foi construído da seguinte forma:

<ID do usuário>-<ID da página em que o usuário realizou um ajuste>.xml

Se um usuário tem ID 55 e realiza um ajuste na página ou nó com ID 273, o nome do arquivo gerado será “55-273.xml”. Usando o ID do usuário, que geralmente é um valor único dentro do sistema, no nome do arquivo garante-se que ajustes feitos por diferentes usuários em uma mesma página não serão gravados no mesmo arquivo, e utilizando o ID da página é assegurado que ajustes feitos por um mesmo usuário em diferentes páginas não serão gravados em um mesmo arquivo. Porém, alguns ajustes realizados pelo usuário podem ser aplicados em todas as páginas do sistema, não só em uma página específica. Para resolver essa questão, foi definido que ajustes feitos para serem aplicados em todas as páginas devem ser gravados em um arquivo com nome diferenciado, construído da seguinte forma:

<ID do usuário>-global.xml

Se o usuário citado anteriormente, com ID 55, realiza um ajuste na interface que deve valer para todas as páginas do sistema, o que é chamado de um ajuste global, esse ajuste será gravado em um arquivo com o nome “55-global.xml”. Fica a cargo do designer da interface a tarefa de decidir se um ajuste será sempre global ou específico para cada página ou ele pode também, se desejar, deixar a cargo do usuário realizar tal escolha.

Em relação às linguagens de programação usadas no desenvolvimento, vale lembrar que este módulo foi desenvolvido com Ajax e PHP, pois uma parte dele, desenvolvida em Ajax, roda na máquina do cliente (*client side*), realizando requisições de leitura, gravação e exclusão de fatos em arquivos e a outra parte, desenvolvida em PHP, roda no servidor do sistema (*server side*), onde foi testado o framework, manipulando arquivos XML.

3.4 Funcionamento

Toda vez que uma página do sistema é carregada o módulo Perception cria um novo contexto ao fazer uma chamada síncrona, via SOAP Client, ao ICE. Um arquivo de contexto armazena, no servidor ICE, informações como o ID do usuário, o nome ou ID do sistema e ID da sessão do usuário. Como esse arquivo deve estar criado antes que qualquer processo de ajuste seja executado, a chamada realizada pelo Perception ao ICE deve ser síncrona (passo 1 da Figura 4). Após o arquivo de contexto ser criado ou

carregado pelo ICE, as normas do sistema, armazenadas no NBIC, são carregadas no arquivo de contexto por meio de uma nova chamada, novamente síncrona, do Perception ao ICE (passo 2 da Figura 4). Após as normas serem carregadas no arquivo de contexto, as operações de ajuste passam a ser válidas. Essas etapas podem ser vistas na Figura 4.

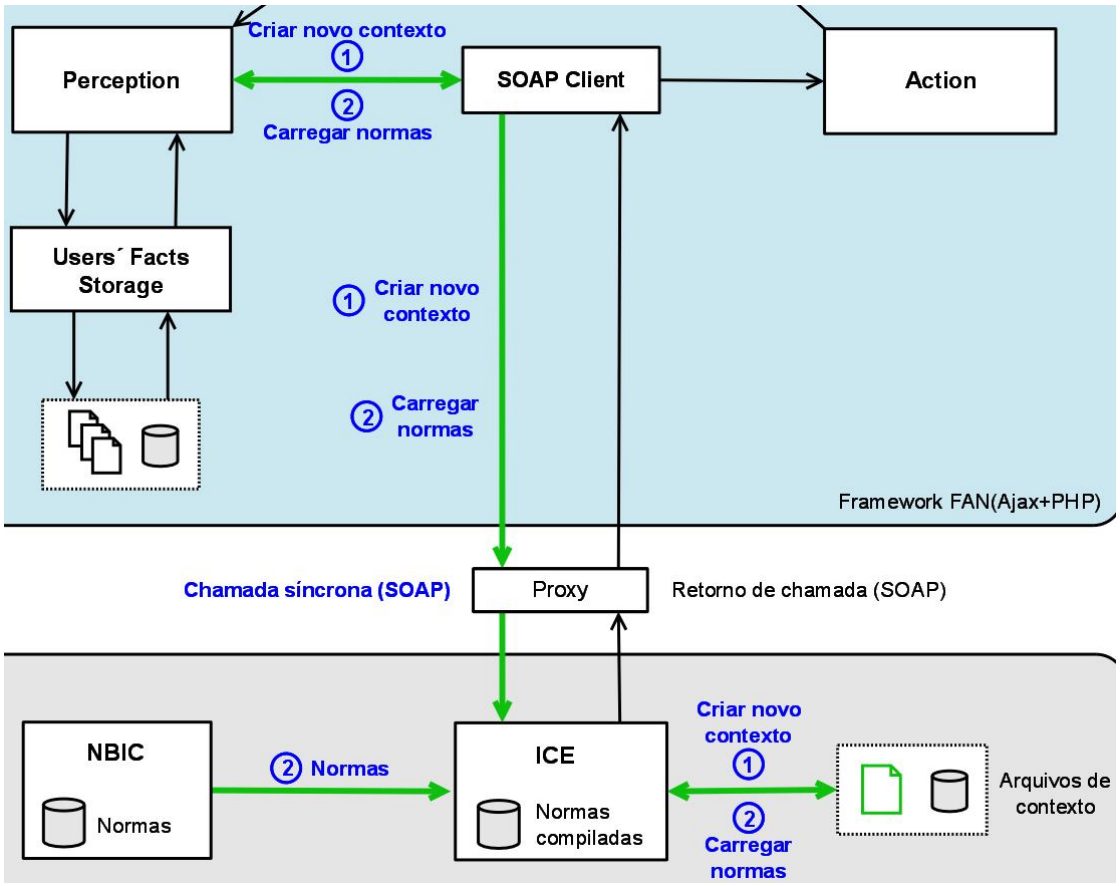


Figura 4: O Perception faz uma requisição (número 1 na figura) ao ICE para criar um novo arquivo de contexto, seguida por outra requisição (número 2) para carregar normas no arquivo de contexto criado.

Assim que as normas do sistema são carregadas no arquivo de contexto, o módulo Perception requisita ao módulo Users' Facts Storage fatos gravados em arquivo XML, relacionados ao usuário corrente e à página ou nó[11] atual, como pode ser visto na Figura 5.

[11] Em alguns sistemas usa-se o termo "nó" para representar uma página.

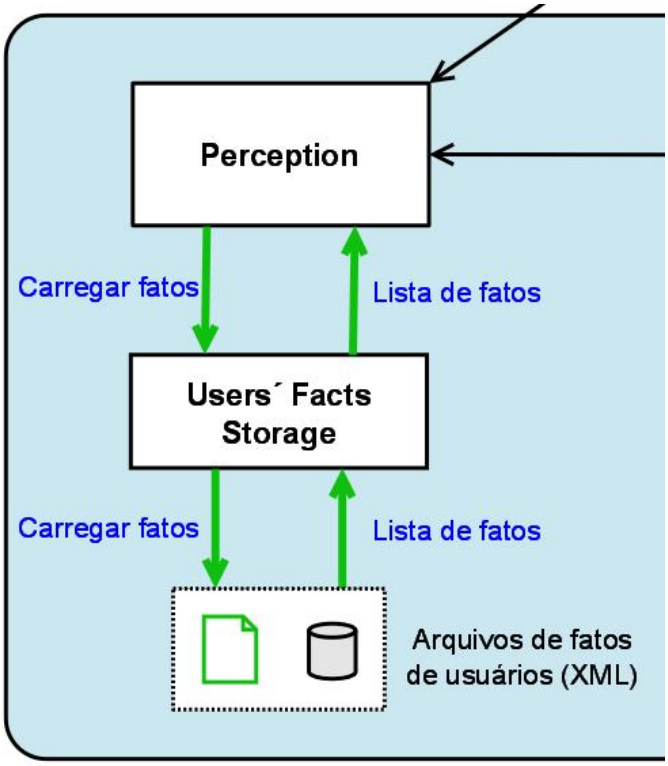


Figura 5: Módulo Perception requisitando ao Users' Facts Storage os fatos do usuário.

Se houver um arquivo contendo fatos associados ao usuário e à página atual, isto é, se houver um arquivo com nome composto pelo ID do usuário e ID da página atual (o que significa que alguma vez o usuário já realizou ajuste na página atual), o módulo Users' Facts Storage abre esse arquivo, lê os fatos contidos nele e os insere em um vetor de fatos. Se tal arquivo não existir ou se não houver fatos gravados no arquivo, um vetor vazio é criado. O vetor de fatos, então, é enviado para o módulo Perception.

Se o módulo Perception receber um vetor vazio retornado pelo Users' Facts Storage, ele ignora esse vetor. Caso o vetor não esteja vazio, o Perception irá iniciar o processo de ajuste da interface, descrito a seguir, para cada fato contido no vetor, de forma que a interface tenha a mesma aparência e o mesmo comportamento que teve na última vez que esse usuário acessou ou modificou a página corrente.

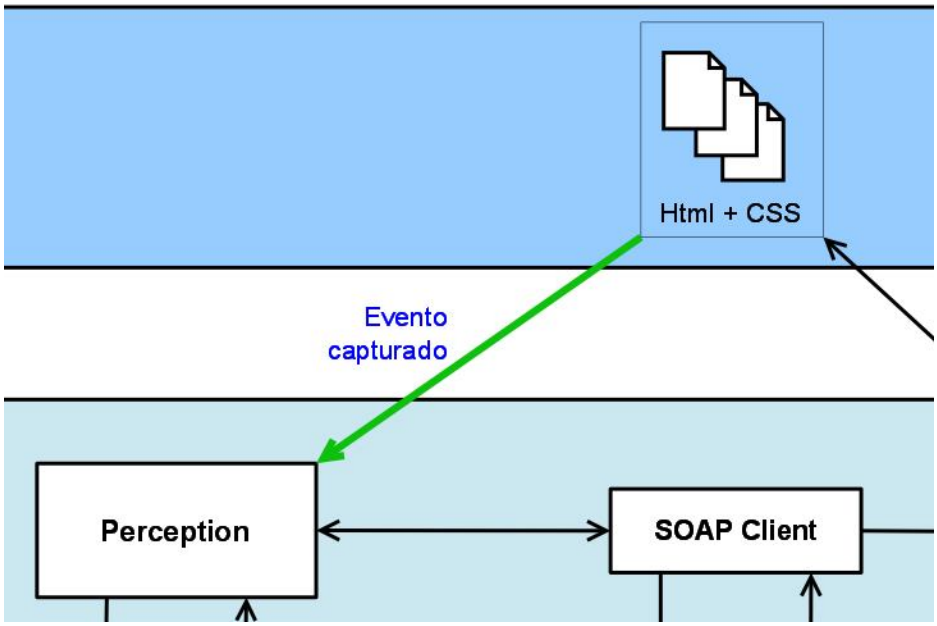


Figura 6: Evento capturado pelo módulo Perception.

