SPA^3: Um *framework* para privacidade

Cereda, P. R. M. e Zorzo, S. D.

***Resumo*—Este artigo apresenta um *framework* para privacidade, chamado *SPA*^{3}, que contribui para a construção simplificada de sistemas, com reforços nos aspectos de privacidade. O *SPA*^{3} utiliza comandos de privacidade, que são reutilizáveis, para a especificação de situações de violação de privacidade. O *framework* apresentado é uma solução viável para incorporar reforços de privacidade nos sistemas.**

***Palavras-chave*—Controle de acesso, Privacidade, Autômato Adaptativo.**

I. INTRODUÇÃO

A privacidade está seriamente comprometida na sociedade de informação global, pois o tráfego gerado em uma rede global transcende os limites territoriais, e não é gerenciado de modo centralizado. Com isso, surgem os riscos de invasão de privacidade, uma vez que dados dos usuários podem ser interceptados ou monitorados. Assim, buscou-se uma padronização de ações consideradas intrusivas que são distintas a uma determinada região ou país para formulação de leis e estabelecendo penas às situações de violação.

Fernandes [1] afirma que "para ter privacidade, uma pessoa precisa ter controle sobre as informações existentes sobre si mesma e exercer este controle de forma consistente com seus interesses e valores pessoais". Assim, durante a interação de um usuário em um determinado sistema, suas informações devem ser divulgadas apenas com o consentimento do mesmo.

Mecanismos de segurança são utilizados para representar, de modo objetivo e simplificado, os requisitos de segurança de um determinado sistema. Entretanto, ao mesmo tempo que exercem um controle sobre os dados de usuários que trafegam no sistema, podem invadir a privacidade de tais usuários, conforme uma pesquisa feita por [2] [3].

Este artigo apresenta um *framework* para privacidade chamado *SPA*^{3} (Sistema de Privacidade Auditável com Autômato Adaptativo), que objetiva simplificar a especificação dos aspectos de privacidade de um determinado sistema. Os comandos de privacidade fornecidos pelo *framework* verificam se uma determinada situação caracteriza invasão de privacidade, de acordo com um conjunto de leis específicas.

Na seção II é apresentada a motivação para o desenvolvimento do *framework* a partir do Sistema de Privacidade Auditável e do Modelo de Controle de Acesso Adaptativo. Na seção III, o *framework* é apresentado, com enfoque em suas características. As considerações finais são apresentadas na seção IV.

Paulo Roberto Massa Cereda e Sérgio Donizetti Zorzo.

Este trabalho foi financiado pela Capes.

Os autores são do Departamento de Computação, Universidade Federal de São Carlos, Rod. Washington Luís, Km 235, Caixa Postal 676, 13565-905, (e-mails: paulo_cereda@dc.ufscar.br e zorzo@dc.ufscar.br).

II. MOTIVAÇÃO

Nesta seção, os tópicos que motivaram a criação do *framework* são apresentados, como Sistema de Privacidade Auditável, que gerencia informações entre domínios, e o Modelo de Controle de Acesso Adaptativo, que utiliza um autômato adaptativo para prover o controle de acesso.

A. Sistema de Privacidade Auditável

Um *Sistema de Privacidade Auditável* [4] é definido como um sistema que permite o gerenciamento de objetos entre domínios e a realização de auditorias em tempo de execução e *post-hoc*.

O gerenciamento de objetos entre domínios é suportado pelos seguintes eventos: transferência, ação, criação, estabelecimento de direitos, notificação e *logging*. O evento de transferência consiste em transferir um objeto, copiando-o e passando a cópia ao domínio do receptor. Na ação, usa-se um objeto privado para um determinado propósito, enquanto que no evento de criação permite-se realizar a adição de novos objetos ou indivíduos ao sistema e à matriz de acesso. No estabelecimento de direitos, um indivíduo ou dono de um dado objeto pode atribuir ou revogar permissões. O evento de notificação informa o indivíduo acerca de ações realizadas por outros indivíduos no sistema, e no *logging* o sistema monitora todos os eventos – os que foram bem-sucedidos e os que falharam (é usado para verificação e auditoria).

