

European Symposium on Research in Computer Security (ESORICS '09)

Dynamic Enforcement of Abstract Separation of Duty Constraints

Samuel Burri, IBM Research – Zurich

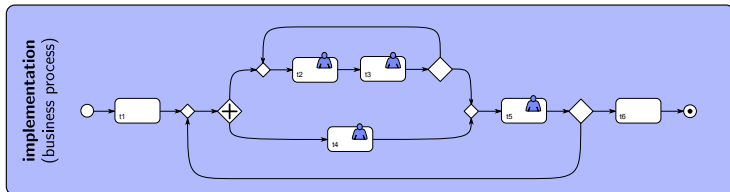
Joint work with:

David Basin, ETH Zurich

Günter Karjoth, IBM Research – Zurich

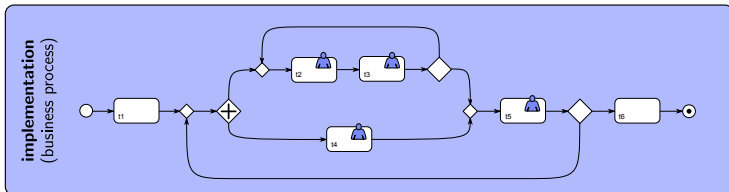
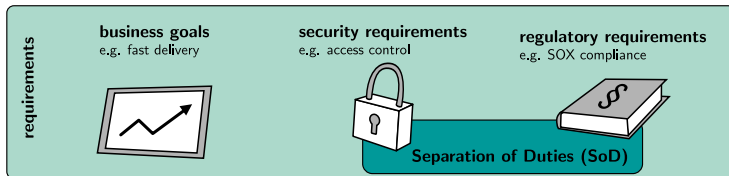
Saint Malo, France, September 22nd 2009

Problem & Approach



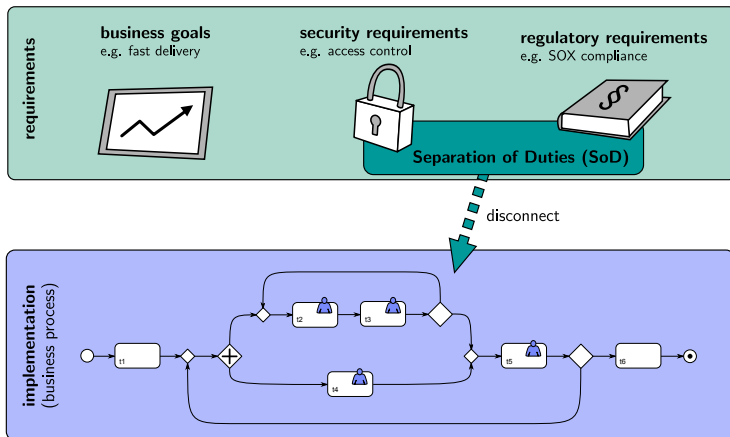
Picture simplifies reality: Refined system may be more complex than initial system.

Problem & Approach



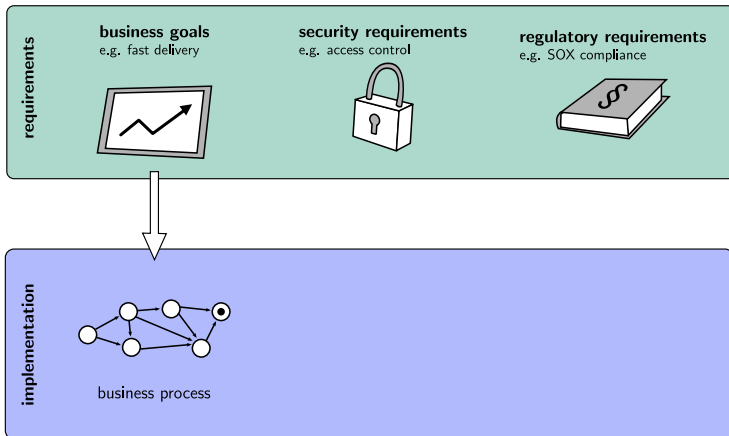
Picture simplifies reality: Refined system may be more complex than initial system.