Quando um evento na interface (ex: usuário clicou em determinado botão da página) é capturado pelo módulo Perception, como exibido na Figura 6, esse módulo cria predicados relacionados ao contexto de uso da interface. Após criar os predicados, o módulo de percepção acessa um serviço do ICE, por meio de uma chamada assíncrona via SOAP Client, para gerar e enviar, para o ICE, um fato, usando os predicados previamente criados, uma vez que um fato é composto por um conjunto de predicados. Como exemplo de um fato, é possível citar, em linguagem de alto nível: "usuário José no sistema Vilanarede prefere menus circulares". Além disso, o Perception faz uma chamada ao módulo Users' Facts Storage para gravar o novo fato em um arquivo XML, com nome associado ao ID do usuário e ID da página atual. Ambas as requisições são apresentadas na Figura 7.

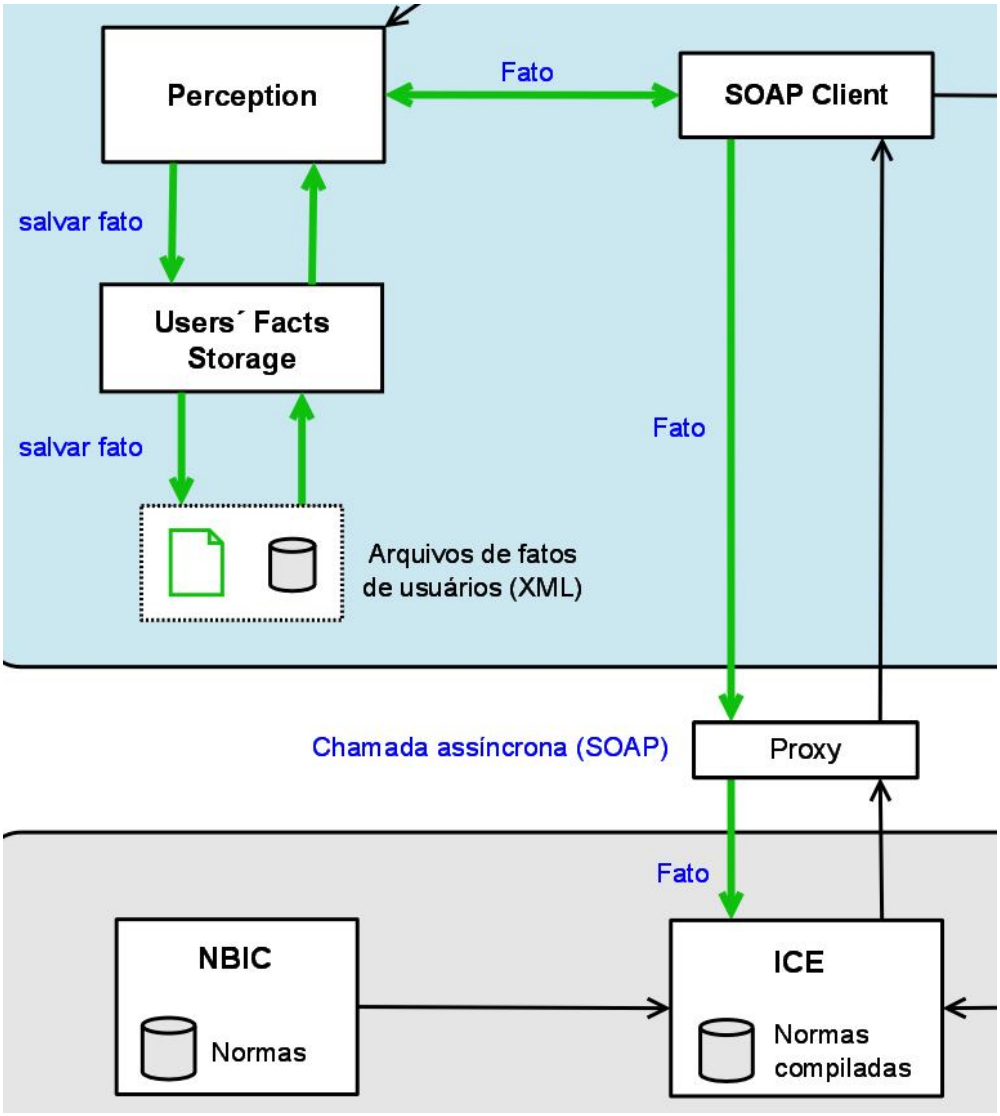


Figura 7: Perception realizando chamadas assíncronas ao ICE e Users' Facts Storage para, respectivamente, adicionar um novo fato ao arquivo de contexto e gravar o novo fato em arquivo XML.

Ao receber um fato, a ferramenta ICE começa a realizar inferências, ao analisar normas do sistema (normas compiladas a partir das normas armazenadas pelo NBIC), o arquivo de contexto e o fato recebido, para construir um Plano de Ação. Como exemplo de norma, em linguagem natural, pode-se dizer: “sempre que um usuário estiver no sistema Vilanarede, se o usuário prefere menu circular então o sistema Vilanarede deve mostrar menu circular”. Um Plano de Ação é, conforme explicação anterior, um arquivo XML que pode conter IDs de objetos da interface que devem ser ajustados, atributos que devem ser modificados, com seus respectivos novos valores, além de poder conter código JavaScript ou operações com DOM para serem executadas na interface. O Plano de Ação

construído pelo ICE é, então, enviado ao módulo SOAP Client, que por sua vez repassa esse Plano de Ação para o módulo Action, como é mostrado na Figura 8.

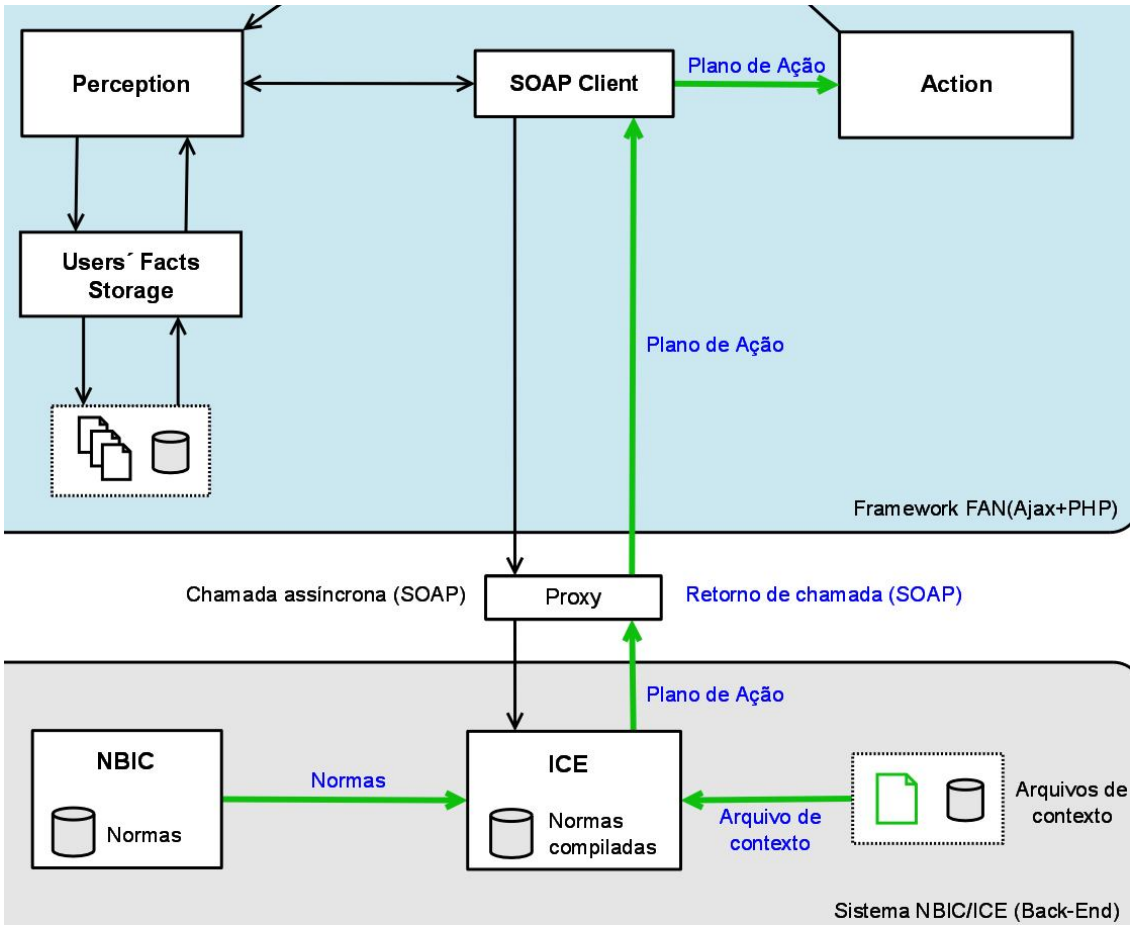


Figura 8: ICE gerando um Plano de Ação e o enviando para o módulo Action.

O último módulo do framework FAN a ser acionado no processo de ajuste é o módulo Action. Ao receber um Plano de Ação o Action executa uma análise no arquivo XML (Plano de Ação) para verificar os ajustes que deverão ser feitos na interface, os tipos dos ajustes, os atributos ou operações envolvidas e conhecer os IDs dos objetos da interface que serão modificados, inseridos ou removidos. O módulo realiza um *loop* sobre os ajustes a serem feitos na interface e, a cada iteração, ele realiza modificações na interface de acordo com as informações contidas no Plano de Ação, como é apresentado na Figura 9. Como exemplo de plano de ação, em linguagem natural, é possível dizer: "o sistema deve executar a função JavaScript 'exibirMenuCircular()'".

Para facilitar o entendimento, são citados dois exemplos de ajustes que podem estar em um Plano de Ação. Como primeiro exemplo de ajuste encontrado em um determinado

Plano de Ação, tem-se que um objeto de interface com ID “objeto2” deve ser inserido dentro de outro objeto com ID “objeto1”, utilizando a função *appendChild()* do DOM. O módulo Action, ao encontrar esse ajuste no Plano de Ação, irá acessar os dois objetos de interface envolvidos no ajuste, utilizando seus IDs e a função *getElementsById()* do DOM, e irá chamar a função *appendChild()*, passando os dois objetos como parâmetro. Em um segundo cenário, há um ajuste que contém um código JavaScript a ser executado na interface. Encontrando esse ajuste ao analisar um Plano de Ação, o Action irá simplesmente executar o código JavaScript, contido no Plano de Ação, utilizando a função *eval()* do JavaScript.

Caso um ajuste envolva somente a modificação de um atributo de um elemento da interface, o módulo Action simplesmente acessa tal objeto, utilizando o ID do objeto e novamente a função *getElementsById()* do DOM, e designa um novo valor para um determinado atributo do objeto, de acordo com o par atributo-valor contido no Plano de Ação.