Um Sistema de Privacidade Auditável necessita de um mecanismo de controle de acesso para atender seus requisitos. O modelo HRU [5] , proposto por Harrison, Ruzzo e Ullman, é um modelo de controle de acesso que foi utilizado no Sistema de Privacidade Auditável proposto por May, Gunter e Lee [4], através de seis operações primitivas presentes no modelo, com a adição de duas novas operações que realizam a notificação (**inform** *s* **of** *t*) e *logging* (**log** *t*). Alterações no modelo HRU foram necessárias para que este implementasse totalmente as características de um Sistema de Privacidade Auditável [4].

A utilização do modelo HRU estendido em um Sistema de Privacidade Auditável justifica-se por sua simplicidade de uso. Entretanto, outros modelos de controle de acesso poderiam ser utilizados [4], como o Modelo de Controle de Acesso Adaptativo, apresentado a seguir.

B. Modelo de Controle de Acesso Adaptativo

Para atender as necessidades do Modelo de Controle de

Acesso Adaptativo, definiu-se uma extensão do autômato adaptativo [6] [7] [8], chamado de *autômato adaptativo de controle de acesso*, apresentado a seguir.

O *Autômato Adaptativo de Controle de Acesso* é uma extensão do Autômato Adaptativo, que permite que um símbolo da cadeia de entrada seja passado como parâmetro a uma dada função adaptativa.

Um *autômato adaptativo de controle de acesso* M é definido por $M = (Q, S, \Sigma, \Gamma, P, q_0, Z_0, F)$, tal que Q é o conjunto finito de estados, $Q \subset Q^A$, Q^A é o conjunto de todos os estados possíveis, Q^A é enumerável, S é o conjunto de sub-máquinas, Σ é o alfabeto finito de entrada, $\Sigma \subset \Sigma^A$, Σ^A é o alfabeto enumerável de todos os símbolos possíveis, Γ é o alfabeto finito da pilha, $\Gamma \subset \Gamma^A$, $\Gamma = Q \cup \{Z_0\}$, Γ^A é o alfabeto enumerável de todos os símbolos possíveis da pilha, $\Gamma^A = Q^A \cup \{Z_0\}$, P é uma função $P: Q^A \times \Sigma^A \times (\Sigma^A \cup \{\epsilon\}) \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$, H^0 definido a seguir, $q_0 \in Q$ é o estado inicial, $Z_0 \in \Gamma$ é o símbolo inicial da pilha, $F \subset Q$ é o conjunto de estados finais.

H^0 é definido como $H^0 = \{f \mid f : E \rightarrow E\}$. Foi utilizado $g: X \rightarrow Y$ para uma relação g tal que $g \subseteq X \times Y$ e se $g(x, y)$ e $g(x, z)$, $x \in X$, $y, z \in Y$, então $y = z$. Como abreviação, usa-se $g(x) = y$ para $g(x, y)$.

Define-se $E = \{N : N$ é um autômato adaptativo de controle de acesso $N = (Q', \Sigma', \Gamma', P', q_0, Z_0, F)$, onde $Q' \subset Q^A$, $\Sigma' \subset \Sigma^A$, $\Gamma' \subset \Gamma^A$, $P': Q^A \times \Sigma^A \times (\Sigma^A \cup \{\epsilon\}) \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$.
Observe que q_0, Z_0 e F são os mesmos em qualquer $N \in E$.

O conjunto de todas as sub-máquinas do autômato adaptativo de controle de acesso é representado por S. Cada *sub-máquina* i é definida como $s_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$, onde $Q_i \subseteq Q$ é o conjunto de estados da sub-máquina i, $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada da sub-máquina i, $P_i \subseteq P$ é o mapeamento da sub-máquina i, $q_{i0} \in Q_i$ é o estado de entrada da sub-máquina i, e $F_i \subseteq Q_i$ é o conjunto de estados finais da sub-máquina i.