Problem & Approach



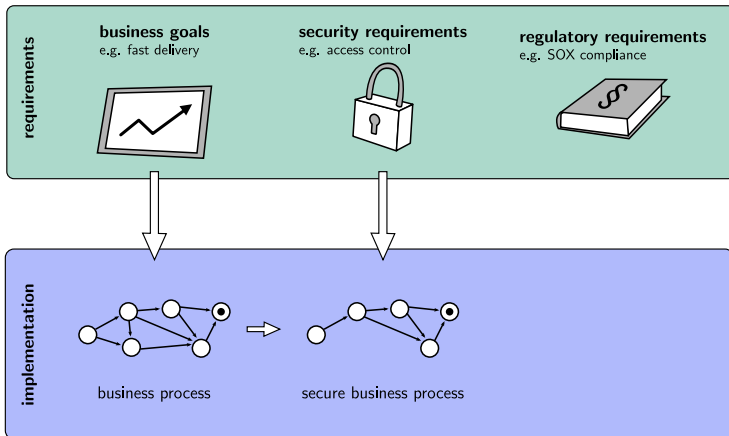
Picture simplifies reality: Refined system may be more complex than initial system.

Problem & Approach



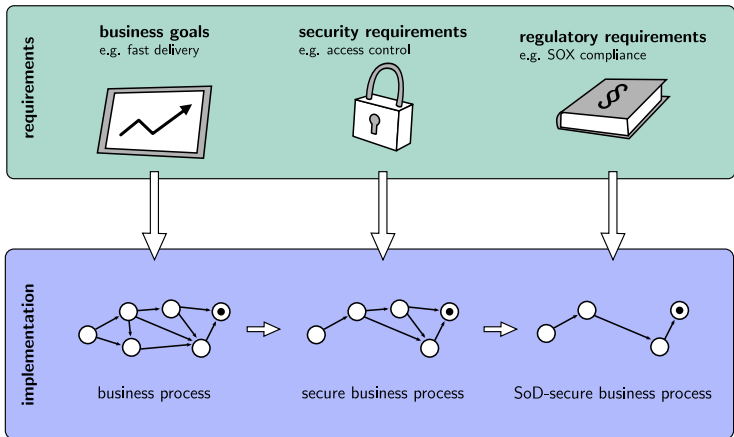
Picture simplifies reality: Refined system may be more complex than initial system.

Problem & Approach



Picture simplifies reality: Refined system may be more complex than initial system.

Problem & Approach



Picture simplifies reality: Refined system may be more complex than initial system.

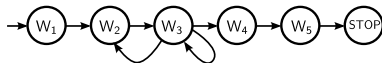
Content

- ① Motivation
- ② Modeling business processes and access control
- ③ SoD algebra over multisets
- ④ SoD algebra over traces
- ⑤ Mapping SoDA terms to CSP processes
- ⑥ Concluding remarks

Modeling business processes

- Set of **users** \mathcal{U}
- Set of **actions** \mathcal{A}
- **Business process** modeled as **CSP process** with **business channel** bc of type $\mathcal{U} \times \mathcal{A}$ and the event *done*
- $\mathcal{E}_B = \{|bc|\}$
set of **business events**

Example payment business process



$$W = W_1$$

$$W_1 = bc?u : \mathcal{U}.receive\ invoice \rightarrow W_2$$

$$W_2 = bc?u : \mathcal{U}.prepare\ check \rightarrow W_3$$

$$W_3 = (bc?u : \mathcal{U}.reject\ payment \rightarrow W_2)$$

$$\square (bc?u : \mathcal{U}.approve\ payment \rightarrow W_3)$$

$$\square (bc?u : \mathcal{U}.approve\ payment \rightarrow W_4)$$

$$W_4 = bc?u : \mathcal{U}.issue\ check \rightarrow W_5$$

$$W_5 = done \rightarrow STOP$$

Modeling access control

- Set of **roles** \mathcal{R}
- **RBAC configuration** is tuple (UA, PA)
for the **user-assignment relation** $UA \subseteq \mathcal{U} \times \mathcal{R}$ and
the **permission-assignment relation** $PA \subseteq \mathcal{R} \times \mathcal{A}$
- The RBAC process, engaging in **business events** and **admin events** $\mathcal{E}_A = \{ac\}$.