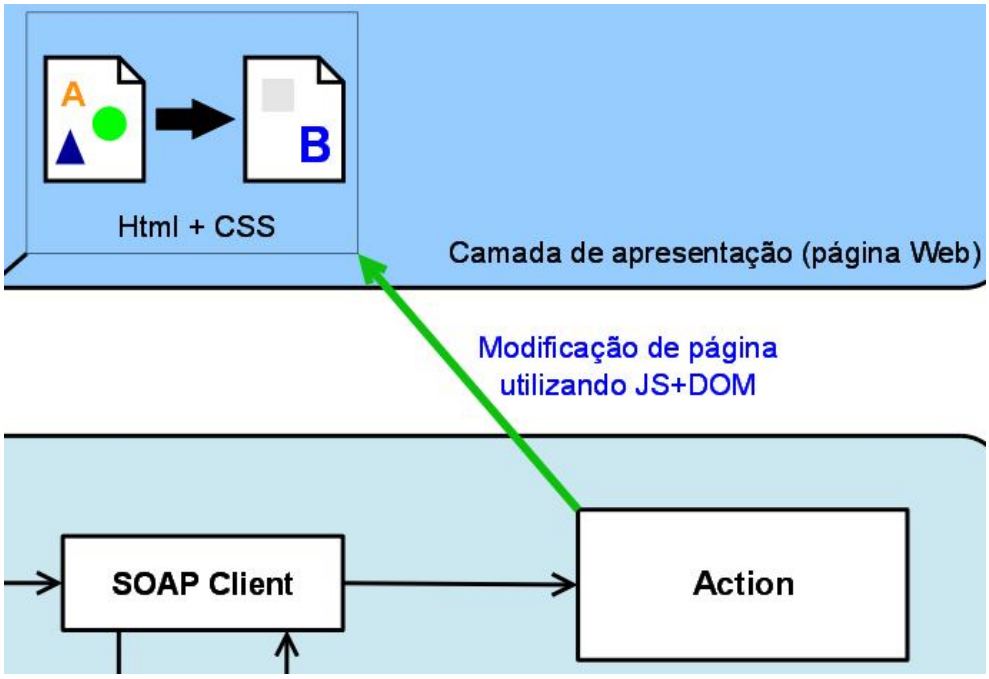


Figura 9: Módulo Action ajustando interface de acordo com um Plano de Ação.

Após o módulo Action iterar sobre todos os ajustes contidos em um Plano de Ação o processo de ajuste para um único fato é concluído e todo o processo de ajuste é iniciado

novamente para outro fato que esteja sendo ou venha a ser gerado ou manipulado pelo módulo Perception.

3.5 Funcionalidades

Apesar de ter uma arquitetura com um número relativamente pequeno de módulos, o framework FAN possui funcionalidades que viabilizam a flexibilidade almejada em interfaces. Ele foi desenvolvido de tal forma que as possibilidades de ajustes de uma interface ficam, essencialmente, limitadas às escolhas do designer da interface do sistema e à dificuldade em se desenvolver, utilizando JavaScript ou DOM, determinados ajustes mais complexos. Caso novos atributos ou operações sejam incorporados às tecnologias HTML, XHTML e CSS o framework FAN poderá suportar tais novidades sem necessidade de modificações no seu código fonte. Como é difícil listar todos os ajustes que podem ser feitos utilizando o framework, é listado abaixo um pequeno conjunto de tipos de ajustes significativos que o framework permite:

- Mudança de atributos de estilo de objetos;
- Realizar *drag-and-drop* com elementos da interface;
- Mudar a posição de um objeto via operações com DOM;
- Modificar atributos relacionados à posição e tamanho de um objeto;
- Inserir ou remover objetos da interface;
- Exibir ou esconder elementos em uma página;
- Habilitar ou desabilitar objetos diversos, em especial botões;
- Ajuste por meio de execução de código JavaScript contido em um Plano de Ação ou por meio chamada de função JavaScript já existente na página;
- Modificar automaticamente um objeto por meio de ações repetitivas do usuário.

A fim de ilustrar três possibilidades de ajustes que o FAN é capaz de oferecer, citadas no parágrafo anterior, na Figura 10 é apresentada uma tela de um sistema de rede

social, onde usuários inserem anúncios diversos classificados por categorias, juntamente com algumas opções de personalização da página que o framework FAN pode permitir, como deslocar um elemento gráfico para uma nova posição na página, ocultar um elemento e mudar a estrutura de uma página ao mudar a ordem de apresentação de seus elementos.

Figura 10: Tela de um sistema de rede social juntamente com três opções de ajustes de interface que o framework FAN pode permitir.

A Figura 11 apresenta o resultado da realização dos ajustes, propostos na Figura 10, realizados por usuário do sistema. A imagem apresenta uma página ajustada, com aparência e comportamento diferentes da página original (Figura 10).

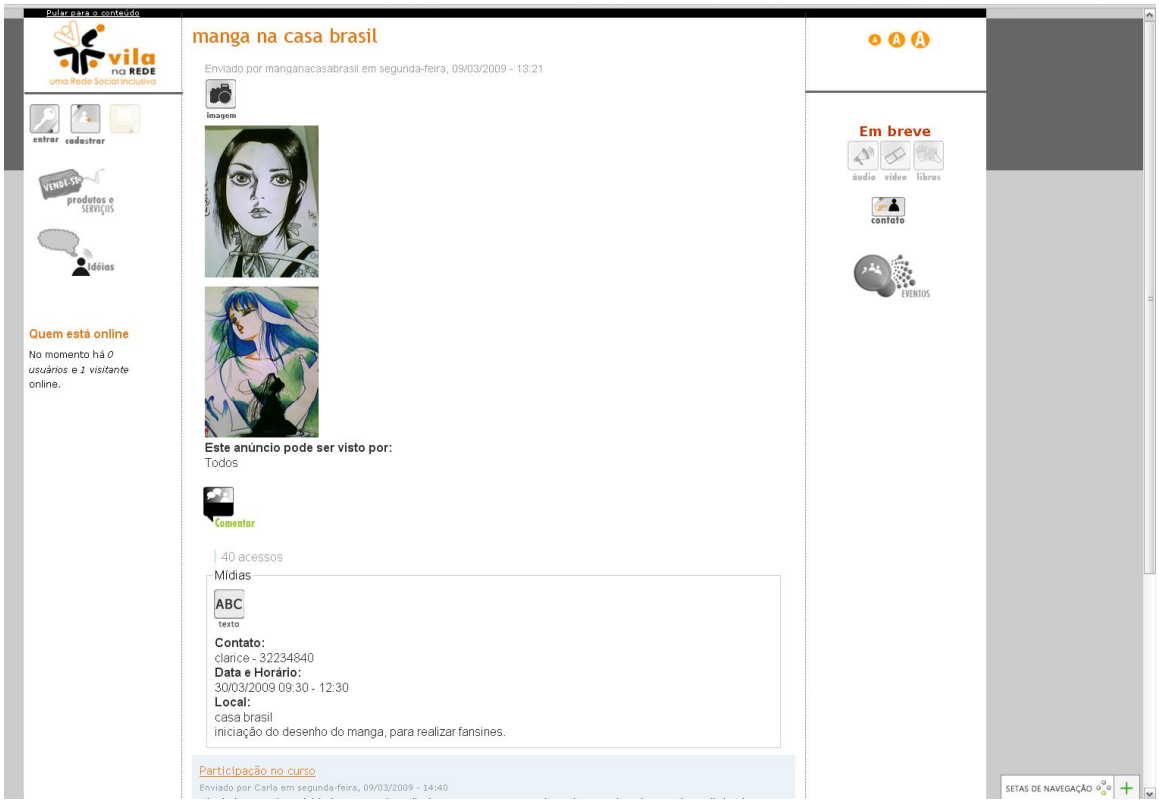


Figura 11: Página com aparência e comportamento alterados após a realização de ajustes permitidos pelo Framework.

A seguir é apresentado um exemplo de mais uma funcionalidade do framework: modificar automaticamente objetos por meio de ações repetitivas do usuário. Em uma situação na qual um usuário clica um número consecutivo de vezes em áreas não clicáveis, ao tentar clicar ou selecionar um determinado objeto, o framework pode aumentar o tamanho dos elementos da interface para ajudar na interação do usuário, como pode ser visto na Figura 12, sempre que houver uma norma que estabeleça que o tamanho dos elementos deve ser aumentado se o usuário tiver dificuldade para navegar na página ou tenha algum problema de visão.

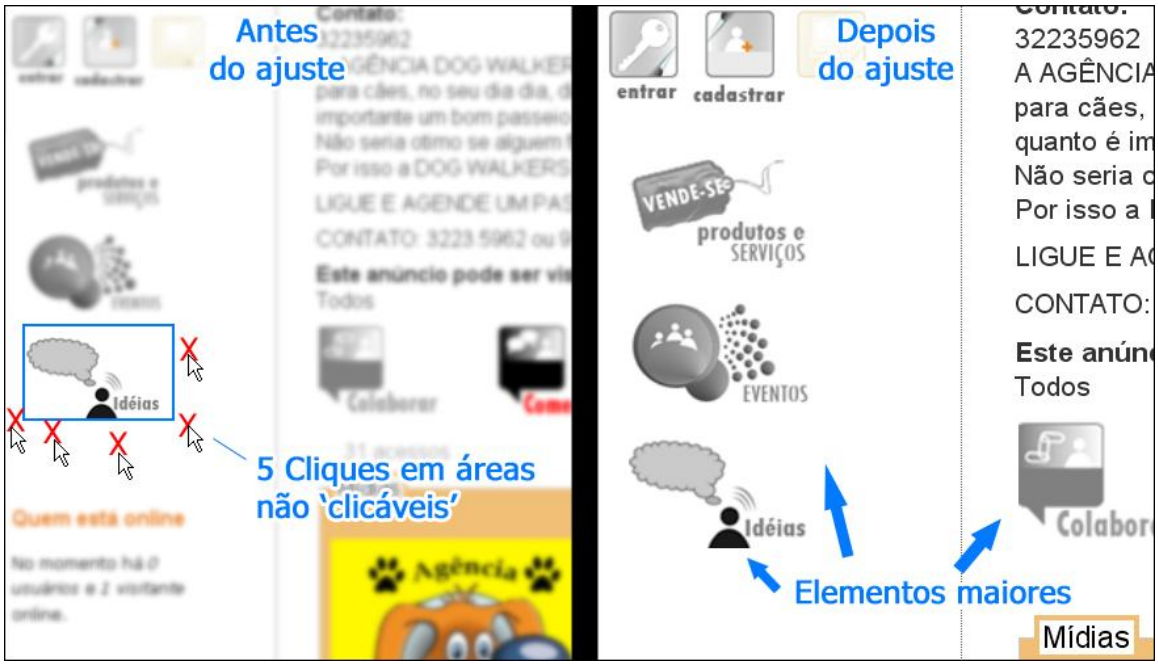


Figura 12: Framework modificando os tamanhos dos elementos por haver uma norma no sistema que determina a execução de tal ação.

Embora o FAN possa ser utilizado para desenvolver funcionalidades e comportamentos sofisticados, a complexidade de seu uso se concentra nas decisões do designer da interface do sistema e dos desenvolvedores da interface, tendo em vista que o designer da interface deve definir normas do sistema que irão reger as opções de ajuste da interface e os desenvolvedores da interface devem desenvolver, em JavaScript, os ajustes que sejam específicos às necessidades do sistema.