Uma transição pode ser da forma $(q, a, \beta) \vdash (q', a', \beta', A, B)$ para $P(q, a, \beta) = (q', a', \beta', A, B)$, ou $(q, ab, \beta) \vdash (q', a', \beta', A, B(b))$ para $P(q, ab, \beta) = (q', a', \beta', A, B(b))$, A e B são funções adaptativas, $A \in H^0$, $B \in H^0$. Se $\tilde{q}$, $\tilde{a}$, $\tilde{b}$ ou $\tilde{\beta}$ não pertencem à máquina atual, então $P(\tilde{q}, \tilde{a}, \tilde{\beta}) = (\tilde{q}, \tilde{a}, \tilde{\beta}, I, I)$ ou $P(\tilde{q}, \tilde{a}\tilde{b}, \tilde{\beta}) = (\tilde{q}, \tilde{a}\tilde{b}, \tilde{\beta}, I, I)$, onde I é a função identidade em E.

A linguagem aceita por um autômato adaptativo de controle de acesso M é dada por $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q_f, \epsilon, Z_0)$, onde $q_f \in F\}$.

A especificação do *Modelo de Controle de Acesso Adaptativo* é descrita por um autômato adaptativo de controle de acesso M. O alfabeto do autômato adaptativo de controle de acesso é representado por Σ, $\Sigma \subset \Sigma^A$, $\Sigma^A = \{w \mid w$ é um nome de indivíduo válido $\} \cup \{x \mid x$ é um nome de objeto válido $\} \cup \{\lambda_x \mid x$ é um nome de objeto válido $\} \cup \{y \mid y$ é um direito válido $\} \cup \{\lambda_{NS}, \lambda_{NO}, \lambda_{RS}, \lambda_{RO}, \lambda_{RAO}, G, T\}$. Assim, o modelo de controle de acesso contém um conjunto finito de indivíduos atuais $S = \{s_1 \ldots s_{n_1}\}$, $S \subset \Sigma$; um conjunto finito de objetos atuais $O = \{o_1 \ldots o_{n_2}\}$, $O \subset \Sigma$; um conjunto finito de direitos $R = \{r_1 \ldots r_{n_3}\}$, $R \subset \Sigma$; um conjunto λ de símbolos que representam as ações sobre os objetos, $\lambda = \{\lambda_{o_1} \ldots \lambda_{o_{n2}}\}$, $\lambda \subset \Sigma$; e um conjunto de símbolos que representam as alterações a serem realizadas na configuração do autômato, $\{\lambda_{NS}, \lambda_{NO}, \lambda_{RS}, \lambda_{RO}, \lambda_{RAO}, G, T\}$.

O Modelo de Controle de Acesso Adaptativo possui um conjunto C de *comandos de privacidade*, $C = \{c_1 \ldots c_i\}, i \in \mathbb{N}$. Esses comandos são codificações de regras-alvo obtidas a partir de uma legislação ou política de privacidade escrita em linguagem natural, e verificam se uma determinada situação pode ser caracterizada como violação de privacidade. A gramática da linguagem de um comando de privacidade é descrita a seguir.

```
Program ::= Valor Id "(" ParamList ")" "início"
            StatementList "fim"
Valor ::= "true" | "false"
ParamList ::= Id | Id "," ParamList
StatementList ::= | Statement StatementList
Statement ::= CondDecl | AtribDecl | ExecDecl |
              returnStat
CondDecl ::= "se" CondExp "então" "{"
             StatementList "\}" | "se" CondExp
             "então" "{" StatementList "}" "senão"
             "{" StatementList "}"
CondExp ::= CondCom | CondCom "e" CondExp
CondCom ::= "eval" "(" Arg "," Id ")" | "equiv"
            "(" Arg "," Id ")"
Arg ::= Id | "<" Id ">" | Id Arg | "<" Id ">" Arg
AtribDecl ::= Var "=" Var
Var ::= Id
ExecDecl ::= Var "=" "exec" "(" Arg "," Id ")"
```

```
returnStat ::= "retornar" Valor
```

Um comando de privacidade $c_i \in C$ é uma função que recebe como parâmetro um conjunto não-vazio de argumentos e retorna *"true"* se as condições presentes no corpo da função foram satisfeitas e se os comandos foram executados com sucesso, ou *"false"* caso contrário.

É definida uma variável de escopo global "e", que representa o autômato adaptativo de controle de acesso do modelo. Todos os comandos de privacidade podem referenciar-se ao autômato através da variável `"e".