$$\begin{aligned}
 RBAC(UA, PA) = & \left(\text{ac} \rightarrow \text{RBAC} \mid \left(\text{ua} \rightarrow \text{RBAC} \mid \text{ra} \rightarrow \text{RBAC} \right) \right) \\
 & \square \left(\text{ua} \rightarrow \text{RBAC} \mid \left(\text{ra} \rightarrow \text{RBAC} \mid \text{ac} \rightarrow \text{RBAC} \right) \right) \\
 & \square \left(\text{ra} \rightarrow \text{RBAC} \mid \left(\text{ua} \rightarrow \text{RBAC} \mid \text{ac} \rightarrow \text{RBAC} \right) \right)
 \end{aligned}$$

- For a business process modeled by W , the **secure business process** is

$$SW(UA, PA) = W \parallel_{\mathcal{E}_B} RBAC(UA, PA).$$

Modeling access control

- Set of roles \mathcal{R}
- RBAC configuration is tuple (UA, PA)
for the user-assignment relation $UA \subseteq \mathcal{U} \times \mathcal{R}$ and
the permission-assignment relation $PA \subseteq \mathcal{R} \times \mathcal{A}$
- The RBAC process, engaging in business events and admin events $\mathcal{E}_A = \{|ac|\}$.

$$\begin{aligned}
 RBAC(UA, PA) &= (bc?(u.a) : \{u.a \mid \exists r \in \mathcal{R}. (u, r) \in UA \wedge (r, a) \in PA\} \rightarrow RBAC(UA, PA)) \\
 &\quad \square (ac.addUA?u : \mathcal{U}?r : \mathcal{R} \rightarrow RBAC(UA \cup \{(u, r)\}, PA)) \\
 &\quad \square (ac.rmUA?u : \mathcal{U}?r : \mathcal{R} \rightarrow RBAC(UA \setminus \{(u, r)\}, PA))
 \end{aligned}$$

- For a business process modeled by W , the secure business process is

$$SW(UA, PA) = W \parallel_{\mathcal{E}_B} RBAC(UA, PA).$$

Modeling access control

- Set of roles \mathcal{R}
- RBAC configuration is tuple (UA, PA)
for the user-assignment relation $UA \subseteq \mathcal{U} \times \mathcal{R}$ and
the permission-assignment relation $PA \subseteq \mathcal{R} \times \mathcal{A}$
- The RBAC process, engaging in business events and admin events $\mathcal{E}_A = \{|ac|\}$.

$$\begin{aligned}
 RBAC(UA, PA) &= (bc?(u.a) : \{u.a \mid \exists r \in \mathcal{R}. (u, r) \in UA \wedge (r, a) \in PA\} \rightarrow RBAC(UA, PA)) \\
 &\quad \square (ac.addUA?u : \mathcal{U}?r : \mathcal{R} \rightarrow RBAC(UA \cup \{(u, r)\}, PA)) \\
 &\quad \square (ac.rmUA?u : \mathcal{U}?r : \mathcal{R} \rightarrow RBAC(UA \setminus \{(u, r)\}, PA))
 \end{aligned}$$

- For a business process modeled by W , the secure business process is

$$SW(UA, PA) = W \parallel_{\mathcal{E}_B} RBAC(UA, PA).$$

Content

- 1 Motivation
- 2 Modeling business processes and access control
- 3 SoD algebra over multisets**
- 4 SoD algebra over traces
- 5 Mapping SoDA terms to CSP processes
- 6 Concluding remarks

SoD algebra (SoDA)¹ intuition

Motivating examples

- **Three different users** must execute actions in the business process.
All \otimes All \otimes All
- One action must be executed by a **Manager but not Bob**. Two further actions must be **executed by different users**, one in the role of an **Accountant** and one in the role of a **Clerk**.
(Manager $\sqcap \neg\{\text{Bob}\}$) \otimes (Accountant \odot Clerk)

SoDA combines

- **quantitative** requirements, i.e. how many users
- **qualitative** requirements, i.e. which kinds of users