Capítulo 4

Validação da Solução do Framework

Neste capítulo é explicado como o framework FAN foi validado com usuários finais, além de serem apresentados os resultados dos testes de validação e uma análise sobre os dados coletados. Detalhes de todo o ambiente de instalação do framework e do ambiente onde a ferramenta foi testada são apresentados também neste capítulo. Por fim, é realizada uma discussão sobre os pontos positivos e negativos do framework.

4.1 Ambientes de testes

Este projeto é parte de um contexto de pesquisa mais amplo, o projeto e-Cidadania: Sistemas e Métodos na Constituição de uma Cultura Mediada por Tecnologias de Informação e Comunicação (Baranauskas, 2007). Através do projeto e-Cidadania foi possível aplicar e testar o framework FAN em um sistema com usuários reais. O Vilanarede [12] é um sistema de rede social desenvolvido no contexto do projeto e-Cidadania com o objetivo de ajudar na promoção de uma cultura digital entre diversas categorias de usuários, envolvendo usuários com diferentes interesses e necessidades.

A aplicação do FAN no sistema Vilanarede tinha por objetivo facilitar a criação de uma interface flexível para o sistema Vilanarede, com opções de ajuste baseadas em requisitos extraídos dos usuários e nas decisões do designer da interface. O uso do framework no sistema Vilanarede deu aos usuários a possibilidade deles modificarem

[12] http://www.vilanarede.org.br/

alguns elementos da interface do sistema, de acordo com seus interesses ou necessidades, de maneira direta (ajuste realizado pelo usuário) ou indireta (ajuste realizado automaticamente pelo sistema após analisar dados do usuário).

4.1.1 Projeto e-Cidadania

O projeto e-Cidadania, financiado pela Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP) e pela Microsoft Research, tem por objetivo estudar e propor soluções aos desafios do design da interação e interface de usuário em sistemas relacionados ao contexto do exercício de cidadania, a partir do desenvolvimento de ações conjuntas com a comunidade-alvo.

Tais estudos e propostas de soluções iniciaram-se com o desenvolvimento de um sistema de Rede Social Inclusiva (RSI) chamado Vilanarede. O desenvolvimento desse sistema, bem como os estudos e propostas de soluções para os desafios do design da interação e interface de usuário, utilizaram, como referencial teórico, conceitos e métodos da Semiótica Organizacional de forma articulada aos princípios do Design Universal ou Design para Todos.

Os resultados obtidos no projeto contribuíram com pesquisas em diversas áreas, como interfaces flexíveis, foco deste trabalho, além de investigar sobre a constituição de uma cultura digital na comunidade alvo do projeto e desenvolvimento de aplicações que fazem sentido para essa comunidade.

4.1.2 Sistema Vilanarede

O Vilanarede é um sistema de rede social desenvolvido no contexto do projeto e-Cidadania com o objetivo de facilitar o acesso à cultura digital entre diversas categorias de usuários, envolvendo usuários com diferentes interesses e necessidades. Esse sistema foi desenvolvido sobre a plataforma Drupal[13], um sistema de gerenciamento de conteúdo – CMS (sigla em inglês para "*Content Management System*") – de código livre. A plataforma Drupal facilitou o desenvolvimento do Vilanarede ao fornecer algumas funcionalidades como, entre outras, um sistema de gerenciamento de usuários e um banco de dados.

[13] http://drupal.org/

O funcionamento do Vilanarede gira em torno de anúncios postados por usuários. Usuários do sistema postam anúncios diversos que são divididos em três categorias: idéias, serviços e produtos. A interação entre usuários ocorre, predominantemente, em torno dos anúncios. Usuários podem postar comentários ou perguntas em anúncios ou podem colaborar com o conteúdo de um anúncio ao postar informações adicionais sobre o mesmo. Um dos pontos mais interessantes desse sistema é fornecer aos usuários a possibilidade de interagirem utilizando múltiplas mídias (texto, imagem, som, vídeo e LIBRAS).

Além de postarem anúncio no sistema Vilanarede, usuários podem conversar entre si por meio de uma ferramenta presente no sistema, que permite conversas utilizando mensagens de texto, som ou vídeo.

4.1.3 Oficinas Participativas

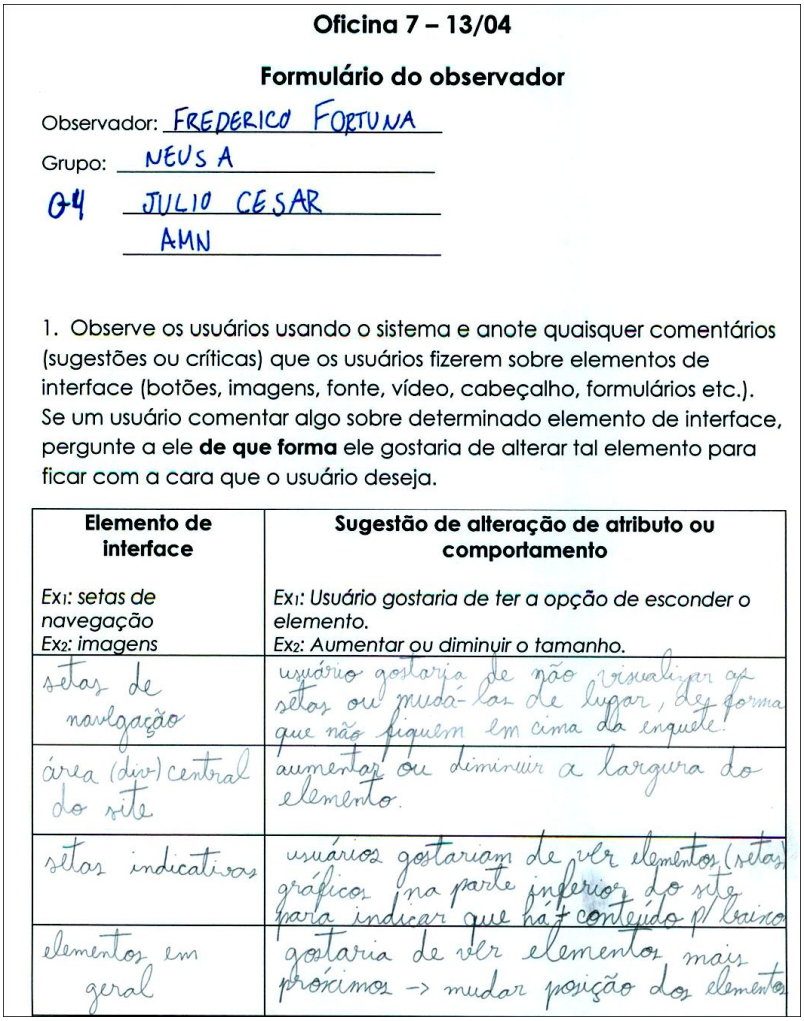
Dentro do contexto do projeto e-Cidadania foram realizadas diversas Oficinas Participativas com usuários do sistema Vilanarede. Nessas oficinas ocorriam encontros com usuários que representam o público alvo do projeto e-Cidadania e nelas esses usuários executavam atividades diversas, que geraram dados importantes para pesquisas inseridas no projeto e-Cidadania. Elas também tinham o propósito de guiar o desenvolvimento do sistema Vilanarede, por meio de atividades envolvendo extração de requisitos e coleta de opiniões ou sugestões (*feedbacks*) de usuários em relação ao sistema Vilanarede.

Foram realizadas onze Oficinas Participativas no projeto e-Cidadania, sendo as primeiras mais focadas na concepção do sistema Vilanarede, com atividades envolvendo artefatos da Semiótica Organizacional como quadro de avaliação e *stakeholders*, sendo tais artefatos usados posteriormente na extração e análise de requisitos e na definição de casos de uso do sistema. Além disso, nas primeiras oficinas foram coletados *feedbacks* dos usuários sobre protótipos do sistema. As oficinas intermediárias e as últimas tinham como foco principal atividades que permitiram a coleta de dados importantes para uso em pesquisas inseridas no projeto e-Cidadania.

As oficinas sete, nove e dez colaboraram de maneira direta com esta pesquisa. Essas oficinas foram elaboradas, com a participação do pesquisador, para geração e coleta de dados úteis para esta pesquisa. Os dados coletados nessas oficinas foram úteis na

determinação de normas do sistema Vilanarede e esses dados foram levados em conta nas atividades de desenvolvimento e testes do framework FAN.

Na oficina sete usuários do sistema Vilanarede deveriam realizar diversas tarefas como cadastrar um novo anúncio no sistema e inserir um comentário em um anúncio já existente. Enquanto os usuários realizavam as tarefas, eles foram observados e alguns dados foram gerados a partir das observações. Os dados foram coletados e registrados em formulários como o exibido na Figura 13. Estes formulários foram utilizados na definição de algumas opções de ajuste para o sistema Vilanarede, como, por exemplo, fornecer ao usuário um meio dele escolher o tipo do menu principal do sistema, podendo ser um menu com estrutura linear (menu linear) ou com estrutura circular (menu circular).



Oficina 7 – 13/04

Formulário do observador

Observador: FREDERICO FORTUNA

Grupo: NEUS A

04 JULIO CESAR

AMN

1. Observe os usuários usando o sistema e anote quaisquer comentários (sugestões ou críticas) que os usuários fizerem sobre elementos de interface (botões, imagens, fonte, vídeo, cabeçalho, formulários etc.). Se um usuário comentar algo sobre determinado elemento de interface, pergunte a ele **de que forma** ele gostaria de alterar tal elemento para ficar com a cara que o usuário deseja.