A função $eval(u,e)$ recebe uma cadeia u e um autômato e, e retorna *"true"* se essa cadeia pertencer à linguagem descrita pelo autômato, $u \in L(e)$, ou *"false"* caso contrário.

A função $exec(u,e)$ recebe uma cadeia u e um autômato e, e retorna um autômato adaptativo. A cadeia u deve possuir algum dos símbolos que representam as alterações a serem realizadas na configuração do autômato. Ao executar a função $exec(u,e)$, espera-se que a cadeia altere a configuração do autômato; assim, o retorno da função será o autômato alterado, em caso de sucesso, ou o autômato original, caso a cadeia não pertença à linguagem descrita por e, $u \notin L(e)$, ou que u seja apenas uma cadeia de consulta.

A função $equiv(e,f)$ toma como argumentos os autômatos adaptativos e e f e verifica se são equivalentes pela inspeção de isomorfismo. Se e e f são isomorfos, a função $equiv(e,f)$ retorna*"true"*, ou *"false"* caso contrário.

As cadeias submetidas às funções são definidas através da concatenação de símbolos pré-definidos pertencentes ao alfabeto Σ e os argumentos do comando de privacidade $c_i, i \in \mathbb{N}$. De acordo com a gramática da linguagem do comando de privacidade, cada símbolo pré-definido $v_j \in \Sigma$ é representado na forma $< v_i >$, e os argumentos são representados através de seus nomes.

Através do Controle de Acesso Adaptativo, uma determinada legislação sobre privacidade pode ser formalizada em um conjunto de comandos de privacidade. A especificação formal de uma dada legislação possibilita o tratamento não ambíguo e obtenção de resultados que a teoria de autômatos adaptativos nos propicia, como tempo e custo de processamento, entre outros.

III. FRAMEWORK

Este artigo descreve um *framework* para privacidade, chamado *SPA*[3] (Sistema de Privacidade Auditável com Autômato Adaptativo). O *SPA*[3] tem como objetivo simplificar a especificação dos aspectos de privacidade de um determinado sistema. O *framework* pode ser utilizado em vários domínios, através da definição da representação do Controle de Acesso pelo Sistema de Privacidade Auditável.

O autômato adaptativo de controle de acesso é representado no *framework* utilizando a linguagem de marcação *XML*[1]. O arquivo *XML* possui a configuração inicial do autômato adaptativo de controle de acesso a ser utilizado. As *tags* do arquivo *XML* representam as transições do autômato, que podem ser na forma $P(q,a,\beta) = (q',a',\beta',A,B)$ ou $P(q,ab,\beta) = (q',a',\beta',A,B(b))$. Um exemplo de arquivo *XML* pode ser visto a seguir.

```
<autômato>
...
    <transicao>
        <origem> q </origem}>
        <simbolo> a </simbolo>
        <topo> β </topo>
        <destino> q' </destino>
        <novosimbolo> a' </novosimbolo>
        <novotopo> β' </novotopo>
        <funcao1> A(...) </funcao1>
        <funcao2> B(...) </funcao2>
    </transicao>
...
</automato>
```

O acesso ao *framework* é feito através de chamadas aos comandos de privacidade do conjunto C. Esses comandos são escritos de acordo com a sintaxe apresentada na seção II. Ao executar um comando de privacidade, o *framework* retorna um valor lógico, *true* se o comando foi satisfeito, ou *false* caso contrário.

A construção do *framework* requer o arquivo *XML*, contendo o autômato adaptativo, e os comandos de privacidade, de acordo com a Figura 1.

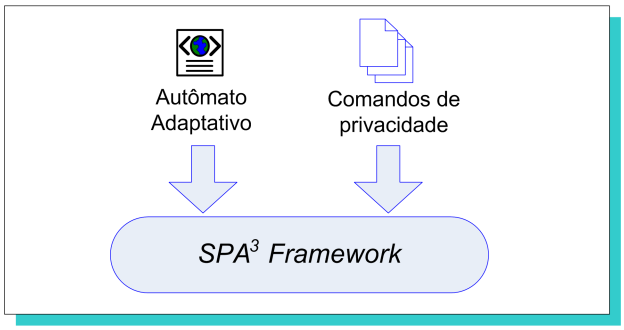

FIG. 1 Construção do *framework*.