¹Li, N., Wang, Q.: Beyond separation of duty: [...]. J. of the ACM, 2008

SoD algebra (SoDA)¹ intuition

Motivating examples

- **Three different users** must execute actions in the business process.
All \otimes All \otimes All
- One action must be executed by a **Manager but not Bob**. Two further actions must be **executed by different users**, one in the role of an **Accountant** and one in the role of a **Clerk**.
(Manager \sqcap \neg {Bob}) \otimes (Accountant \odot Clerk)

SoDA combines

- **quantitative** requirements, i.e. how many users
- **qualitative** requirements, i.e. which kinds of users

¹Li, N., Wang, Q.: Beyond separation of duty: [...]. J. of the ACM, 2008

SoDA syntax

An **atomic term** is either

- a set of users $U \subseteq \mathcal{U}$
- a role $r \in \mathcal{R}$
- the keyword All

A **term** is either

- an atomic term
- $\neg\phi$
- ϕ^+
- $\phi \sqcup \psi$
- $\phi \sqcap \psi$
- $\phi \otimes \psi$
- $\phi \odot \psi$

for two terms ϕ and ψ

A **unit term** is a term that does not contain the operators $^+$, \otimes , and \odot .

SoDA multiset semantics $\text{SoDA}^{\mathcal{M}}$

For a **multiset of users** \mathbf{U} , a **term** ϕ , and a **user-assignment relation** UA .

Multiset satisfiability of ϕ by \mathbf{U} w.r.t. UA , written $\mathbf{U} \models_{UA}^{\mathcal{M}} \phi$, is defined by rules such as:

$$\frac{}{\{u\} \models_{UA}^{\mathcal{M}} \text{All}} \quad \exists r \in \mathcal{R}. (u, r) \in UA$$

$$\frac{}{\{u\} \models_{UA}^{\mathcal{M}} r} \quad (u, r) \in UA$$

$$\frac{}{\{u\} \models_{UA}^{\mathcal{M}} U} \quad u \in U \text{ and } \exists r \in \mathcal{R}. (u, r) \in UA$$

$$\frac{\{u\} \not\models_{UA}^{\mathcal{M}} \phi}{\{u\} \models_{UA}^{\mathcal{M}} \neg \phi}$$

$$\frac{\mathbf{U} \models_{UA}^{\mathcal{M}} \phi, \mathbf{V} \models_{UA}^{\mathcal{M}} \psi}{(\mathbf{U} \uplus \mathbf{V}) \models_{UA}^{\mathcal{M}} (\phi \odot \psi)}$$

$$\frac{\mathbf{U} \models_{UA}^{\mathcal{M}} \phi, \mathbf{V} \models_{UA}^{\mathcal{M}} \psi}{(\mathbf{U} \uplus \mathbf{V}) \models_{UA}^{\mathcal{M}} (\phi \otimes \psi)} \quad (\mathbf{U} \cap \mathbf{V}) = \emptyset$$

For $r \in \mathcal{R}$, $u \in U$, $U \subseteq \mathcal{U}$, and \mathbf{U} and \mathbf{V} two multisets of users.

Unit terms are only satisfied by multisets containing **one user**.

SoDA^M examples

- $UA = \{(Alice, Manager), (Bob, Accountant), (Bob, Clerk), (Claire, Customer)\}$
- $\{Alice, Bob, Claire\} \models_{UA}^M All \otimes All \otimes All$
- $\{Alice, Bob, Bob\} \not\models_{UA}^M All \otimes All \otimes All$
- $\{Alice, Bob, Bob\} \models_{UA}^M (Manager \sqcap \neg\{Bob\}) \otimes (Accountant \odot Clerk)$
- $\{Bob, Bob, Claire\} \not\models_{UA}^M (Manager \sqcap \neg\{Bob\}) \otimes (Accountant \odot Clerk)$

Content

- 1 Motivation
- 2 Modeling business processes and access control
- 3 SoD algebra over multisets
- 4 SoD algebra over traces
- 5 Mapping SoDA terms to CSP processes
- 6 Concluding remarks

Traces prerequisites

Given the traces $t, t_1, t_2 \in (\mathcal{E}_B \cup \mathcal{E}_A)^*$.