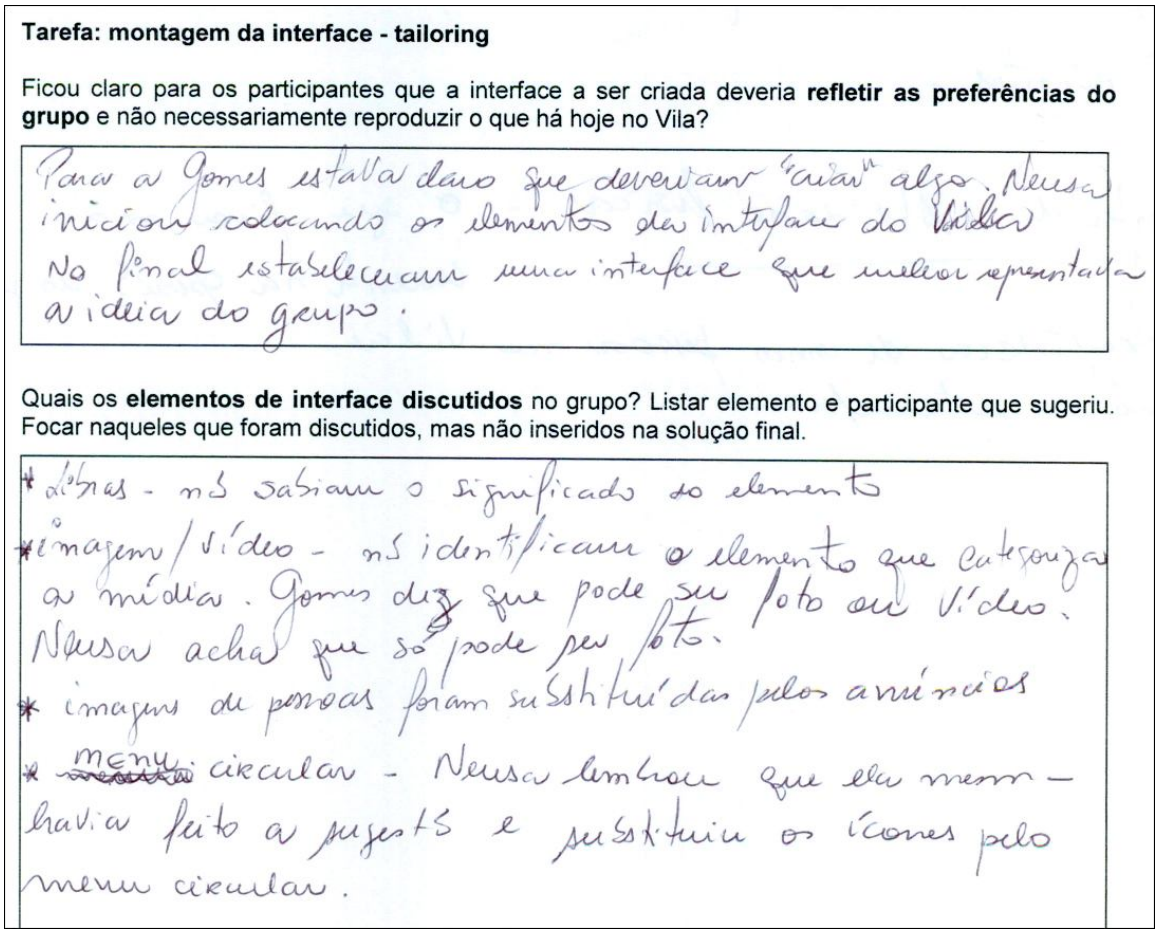
Elemento de interface *Ex1: setas de navegação* *Ex2: imagens*	**Sugestão de alteração de atributo ou comportamento** *Ex1: Usuário gostaria de ter a opção de esconder o elemento.* *Ex2: Aumentar ou diminuir o tamanho.*
setas de navegação	usuário gostaria de não visualizar as setas ou mudá-las de lugar, de forma que não fiquem em cima da enquete.
área (div) central do site	aumentar ou diminuir a largura do elemento.
setas indicativas	usuários gostariam de ver elementos (setas) gráficos na parte inferior do site para indicar que há + conteúdo p/ baixo
elementos em geral	gostaria de ver elementos mais próximos -> mudar posição dos elementos

Figura 13: Formulário utilizado na oficina sete, preenchido por um dos observadores.

A oficina nove tinha como objetivo principal realizar uma avaliação ampla do sistema Vilanarede, a fim de testar o funcionamento de diversas ferramentas do sistema e, em um segundo momento, elicitar normas de sistema relacionadas a ajustes de interface e a ordem da apresentação de anúncios no sistema. Uma das tarefas dos usuários, realizada em grupos, consistia em elaborar, em um painel com o desenho de uma janela vazia de um navegador, uma interface gráfica para o sistema Vilanarede. Para elaborar tal interface os usuários poderiam colar, no painel, elementos de interface pré-determinados ou desenhar, com caneta, os elementos que precisassem, conforme a Figura 14 que mostra uma usuária montando um dos painéis. Os painéis montados na oficina nove contribuíram, assim como os dados coletados na oficina sete, na definição de alguns cenários possíveis de ajuste da interface como, por exemplo, a opção de escolher uma estrutura linear ou circular para o menu principal do sistema e esconder ou exibir as setas de navegação do sistema. Na Figura 14 é possível ver uma usuária sugerindo o uso de um menu circular, localizado à esquerda no painel.

Figura 14: Usuária montando, em um painel, uma nova interface gráfica para o sistema Vilanarede, durante atividade na oficina nove.

Durante o processo de montagem dos painéis foram registrados, em formulários como o da Figura 15, informações importantes sobre o processo de criação de uma nova interface, realizado por cada grupo de usuários usando os painéis citados anteriormente. Os dados registrados ajudaram em uma melhor definição de quais opções de ajustes de interface, propostos pelos usuários, seriam desenvolvidas.



Tarefa: montagem da interface - tailoring

Ficou claro para os participantes que a interface a ser criada deveria **refletir as preferências do grupo** e não necessariamente reproduzir o que há hoje no Vila?

Para a Gomes estava claro que deveriam "criar" algo. Nessa iniciou colocando os elementos da interface do Vila. No final estabeleceram uma interface que melhor representava a ideia do grupo.

Quais os **elementos de interface discutidos** no grupo? Listar elemento e participante que sugeriu. Focar naqueles que foram discutidos, mas não inseridos na solução final.

* letras - não sabiam o significado do elemento
* imagem/vídeo - não identificam o elemento que categoriza as mídias. Gomes diz que pode ser foto ou vídeo. Nessa acha que só pode ser foto.
* imagens de pessoas foram substituídas pelos avatares
* ~~menu~~ circular - Nessa lembrou que ela mesma havia feito a pesquisa e substituiu os ícones pelo menu circular.

Figura 15: Formulário preenchido por um dos observadores na oficina nove.

A oficina dez teve como foco o teste de três novas ferramentas, sendo uma delas o framework FAN que foi incorporado ao sistema Vilanarede. Essa oficina foi de grande importância para esta pesquisa, pois possibilitou a avaliação do funcionamento do framework FAN, rodando no servidor de produção do sistema Vilanarede, e possibilitou também a avaliação do funcionamento das opções de ajuste da interface, que foram desenvolvidas para o sistema Vilanarede, ao serem utilizadas por usuários do sistema. Entre as diversas demandas de ajustes coletadas nas oficinas anteriores, foram escolhidas duas opções de ajuste que ao mesmo tempo fossem significativas aos usuários e

possibilitassem a avaliação da ferramenta. Foram, portanto, oferecidas aos usuários na oficina dez: a possibilidades de alterar, de forma direta, o tipo do menu principal do sistema (menu linear ou circular) e a possibilidade de aumentar ou diminuir o tamanho dos elementos da interface, de maneira indireta, de acordo com informações do perfil do usuário.

Para acessar o sistema Vilanarede e testar as opções de ajuste os usuários utilizaram uma estação que era constituída de um notebook, uma câmera digital e uma webcam, ambas utilizadas para capturar falas, reações e sentimentos dos usuários enquanto utilizavam o sistema e testavam as opções de ajuste, como é possível ver na Figura 16. Os vídeos poderiam ajudar a identificar dificuldades encontradas pelos usuários ao realizarem as operações de ajustes, dificuldades essas que poderiam sugerir melhorias na forma de exibição das opções de ajuste que foram disponibilizadas no sistema Vilanarede.

Figura 16: Estação utilizada para testar o framework FAN na oficina dez.

Os *feedbacks* dos usuários, sobre as opções de ajustes que foram fornecidas no sistema e sobre formas como os ajustes foram feitos, foram registrados em formulários, como o que é mostrado parcialmente na Figura 17.

Aspectos	Comportamento observado
1. O usuário percebeu que a forma de apresentação do menu estava ajustada para circular?	(X) Sim ( ) Não
2. Que passos o usuário executou na tentativa de mudar o menu para o formato de uma lista?	- NÃO LOCALIZOU O BOTÃO "AJUSTAR" SOZINHA, APENAS COM AJUDA DO FACILITADOR. - LOCALIZOU O BOTÃO "REMOVER AJUSTES" SEM DIFICULDADE
2.1 O usuário completou a tarefa de mudar o menu para o formato de uma lista com sucesso?	(X) Sim ( ) Não

Figura 17: Parte de um dos formulários preenchidos durante a oficina dez.

Nas seções seguintes são apresentadas mais informações sobre as atividades de teste do framework FAN, como detalhes sobre software e hardware utilizados na máquina cliente (notebook) e nos servidores (sistema Vilanarede e NBIC/ICE); também são apresentados mais detalhes sobre os cenários de ajuste de interface que foram testados na oficina dez e dois cenários que não foram testados na oficina, porém foram parcialmente inseridos no sistema.

4.1.4 Descrição dos servidores

A fim de possibilitar os testes do framework FAN na oficina dez e fornecer opções de ajustes na interface do sistema Vilanarede, o código do framework, desenvolvido com Ajax e PHP, foi incorporado ao código do sistema no servidor de produção do sistema Vilanarede, desenvolvido em PHP em cima da plataforma Drupal. O servidor de produção, executando o sistema Vilanarede e o framework FAN, tinha a seguinte configuração:

- Processador: 2 processadores Intel Xeon E5310 quad core - 1.6 Ghz;
- Memória: 8GB DDR2 800Mhz;

- Armazenamento: 4 HDDs SATA de 250GB cada;

A ferramenta NBIC/ICE, necessária para o funcionamento adequado do framework, foi instalada em outro servidor, a fim de evitar sobrecarga do servidor de produção do sistema Vilanarede, porém a configuração deste servidor era bem mais modesta:

- Processador: Intel Pentium 4 - 2.26 GHz;
- Memória: 2GB DDR 800MHz;
- Armazenamento: 4 HDDs SATA de 250GB cada;

A quantidade de memória adequada no servidor do sistema Vilanarede evitou que o framework e outros serviços instalados no Vilanarede sofressem problemas com falta de espaço na memória. O mesmo ocorreu no servidor da ferramenta NBIC/ICE, uma vez que 2GB foram suficientes para suportar a ferramenta, executada pelo servidor de aplicações JBoss, incluindo o Jess, a máquina de inferência utilizada pela ferramenta NBIC/ ICE para realizar inferências sobre normas no processo de geração de planos de ação.

O armazenamento não foi um fato importante nas atividades de teste do framework, pois os testes foram conduzidos com um número reduzido de usuários e pelo fato da capacidade de armazenamento do servidor de produção do sistema Vilanarede ser suficiente, atualmente, para o tamanho de sua base de usuários. Porém, é importante notar que o framework pode necessitar de uma boa capacidade de armazenamento caso ele seja aplicado em sistemas com bases de usuários muito grandes e o número de opções de ajustes de interface e páginas do sistema sejam muito altos, pois os arquivos XML com fatos gravados, associados a diferentes usuários e páginas, podem ocupar bastante espaço. Em relação ao servidor do sistema NBIC/ICE, sua capacidade de armazenamento também não interferiu no nas atividades de teste do framework FAN, pois foram gerados poucos arquivos de contexto durante a atividade e tais arquivos tinham um tamanho reduzido, na ordem de dezenas de kilobytes. Entretanto, se o sistema NBIC/ICE for utilizado em sistemas com grande número de usuários, opções de ajustes e páginas, o NBIC/ICE pode ter problemas com armazenamento assim como o framework FAN.

4.1.5 Descrição dos clientes

A plataforma usada pelos usuários para testar o framework FAN durante a oficina dez consistia, como citado anteriormente, de um notebook conectado à internet, com suporte de uma webcam e uma câmera digital. Tudo o que foi feito pelo usuário, na interface, foi capturado pelo software Camtasia Studio[14], que é uma ferramenta que grava atividades realizadas na tela do computador. O browser utilizado no notebook durante as atividades da oficina foi o Firefox 3.5. Foi utilizada a versão mais recente desse browser, até o momento da realização da oficina, pois versões mais antigas poderiam ter problemas de suporte à tecnologia JSON.