O *SPA*[3] atua entre o usuário e o sistema, verificando se um determinado comando do sistema pode ser executado de acordo com o conjunto de comandos de privacidade definidos para o *framework*, conforme é ilustrado na Figura 2.

[1] *eXtensible Markup Language*

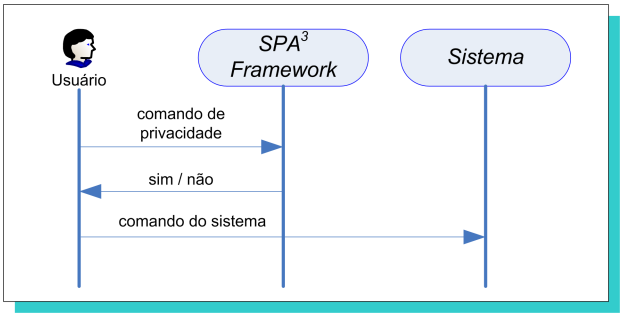

FIG. 2 Usuário, *framework* e sistema.

O *framework* fornece ao usuário um conjunto de comandos de privacidade (Figura 3). Os comandos verificam se uma determinada situação do sistema pode ser caracterizada como invasão de privacidade, de acordo com essa legislação.

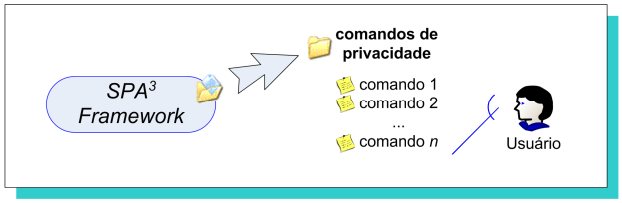

FIG. 3 Conjunto de comandos de privacidade disponíveis ao usuário.

Como exemplo, considere uma situação onde, em um determinado sistema, um usuário só poderá copiar um determinado arquivo para outro usuário, se este primeiro for o dono do arquivo a ser copiado, e o arquivo copiado deverá ser somente leitura (*read-only*). O comando de privacidade referente a essa situação pode ser codificado como a seguir (*e* é o autômato adaptativo).

CopiarArquivo(*a, // usuário dono do arquivo*
b, // usuário que receberá a cópia
c) // arquivo a ser copiado
início
// verifica se o usuário a é dono do arquivo c
// "o" representa "owner"
se *eval({a}{c} "o", e)* **então** {
// cria-se o arquivo c associado ao usuário c
tmp = exec({b} λ_{NO} *{c}, e)*
// se os autômatos são iguais, não foi possível
// criar o arquivo c para o usuário b
se *equals(tmp, e)* **então retornar** *false*
tmp = exec({b} $\lambda_{\{c\}}$ *"Gr", e)*
// se os autômatos são iguais, não foi possível
// inserir o direito de leitura r no arquivo c
// do usuário b
se *equals(tmp, e)* **então retornar** *false*
// o comando foi executado com sucesso
retornar *true*
}
senão {
// o usuário a não é dono do arquivo c
retornar *false*
}
fim

O comando **CopiarArquivo** é descrito utilizando-se a sintaxe apresentada, e o *framework* realizará a interpretação do mesmo, disponibilizando-o em seguida.

Ao tentar realizar uma operação de cópia de arquivo, o usuário solicita a permissão de execução da operação ao *SPA*^{3}, que analisará a situação e retornará um valor lógico. Se o retorno for *true*, o sistema realizará a operação de cópia de arquivo (Figura 4), caso contrário o sistema informará ao usuário que não foi possível realizar a operação de cópia porque a situação não está em conformidade com as regras definidas no comando de privacidade (Figura 5).

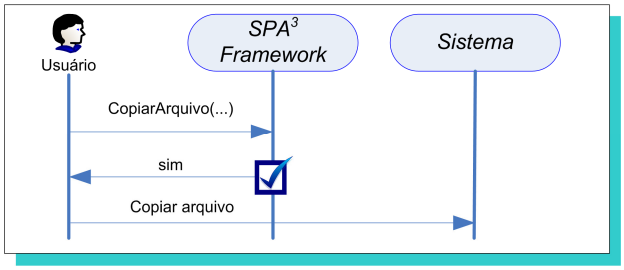

FIG. 4 Operação permitida pelo comando de privacidade.