The **synchronized interleaving** predicate $si(t, t_1, t_2)$ holds

if t_1 and t_2 “partition” t such that

- each **business event** $b_i \in \mathcal{E}_B$ in t is either in t_1 or t_2 (but not in both)
- each **admin event** $a_j \in \mathcal{E}_A$ in t is both in t_1 and t_2
- The relative order of the events in $t, t_1,$ and t_2 is the same.

Example: $si(t, t_1, t_2)$ holds for

$$\begin{aligned} t &= \langle b_1, b_2, b_3, a_1, b_4, a_2, b_5, a_3, b_6, a_4 \rangle \\ t_1 &= \langle b_1, b_3, a_1, b_4, a_2, a_3, b_6, a_4 \rangle \\ t_2 &= \langle b_2, a_1, a_2, b_5, a_3, a_4 \rangle \end{aligned}$$

The auxiliary function **users** returns the multiset of users in business events in a trace.

$$\text{users}(\langle bc.Bob.a_1, bc.Alice.a_2, ac.addUA.Claire.a_3, bc.Bob.a_4 \rangle) = \{\text{Bob}, \text{Bob}, \text{Alice}\}$$

Traces prerequisites

Given the traces $t, t_1, t_2 \in (\mathcal{E}_B \cup \mathcal{E}_A)^*$.

The **synchronized interleaving** predicate $si(t, t_1, t_2)$ holds

if t_1 and t_2 “partition” t such that

- each **business event** $b_i \in \mathcal{E}_B$ in t is either in t_1 or t_2 (but not in both)
- each **admin event** $a_j \in \mathcal{E}_A$ in t is both in t_1 and t_2
- The relative order of the events in t , t_1 , and t_2 is the same.

Example: $si(t, t_1, t_2)$ holds for

$$\begin{aligned} t &= \langle b_1, b_2, b_3, a_1, b_4, a_2, b_5, a_3, b_6, a_4 \rangle \\ t_1 &= \langle b_1, b_3, a_1, b_4, a_2, a_3, b_6, a_4 \rangle \\ t_2 &= \langle b_2, a_1, a_2, b_5, a_3, a_4 \rangle \end{aligned}$$

The auxiliary function **users** returns the multiset of users in business events in a trace.

$$\text{users}(\langle bc.Bob.a_1, bc.Alice.a_2, ac.addUA.Claire.r, bc.Bob.a_3 \rangle) = \{\text{Bob}, \text{Bob}, \text{Alice}\}$$

SoDA traces semantics SoDA^T

For traces $t, t_1, t_2 \in (\mathcal{E}_B \cup \mathcal{E}_A)^*$, a user-assignment relation UA , a term ϕ , and a unit term ϕ_{ut} .

Trace satisfiability of ϕ by t w.r.t. UA , written $t \models_{UA}^T \phi$, is defined by rules such as:

$$\frac{\{\text{user}(b)\} \models_{UA}^M \phi_{ut}}{\langle b \rangle \models_{UA}^T \phi_{ut}}$$

$$\frac{t \models_{UA \cup \{(u,r)\}}^T \phi}{\langle \text{ac.add}UA.u.r \rangle \wedge t \models_{UA}^T \phi}$$

$$\frac{t_1 \models_{UA}^T \phi, t_2 \models_{UA}^T \psi}{t \models_{UA}^T \phi \odot \psi} \quad \text{si}(t, t_1, t_2)$$

$$\frac{t_1 \models_{UA}^T \phi, t_2 \models_{UA}^T \psi}{t \models_{UA}^T \phi \otimes \psi} \quad \text{si}(t, t_1, t_2) \\ \text{users}(t_1) \cap \text{users}(t_2) = \emptyset$$

For $b \in \mathcal{E}_B$.