A configuração de hardware do notebook era suficiente para executar os códigos JavaScript associados às operações de ajuste da interface que foram implementadas e testadas durante a oficina dez, porém a conexão à internet, banda larga com link de aproximadamente 2Mbps, utilizada durante os testes apresentou grande instabilidade praticamente durante toda a atividade de teste do framework com os usuários.

4.2 Cenários de avaliação

Nesta seção são apresentados, em detalhes, cada um dos dois cenários de ajuste da interface do sistema Vilanarede que foram efetivamente testados com usuários na oficina dez (cenários 1 e 2) e são apresentados também outros dois cenários que não foram testados com usuários na oficina dez, porém os fatos e normas relacionados a eles foram inseridos no sistema Vilanarede e no sistema de gerenciamento de normas NBIC/ICE e foram feitos testes locais para avaliar a correta ligação entre tais fatos e normas.

4.2.1 Cenário 1: tamanho da fonte

O primeiro ajuste testado com usuários na oficina dez foi o aumento automático do tamanho da fonte do sistema. Na primeira atividade da oficina, os usuários foram solicitados a preencherem informações em seus perfis de usuário no sistema Vilanarede. No perfil havia um item que perguntava a frequência com a qual o usuário acionava o

[14] http://www.techsmith.com/camtasia.asp

mecanismo de aumento de fonte, exibido na Figura 18, e o usuário tinha três opções para escolher: “nunca”, “às vezes” e “sempre”, como exibido na figura 19.

Figura 18: Mecanismo para alterar tamanho da fonte no sistema Vilanarede.

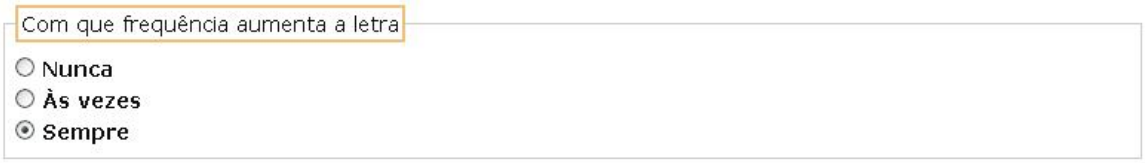


Figura 19: Campo do perfil de usuário sobre a frequência de ajustes da fonte.

Assim que o usuário clicava no botão para confirmar a edição do perfil, após terminar de preenchê-lo, o sistema verificava a opção selecionada pelo usuário ao preencher o item mostrado na Figura 19 e, caso o usuário informasse que sempre aumentava o tamanho da fonte, um fato era criado, utilizando o FAN, informando que o usuário em questão sempre aumenta a fonte. Tal fato era composto por três predicados com os *affordances* “usuário”, “usarsistema” e “fontegrande”, que pode ser interpretado, em linguagem natural, como “usuário usando o sistema deseja fonte grande”. Uma vez que o fato era criado e enviado ao sistema NBIC/ICE, o framework recebia um plano de ação informando que uma função, já existente no código do sistema Vilanarede, deveria ser executada para aumentar a fonte do sistema e, portanto, o framework realizava a chamada a tal função, resultando no aumento da fonte como pode ser visto na Figura 20. Essa função alterava o tamanho da fonte base do sistema, o que influenciava no tamanho de elementos diversos, não só textos, com tamanhos associados ao tamanho da fonte base.

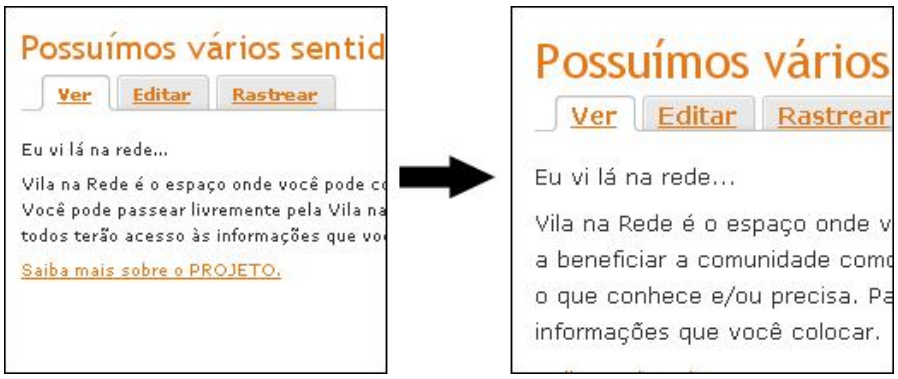


Figura 20: Framework aumentando o tamanho da fonte base do sistema Vilanarede.

Com isso, o framework passava a alterar, em todas as páginas do sistema, o tamanho da fonte, de "normal" para "grande", para cada usuário que informasse no perfil que ele tinha o hábito de aumentar a fonte do sistema sempre que o acessava.

4.2.2 Cenário 2: menu circular

A segunda opção de ajuste, provida pelo framework e testada na oficina dez com usuários do sistema Vilanarede, permitia aos usuários modificar, de forma direta, o tipo do menu principal do sistema, podendo escolher um menu linear ou circular. A atividade de teste dessa opção de ajuste foi realizada da seguinte maneira: um usuário sentava-se em frente a um notebook com uma página do sistema Vilanarede, utilizando menu circular (no lugar do menu padrão, linear), já carregada no browser, e o usuário era requisitado a observar a página e verificar se ele havia notado algum elemento diferente ou novo na página. Os usuários levaram, no geral, alguns poucos segundos para notar que havia um elemento novo na interface: o menu circular.

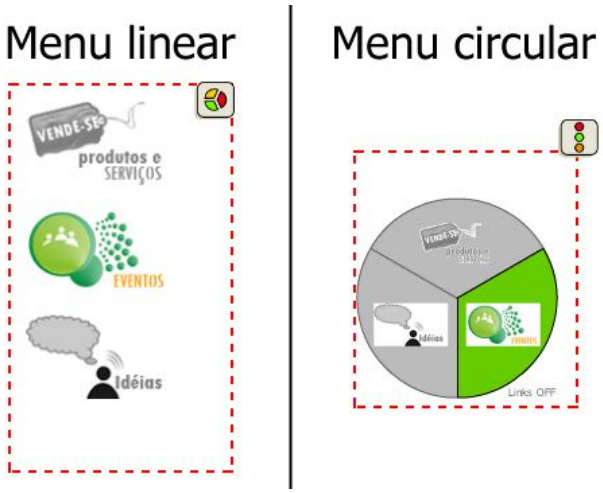


Figura 21: Diferentes formas de apresentação do menu: uma forma linear e uma forma circular.

Na etapa seguinte, o usuário era informado que, a partir daquele momento em diante, ele teria a opção de alternar o tipo do menu entre um menu linear, padrão do site, e um menu circular, como pode ser visto na Figura 21; então era solicitado ao usuário que ele tentasse mudar o menu circular, carregado na tela, para o menu linear. Tal pedido foi feito para avaliar a dificuldade que o usuário teria para descobrir os passos que deveriam ser executados para mudar o tipo do menu, o que poderia motivar melhorias nesses passos. Os usuários foram bem sucedidos nessa etapa, mas sugeriram algumas mudanças nos passos que deveriam ser executados para mudar o tipo do menu, em especial fizeram sugestões sobre a aparência do botão que deveria ser pressionado para ativar opções de ajuste na tela, que permitiriam o ajuste no menu principal.

Figura 22: Primeiro passo para alterar o tipo de menu.

A fim de mudar o tipo do menu, primeiro o usuário deveria clicar no botão "Ajustar", exibido na Figura 22, para habilitar opções de ajuste na tela. Em seguida o usuário deveria clicar no botão, com uma imagem de um menu linear, localizado próximo a parte superior direita do menu circular, como é mostrado na Figura 23.

Figura 23: Segundo passo do processo de mudança do tipo de menu.

Ao clicar no botão para mudar o tipo de menu, como mostrado na Figura 23, um fato era criado, utilizando o framework FAN, contendo a informação de que um dado usuário tinha preferência por menu linear. Havia uma norma no sistema que dizia que "se um usuário prefere menu linear, então o sistema deve exibir menu linear". Logo, a ferramenta ICE inferia, a partir do fato gerado e a partir das normas do sistema, que o sistema deveria exibir um menu linear para o usuário. O plano de ação então requisitava o framework a chamar uma função, já existente no código do sistema, para apagar o menu circular e exibir um menu linear. Logo tal função era chamada pelo framework e o resultado de tal ação é mostrado na Figura 24.

Figura 24: Resultado da mudança do menu circular para o menu linear.

Como o menu linear era o menu padrão do sistema, além de requisitar a mudança do menu circular para um menu linear, o framework apagava, do arquivo de fatos globais do usuário, uma vez que o tipo de menu selecionado vale para todas as páginas do sistema, o fato relacionado à exibição de menu circular. Dessa forma o menu linear seria carregado em todas as páginas.

4.2.3 Cenário 3: setas de navegação

Este cenário, que foi desenvolvido apenas para testes locais, estava ligado à opção de permitir que usuários avançados, isto é, com experiência em navegação Web, pudessem esconder as setas de navegação do sistema Vilanarede, uma vez que alguns usuários do sistema, insatisfeitos com a presença das setas de navegação na tela, sugeriram, durante uma das oficinas realizadas, a opção de esconder ou remover as setas. Logo, foi criada no sistema NBIC uma norma que define "se um usuário está no sistema e ele é um usuário avançado, o sistema deve permitir que o usuário esconda as setas de navegação". Em seguida, foi desenvolvido o fato, que seria usado para informar à ferramenta NBIC/ICE que um usuário era avançado, composto por três predicados com os seguintes *affordances*: "usuário", "usarsistema" e "escondersetas". Esse fato pode ser interpretado, em

linguagem natural, como algo próximo a “usuário do sistema deseja esconder setas”. Tal opção de ajuste é ilustrada na Figura 25.

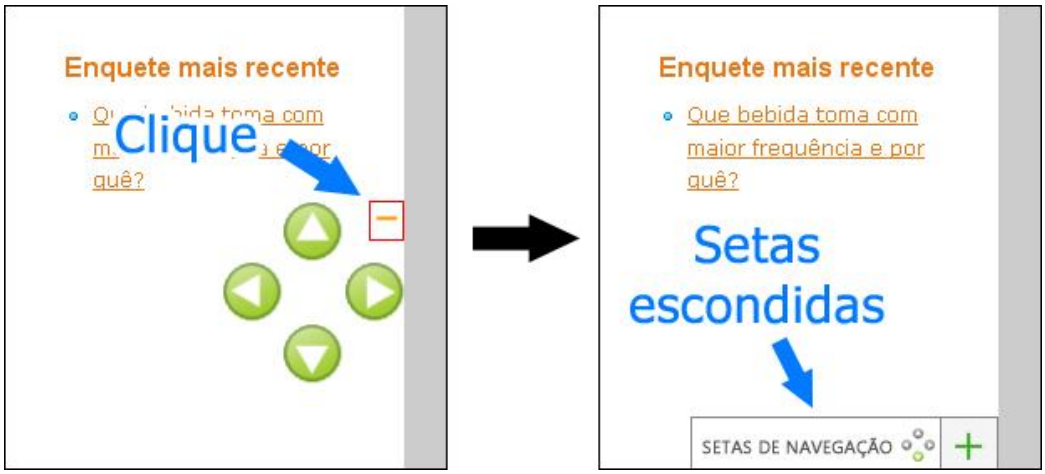


Figura 25: Etapas para esconder as setas de navegação.