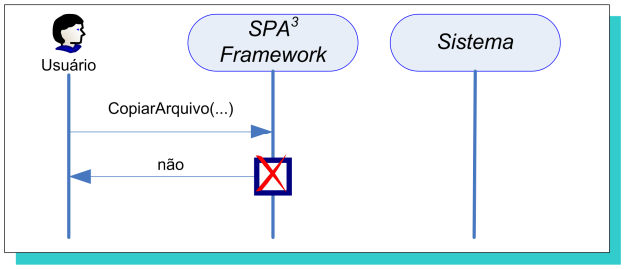

FIG. 5 Operação proibida pelo comando de privacidade.

O *SPA*^{3} permite que os comandos de privacidade sejam executados e avaliados em um sistema, aumentando o controle sobre as informações que trafegam e minimizando situações que caracterizem violação de privacidade.

IV. CONSIDERAÇÕES FINAIS

A utilização de um *framework* é justificada pela facilidade proporcionada no desenvolvimento de sistemas. Um *framework* para privacidade contribui para a construção simplificada de sistemas com reforço em privacidade de modo flexível e extensível.

O *framework SPA*^{3}, apresentado neste trabalho, provê uma solução que reforça os aspectos de privacidade de um determinado sistema. A utilização do formato *XML* para representação do autômato adaptativo proporciona um padrão bem definido, simples e formal para ser utilizado como entrada do *framework*. A modelagem do autômato adaptativo

através de *XML* é clara e concisa, além de permitir uma especificação completa do mesmo. Os comandos de privacidade possuem uma sintaxe de alto nível e de fácil representação, contribuindo para a especificação de situações de violação de privacidade em um sistema.

O trabalho apresentado contribui com a área de Tecnologias Adaptativas, conferindo um novo domínio de aplicação, e a utilização de um formalismo adaptativo provê ao modelo redução de tempo e espaço computacional.

O *SPA*^{3} é uma solução viável para incorporar reforços de privacidade nos sistemas. Os comandos de privacidade são reutilizáveis e podem ser executados a qualquer momento em um sistema, conferindo ao *framework* um nível de abstração para ser utilizado em diversas classes de problemas, sem sofrer modificações.

AGRADECIMENTOS

Os autores agradecem aos revisores do WTA 2008 pelas recomendações feitas para a elaboração da versão final do artigo.

REFERÊNCIAS

[1] C. H. Fernandes, A Privacidade na Sociedade da Informação, 2003..

[2] S. Fischer-Hübner, L. Yngström, and J. Holvast, "Addressing vulnerability and privacy problems generated by use of IT-security mechanisms," in *Proceedings of the IFIP 12th World Computer Congress on Education and Society – Information Processing '92 – Volume 2*. Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co., 1992, pp. 314-321.

[3] S. Fischer-Hübner, *IT-security and privacy: design and use of privacy-enhancing security mechanisms.* New York, NY, USA: Springer-Verlag New York, Inc., 2001.

[4] M. J. May, C. A. Gunter, and I. Lee, "Privacy APIs: Access control techniques to analyze and verify legal privacy policies," in *CSFW '06: Proceedings of the 19th IEEE workshop on Computer Security Foundations*. Washington, DC, USA: IEEE Computer Society 2006, pp. 85-97.

[5] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Commun. ACM,* vol. 19, no. 8, pp. 461-471, 1976.

[6] J. J. Neto, *Contribuições à Metodologia de Construção de Compiladores*, Thesis (Livre Docência) – Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

[7] _____, "Adaptive automata for context-dependent languages," *ACM SIGPLAN Notices*, vol. 29, no. 9, 1994..

[8] P. R. M. Cereda and S. D. Zorzo, "Formalismo com autômato adaptativo em mecanismo de privacidade e personalização," in *XXXIII Conferencia LatinoAmericana en Informatica - CLEI*. 2007, San Jose, Costa Rica, 2007.