SoDA^T examples

- $UA = \{(Alice, Manager), (Bob, Accountant), (Bob, Clerk), (Claire, Customer)\}$
- $\langle bc.Alice.a_1, bc.Bob.a_2, bc.Claire.a_3 \rangle \models_{UA}^T All \otimes All \otimes All$
- $\langle bc.Alice.a_1, bc.Bob.a_2, bc.Bob.a_3 \rangle \not\models_{UA}^T All \otimes All \otimes All$
- $\{Bob, Bob, Claire\} \not\models_{UA}^M (Manager \sqcap \neg\{Bob\}) \otimes (Accountant \odot Clerk)$
- $\langle bc.Bob.a_1, bc.Bob.a_2, ac.addUA.Claire.Manager, bc.Claire.a_3 \rangle$
 $\models_{UA}^T (Manager \sqcap \neg\{Bob\}) \otimes (Accountant \odot Clerk)$

Content

- 1 Motivation
- 2 Modeling business processes and access control
- 3 SoD algebra over multisets
- 4 SoD algebra over traces
- 5 Mapping SoDA terms to CSP processes**
- 6 Concluding remarks

Mapping SoDA terms to CSP processes

For a set of users U , a user-assignment relation UA , terms ϕ and ψ .

The mapping $\llbracket \cdot \rrbracket_{UA}^U$ returns a CSP process parametrized by UA . Extract from rules:

$$\llbracket \phi_{ut} \rrbracket_{UA}^U = bc?u : \{u' \in U \mid \{u'\} \models_{UA}^M \phi_{ut}\}.a : \mathcal{A} \rightarrow FIN$$

$$\square ac.addUA?u : U?r : \mathcal{R} \rightarrow \llbracket \phi_{ut} \rrbracket_{UA \cup \{(u,r)\}}^U$$

$$\square ac.rmUA?u : U?r : \mathcal{R} \rightarrow \llbracket \phi_{ut} \rrbracket_{UA \setminus \{(u,r)\}}^U$$

$$\llbracket \phi \odot \psi \rrbracket_{UA}^U = \llbracket \phi \rrbracket_{UA}^U \parallel_{\mathcal{E}_A \cup \{done\}} \llbracket \psi \rrbracket_{UA}^U$$

$$\llbracket \phi \otimes \psi \rrbracket_{UA}^U = \{ (U_\phi, U_\psi) \mid U_\phi \cup U_\psi = U \text{ and } U_\phi \cap U_\psi = \emptyset \} \llbracket \phi \rrbracket_{U_\phi}^{U_\phi} \parallel_{\mathcal{E}_A \cup \{done\}} \llbracket \psi \rrbracket_{U_\psi}^{U_\psi}$$

SoD-secure business process

- The CSP process $SOD_\phi(UA) = \llbracket \phi \rrbracket_{UA}^U$ is called **SoD-enforcement process**.
- For a business process modeled by W , the SoD-secure business process is

$$\begin{aligned}
 SSW_\phi(UA, PA) &= SW(UA, PA) \parallel_{\mathcal{E}_B \cup \mathcal{E}_A \cup \{done\}} SOD_\phi(UA) \\
 &= (W \parallel_{\mathcal{E}_B} RBAC(UA, PA)) \parallel_{\mathcal{E}_B \cup \mathcal{E}_A \cup \{done\}} SOD_\phi(UA)
 \end{aligned}$$

- In the paper, we prove a suitable notion of **correctness** for the mapping $\llbracket \cdot \rrbracket_{UA}^U$.

SoD-secure business process

- The CSP process $SOD_\phi(UA) = \llbracket \phi \rrbracket_{UA}^U$ is called **SoD-enforcement process**.
- For a business process modeled by W , the **SoD-secure business process** is

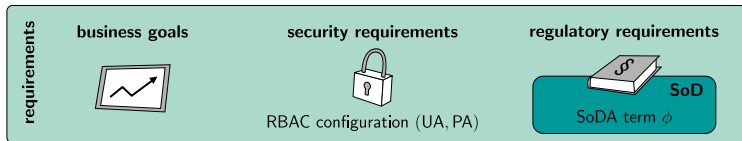
$$\begin{aligned}
 SSW_\phi(UA, PA) &= SW(UA, PA) \underset{\mathcal{E}_B \cup \mathcal{E}_A \cup \{done\}}{\parallel} SOD_\phi(UA) \\
 &= (W \underset{\mathcal{E}_B}{\parallel} RBAC(UA, PA)) \underset{\mathcal{E}_B \cup \mathcal{E}_A \cup \{done\}}{\parallel} SOD_\phi(UA)
 \end{aligned}$$

- In the paper, we prove a suitable notion of **correctness** for the mapping $\llbracket \cdot \rrbracket_{UA}^U$.