A fim de saber quais usuários eram avançados, foi criado um campo no perfil de usuário no sistema Vilanarede para colaborar nessa tarefa, como pode ser visto na Figura 26.

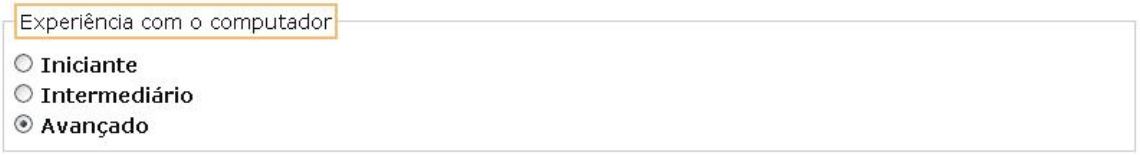


Figura 26: Campo de perfil criado para ajudar o sistema a identificar usuários avançados.

Alguns testes foram executados para verificar se o ICE geraria um plano de ação adequado diante do fato e norma criados, relacionados à opção de esconder as setas, e os testes foram bem sucedidos. A única etapa que faltou para essa opção de ajuste ficar totalmente funcional no sistema de produção do Vilanarede foi o desenvolvimento, no sistema Vilanarede, da função JavaScript para esconder as setas de navegação.

4.2.4 Cenário 4: contraste

Neste cenário, o sistema poderia mudar automaticamente o contraste das páginas caso o usuário tivesse alguma dificuldade visual ou caso o sistema identificasse que o usuário sempre mudava o contraste das páginas ao acessar o sistema. Uma forma de o sistema identificar tal comportamento do usuário seria pelo perfil do usuário, que possuía um

campo que perguntava a frequência com a qual o usuário mudava o contraste das páginas, como pode ser visto na Figura 27. Outra forma seria avaliar com que freqüência o usuário requisitava a mudança de contraste.

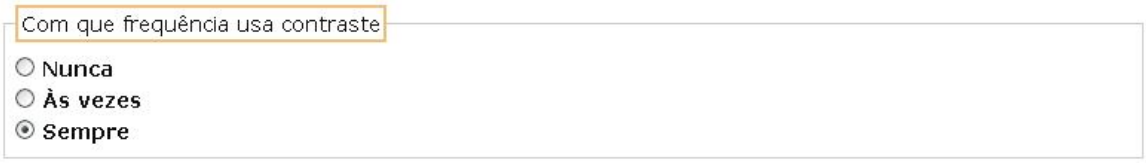


Figura 27: Campo de perfil criado para ajudar o sistema a identificar usuários que sempre modificam o contraste das páginas.

É possível notar que este cenário se assemelha muito ao cenário 1, relacionado ao aumento da fonte das páginas, sendo ambos os ajustes feitos de maneira automática pelo sistema após análise de informações sobre o usuário. Um exemplo da interface do sistema Vilanarede com contraste alterado pode ser visto na Figura 28. A norma associada ao ajuste do contraste dizia algo, em linguagem natural, como: "se um usuário acessa o sistema e o usuário tem dificuldade de visualização ligada ao contraste, então o sistema deve exibir contraste diferenciado". O fato, por sua vez, que estava ligado a essa norma, era composto por três predicados, como o fato associado ao ajuste de fonte, com os seguintes *affordances*: "usuário", "usarsistema" e "contraste".

Figura 28: Exemplo da interface do Sistema Vilanarede com contraste alterado.

Assim como no cenário relacionado às setas de navegação, este cenário não foi completamente desenvolvido, pois faltava o desenvolvimento, no sistema de produção do Vilanarede, de uma função JavaScript para alterar o contraste das páginas. A norma e fato criados funcionaram de maneira correta nos testes realizados com o framework FAN e o sistema NBIC/ICE, isto é, o plano de ação gerado pelo ICE, ao realizar inferências utilizando o fato e a norma em questão, estava de acordo com o esperado e foi corretamente interpretado pelo FAN.

4.3 Resultados obtidos

Os dados coletados durante as atividades de teste do framework FAN, realizadas com sete usuários do sistema Vilanarede, mostraram que o framework funcionou adequadamente durante os testes e apontaram alguns pontos positivos e negativos do FAN e dos mecanismos de ajustes desenvolvidos no sistema Vilanarede. Observações feitas durante os testes mostraram que a maioria dos sete usuários gostou dos ajustes oferecidos e da forma como eles foram oferecidos; os usuários gostaram em especial do ajuste automático (seção 4.2.1, cenário 1) do tamanho da fonte, o que pôde ser demonstrado por frases ditas pelos usuários, como “Achei muito bom os ajustes, muito práticos” e “É muito boa a possibilidade de mudar o tipo do menu” .

Analisando os ajustes feitos pelos sete usuários no primeiro cenário, relacionado ao tipo do menu principal do sistema, constatou-se que a maior parte dos usuários levou alguns minutos para identificar todos os passos que deveriam ser executados para mudar o tipo do menu. Todos os usuários conseguiram realizar a tarefa de mudar o menu principal, porém alguns precisaram de um pequeno apoio de pesquisadores que observavam os usuários executando suas tarefas. Além disso, constatou-se que seis usuários, de um total de sete, preferiram utilizar menu circular no lugar do menu padrão, linear.

No cenário relacionado ao aumento de fonte, foi observado que quatro usuários notaram a modificação automática do tamanho da fonte e um usuário não notou a modificação. Dois usuários não puderam ser avaliados nesse cenário, pois devido a

problemas de instabilidade na conexão da máquina cliente (notebook) à internet, algumas funções JavaScript do framework não funcionaram adequadamente por sofrerem problemas de falta de resposta (*timeout*) do servidor do NBIC/ICE. Como o aumento no tamanho da fonte não era substancial, de 13px para 14px, acredita-se que esse tenha sido o motivo de um usuário não ter notado a modificação no tamanho da fonte e outros elementos da interface (pois eles possuem tamanhos associados ao tamanho da fonte).

Durante e no fim das atividades de teste, alguns usuários deram sugestões para melhorias a serem feitas na forma como as opções de ajuste foram oferecidas. A maioria absoluta dos sete usuários sugeriu que o botão "Ajustar", responsável por exibir as opções de ajuste, deveria ter uma aparência diferente, com maior tamanho, destaque e contendo uma imagem apropriada para um botão, no lugar de ter a aparência de um link textual como ocorreu durante os testes.

Uma análise posterior, feita nos arquivos de fatos gravados, mostrou que 21, de aproximadamente 270 usuários do sistema Vilanarede já realizaram algum tipo de ajuste na interface, entre as opções de modificar o tipo do menu principal ou modificar o tamanho da fonte, por meio do framework.

4.4 Discussão

Apesar das vantagens de se utilizar o framework, confirmadas nos dados coletados nas atividades de testes do FAN, antes e durante as atividades foi possível identificar alguns pontos negativos do framework e alguns usuários apontaram aspectos do processo de ajuste que poderiam ser melhorados, como a aparência e a posição do botão "Ajustar". De forma geral, eles gostariam de ver um botão maior, com mais destaque, em uma posição que facilitaria sua identificação e que fosse mais fácil de ser clicado.

Durante a atividade foi possível observar que um dos pontos negativos do framework está relacionado à sua performance. O framework pode levar alguns segundos para ajustar a interface de um sistema se o sistema tiver muitas normas complexas ou se o servidor NBIC/ICE não for uma máquina tão rápida. O framework pode também ter problemas com performance especialmente se o usuário (cliente) tiver uma conexão à

internet muito fraca ou problemática, uma vez eu o framework FAN pode sofrer com falta de resposta (*timeout*) do servidor NBIC/ICE se pacotes forem perdidos, corrompidos ou levarem muito tempo em seus trajetos entre o servidor NBIC/ICE e a máquina do cliente.

Outro ponto negativo do FAN é que não é fácil realizar a sincronização das requisições feitas pelo módulo de percepção, do framework, ao NBIC/ICE, via o Cliente SOAP, de forma que o NBIC/ICE retorne, para o framework, os dados corretos e esperados. Além disso, o framework permite a desenvolvedores a flexibilização de praticamente todos os elementos da interface, usando normas simples, porém o desenvolvimento de código JavaScript para implementar o comportamento ajustável de certos elementos pode não ser muito simples.

Capítulo 5

Conclusão

Diferentes usuários possuem diferentes necessidades e interesses e é essencial que interfaces se adequem às diferentes necessidades. Com o objetivo de lidar com esta diversidade de usuários e contextos de uso, interfaces devem ser flexíveis.

Esta pesquisa investigou o desenvolvimento de interfaces flexíveis e propôs um framework para facilitar o design e desenvolvimento de tais interfaces. Esse framework, chamado FAN (Flexibilidade por meio de Ajax e Normas), foi fundamentado no conceito de normas e foi desenvolvido com Ajax e PHP com intuito de prover ajustes de interface, baseados em normas, que possam ser feitos e mantidos sem haver a necessidade de re-carregamento de página. O framework proposto funciona em conjunto com o sistema NBIC/ICE, de gerenciamento de normas, para conseguir prover opções de ajustes baseadas em normas.

5.1 Considerações finais

O framework FAN foi validado com sucesso no sistema Vilanarede, um sistema de rede social desenvolvido no contexto do projeto e-Cidadania, durante uma oficina participativa organizada pelo projeto e-Cidadania com a participação de usuários do sistema Vilanarede. As atividades realizadas na validação do framework ajudaram na identificação de vantagens de se utilizar o framework para prover interfaces ajustáveis, assim como ajudaram na identificação de seus pontos fracos e os usuários, que participaram dos testes, deram *feedbacks* importantes que contribuíram para melhorias no FAN. Os resultados dos testes atestaram o correto funcionamento do framework e

mostraram que os usuários gostaram das opções de ajuste fornecidas no sistema Vilanarede com o uso do FAN. Entretanto os usuários sugeriram melhorias na forma como as opções de ajuste foram exibidas na interface e sugeriram também outras opções de ajuste que eles gostariam de ver na interface do sistema.

O framework mostrou-se potencialmente promissor sendo capaz de criar, para designers e desenvolvedores, meios para o desenvolvimento de interfaces flexíveis. Porém, alguns pontos importantes do framework FAN ainda devem ser melhorados, como a questão de seu desempenho e a questão da dificuldade com a sincronização de chamadas feitas pelo FAN ao sistema NBIC/ICE.

5.2 Contribuições da pesquisa

A fim de encontrar uma solução para o problema de como desenvolver interfaces Web flexíveis, este trabalho propôs uma abordagem fundamentada no conceito de normas, que resultou no desenvolvimento de um framework. Esse framework foi validado de maneira satisfatória com usuários finais e pode-se dizer que esse framework constitui uma solução possível para o desafio de se desenvolver interfaces Web ajustáveis. A síntese dos resultados deste trabalho também está em Fortuna et al. (2010).

A principal contribuição deste trabalho, portanto, está na proposta e desenvolvimento do framework FAN, incluindo a sua concepção e design, que provê, a designers e desenvolvedores, meios para a realização de ajustes de interfaces Web, em tempo real, com ajustes fundamentados em normas. Além disso, o framework permite que as opções de ajuste de interface sejam diferentes para cada página do sistema e para cada usuário.

Essa abordagem se mostrou eficaz após a validação do framework FAN com usuários finais, mas certamente essa abordagem pode ser expandida em trabalhos futuros, que são discutidos na seção seguinte.

5.3 Trabalhos futuros

Como trabalho futuro, um modelador de normas, usando tecnologias Web, está sendo desenvolvido com o objetivo de tornar mais fácil e acessível para desenvolvedores a inclusão de novas normas no sistema NBIC ou a modificação de normas já existentes.

A fim de melhorar o desempenho do processo de ajuste de interface de acordo com as necessidades dos usuários é necessária a otimização de sua implementação incluindo os aspectos ligados a protocolos de comunicação e armazenagem de dados. Entretanto, um servidor mais rápido, poderoso, pode ser facilmente utilizado, em conjunto com uma conexão banda larga mais rápida, para rodar a ferramenta NBIC/ICE, uma vez que ela estava rodando em uma máquina antiga, um servidor Pentium 4, com uma conexão banda larga limitada.

Uma proposta promissora considerada como trabalho futuro é a integração do framework FAN com tecnologias da Web semântica. Ao invés de armazenar normas e fatos dentro de arquivos ou em banco de dados no sistema NBIC/ICE, podem ser utilizadas tecnologias como RDFS ou SWRL (*Semantic Web Rule Language*) para armazenar normas e fatos na Web. Consultas sobre normas, atualmente feitas pelo ICE ao NBIC, podem ser feitas usando SparQL, uma linguagem de consulta para RDF. No lugar de utilizar o sistema NBIC/ICE para realizar inferências sobre normas e fatos, podem ser utilizadas outras tecnologias como RDF/RDFS, OWL e SWRL na realização de inferências. Dessa forma, usuários podem armazenar, em arquivos RDF hospedados na Web, informações sobre seus interesses ou necessidades (que no framework FAN estão associados a fatos) e o framework pode utilizar os fatos guardados em arquivos RDF para realizar inferências, utilizando as tecnologias supracitadas, e ajustar a interface de um sistema de acordo com os interesses ou necessidades do usuário. Com isto será possível realizar ajustes utilizando somente tecnologias Web e arquivos hospedados de maneira distribuída na grande rede.

Referências

Baranauskas, M.C.C.: *e-Cidadania: Sistemas e Métodos na Constituição de uma Cultura mediada por Tecnologias de Informação e Comunicação.* Proposta para o Edital Instituto Microsoft Research – FAPESP de Pesquisas em TI, 2007.

Bonacin, R. ; Baranauskas, M. C. C. ; Liu, K. ; Sun, L. . *Norms-Based Simulation for Personalized Service Provision*. Semiotica (Berlin), v. 2009, p. 403-428, 2009.

Casati, M. (2006) *JavaScript SOAP Client*, CodeProject, http://www.codeproject.com/KB/ajax/JavaScriptSOAPClient.aspx, last access January, 25, 2010.

Dourish, P. (1996) *Open implementation and flexibility in CSCW toolkits.* Ph.D. Thesis, University College London, ftp://cs.ucl.ac.uk/darpa/jpd/dourish-thesis.ps.gz, last access October 17, 2007.

Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., and Mehandjiev, N. (2004) Meta-Design: a Manifesto for End-User Development. *Communications of the ACM*, vol. 47, Issue 9.

Fortuna, F., Bonacin, R., Baranauskas, M. C. C. (2010) A Framework for Flexibility at the Interface: joining Ajax technology and Semiotics; *12th International Conference on Enterprise Information Systems (ICEIS)*, Funchal, June, 2010.

Garrett, J.J. *Ajax: A New Approach to Web Applications*, Adaptive Path, February 18, 2005. http://www.adaptivepath.com/ideas/essays/archives/000385.php, last access December, 21, 2009.

Institute of Electrical and Electronics Engineers. (1990) *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY: 1990

Kahler, H, Morch, A., Stiemerling, O. and Wulf, V. (2000) Special Issue on Tailorable Systems and Cooperative Work, *Computer Supported Cooperative Work*, 1, 9, Kluwer Academic Publishers.

Liu K. (2000) *Semiotics in information systems engineering*, Cambridge University Press.

Moran, T. and Dourish, P. (2001) Introduction to the special issue on Context-Aware Computing. *Human-Computer Interaction*, 16 (2), 2-3, Lawrence Erlbaum Associates.

Morch A. and Mehandjiev N. (2000) Tailoring as Collaboration: the Mediating Role of Multiple Representations and Application Units. Kahler, H, Morch, A., Stiemerling, O., Wulf, V. (Eds), Special Issue on Tailorable Systems and Cooperative Work, *Computer Supported Cooperative Work*, v. 9, Issue 1, Kluwer.

Newcomer, E. and Lomow, G. (2004) *Understanding SOA with Web Services*. Addison-Wesley, Upper Saddle River, NJ.

Savidis, A. and Stephanidis C. (2004) "Unified user interface development: the software engineering of universally accessible interactions". *Universal Access in the Information Society*, 3, 165-193.

Stamper, R. K. (1973) *Information in Business and Administrative Systems*, NY, USA: John Wiley & Sons, Inc.

Stamper, R., 1993. Social Norms in Requirements Analysis – an outline of MEASUR. In Jirotka, M., Goguen, J. and Bickerton, M. (eds.), *Requirements Engineering, Technical and Social Aspects*. Academic Press. New York.

Stamper, R., Liu, K., Hafkamp, M. and Ades, Y., (2000) Understanding the roles of signs and norms in organizations – a semiotic approach to information system design, *Behaviour & Information Technology*, 19, 1, 15-27, Taylor & Francis.

Teege, G., Kahler, H. and Stiemerling, O. (1999) Implementing Tailorability in Groupware, In *SIGGROUP Bulletin*, 20, 2, 57-59, Association for Computing Machinery.

W3C, XMLHttpRequest, *W3C Working Draft*, November, 19, 2009, http://www.w3.org/TR/XMLHttpRequest/, last access December, 21, 2009.

W3C, Document Object Model (DOM) Level 1 Specification, *W3C Recommendation*, October, 01, 1998, http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/, last access January, 21, 2010.

W3C, Document Object Model (DOM) Level 3 Core Specification, *W3C Recommendation*, April, 07, 2004, http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/, last access January, 21, 2010.

W3C, Simple Object Access Protocol (SOAP) 1.1, *W3C Note*, May, 08, 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, last access January, 23, 2010.

Anexo A: Formulários de Observação

A.1 Formulário de observação – Oficina 7

A.2 Formulário de observação – Oficina 9

A.3 Formulário de observação – Oficina 10

A.1 Formulário de observação – Oficina 7

Oficina 7 – 13/04

Formulário do observador

Observador: _____

Grupo: _____

_____

_____

1. Observe os usuários usando o sistema e anote quaisquer comentários (sugestões ou críticas) que os usuários fizerem sobre elementos de interface (botões, imagens, fonte, vídeo, cabeçalho, formulários etc.). Se um usuário comentar algo sobre determinado elemento de interface, pergunte a ele **de que forma** ele gostaria de alterar tal elemento para ficar com a cara que o usuário deseja.

Elemento de interface *Ex_1: setas de navegação* *Ex_2: imagens*	**Sugestão de alteração de atributo ou comportamento** *Ex_1: Usuário gostaria de ter a opção de esconder o elemento.* *Ex_2: Aumentar ou diminuir o tamanho.*

2. No momento de compor a descrição, os participantes se preocuparam com:
(Marque com X no que se aplica).

Níveis de dificuldade

- ☐ Preocupação com vocabulário (ex.: erros gramaticais, organização da frase...)
- ☐ Preocupação com os detalhes/referências (ex.: especificar que determinado ícone serve para fechar a janela)
- ☐ Preocupação com a quantidade de informação (ex.: muito texto, muitos passos)

Formato

- ☐ Usaram desenho
- ☐ Usaram metáfora
- ☐ Fizeram passo a passo
- ☐ Escreveram texto corrido

Fluxos

- ☐ Consideraram várias alternativas de fluxo (ex.: formas diferentes de fazer a mesma coisa)

Comportamento

- ☐ Discutiram sobre como se comportar em frente à mídia (ex.: como se posicionar em frente à câmera, foco...)

3. Outras observações.

A.2 Formulário de observação – Oficina 9

9 ª Oficina do Projeto e-Cidadania
Formulário de Observação - Atividade Ordenação e Tailoring

Dados básicos

Integrantes do grupo:

_____

Observador:

_____

Tarefa: ordenação

Os participantes compreenderam e executaram corretamente a atividade (colocar cartões em ordem de preferência para visualização)? Quais as **dificuldades encontradas**?

Quais os **critérios discutidos** no grupo? Listar critérios e participante que sugeriu.

Produtos e serviços	Eventos	Idéias

Qual o **critério escolhido** pelo grupo em cada uma das categorias? A decisão foi unânime?

Produtos e serviços	Eventos	Idéias
Critério: Unanimidade: ☐ Sim ☐ Não	Critério: Unanimidade: ☐ Sim ☐ Não	Critério: Unanimidade: ☐ Sim ☐ Não

Quando houve empate (dois cartões com o mesmo dado no critério escolhido), qual o critério escolhido para desempate?

Tarefa: montagem da interface - tailoring

Ficou claro para os participantes que a interface a ser criada deveria **refletir as preferências do grupo** e não necessariamente reproduzir o que há hoje no Vila?

Quais os **elementos de interface discutidos** no grupo? Listar elemento e participante que sugeriu. Focar naqueles que foram discutidos, mas não inseridos na solução final.

A.3 Formulário de observação – Oficina 10

1. FORMULÁRIO DE OBSERVAÇÃO

Observador: _____

Participante: _____

Aspectos	Comportamento observado
1. O usuário percebeu que a forma de apresentação do menu estava ajustada para circular?	( ) Sim ( ) Não
2. Que passos o usuário executou na tentativa de mudar o menu para o formato de uma lista?	
2.1 O usuário completou a tarefa de mudar o menu para o formato de uma lista com sucesso?	( ) Sim ( ) Não
2.2 Quais as impressões do participante a respeito da possibilidade e forma como está sendo oferecido o ajuste no sistema Vila na Rede?	

Aspectos	Comportamento observado
3. O usuário percebeu o ajuste automático do tamanho dos elementos na interface (quando houver)?	( ) Sim ( ) Não ( ) Não houve ajuste
4. Quais as impressões do participante com relação ao ajuste automático (quando houver)?	
5. O participante mencionou / sinalizou outros tipos de ajustes que gostaria de ver no sistema? Se sim, quais?	