Content

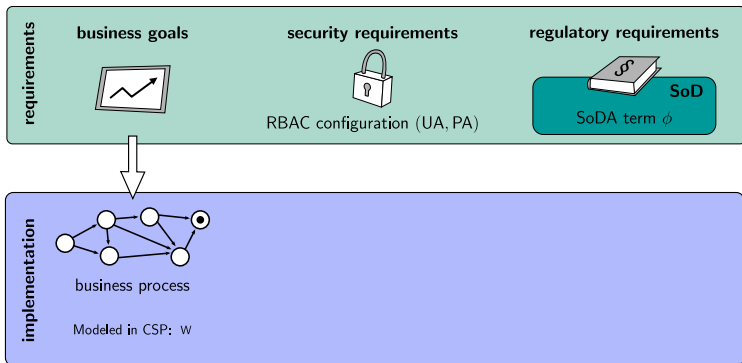
- 1 Motivation
- 2 Modeling business processes and access control
- 3 SoD algebra over multisets
- 4 SoD algebra over traces
- 5 Mapping SoDA terms to CSP processes
- 6 Concluding remarks

Summing up



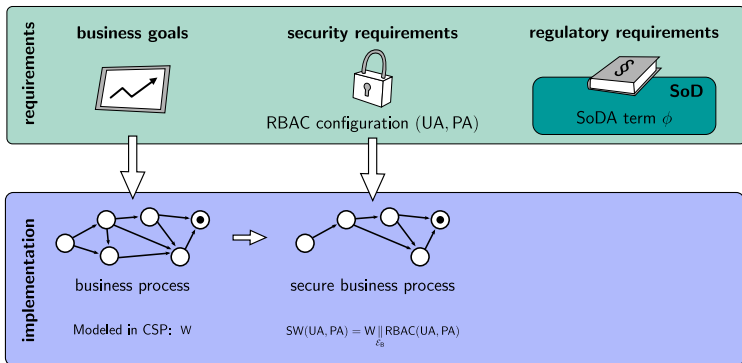
- Closed gap between high-level SoD constraint specification and dynamic enforcement.
- Enforcement of SoD constraints in the presence of administrative changes.

Summing up



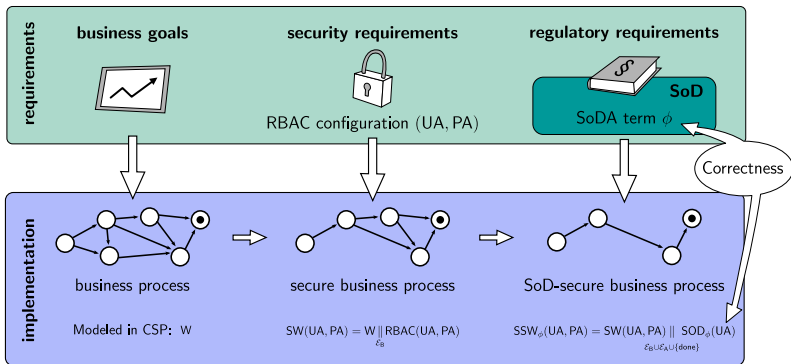
- Closed gap between high-level SoD constraint specification and dynamic enforcement.
- Enforcement of SoD constraints in the presence of administrative changes.

Summing up



- Closed gap between high-level SoD constraint specification and dynamic enforcement.
- Enforcement of SoD constraints in the presence of administrative changes.

Summing up



- **Closed gap** between high-level SoD constraint specification and dynamic enforcement.
- Enforcement of SoD constraints in the presence of **administrative changes**.

Present and future work

Implementation

- **Integration** into real-world process engine
(we use IBM WebSphere Process Server)
- **Efficiency**: How to go **beyond** a simple mapping from the **operational semantics** of CSP to a reference monitor.
 - **Data-structures** for efficient representation of extended state-machines.
 - **Optimization techniques** such as pruning states never visited.