Imperial College London

On the Modeling and Verification of Security-Aware and Process-Aware Information Systems

Michael Huth & Jason Crampton

29 August 2011

What are workflows to us?

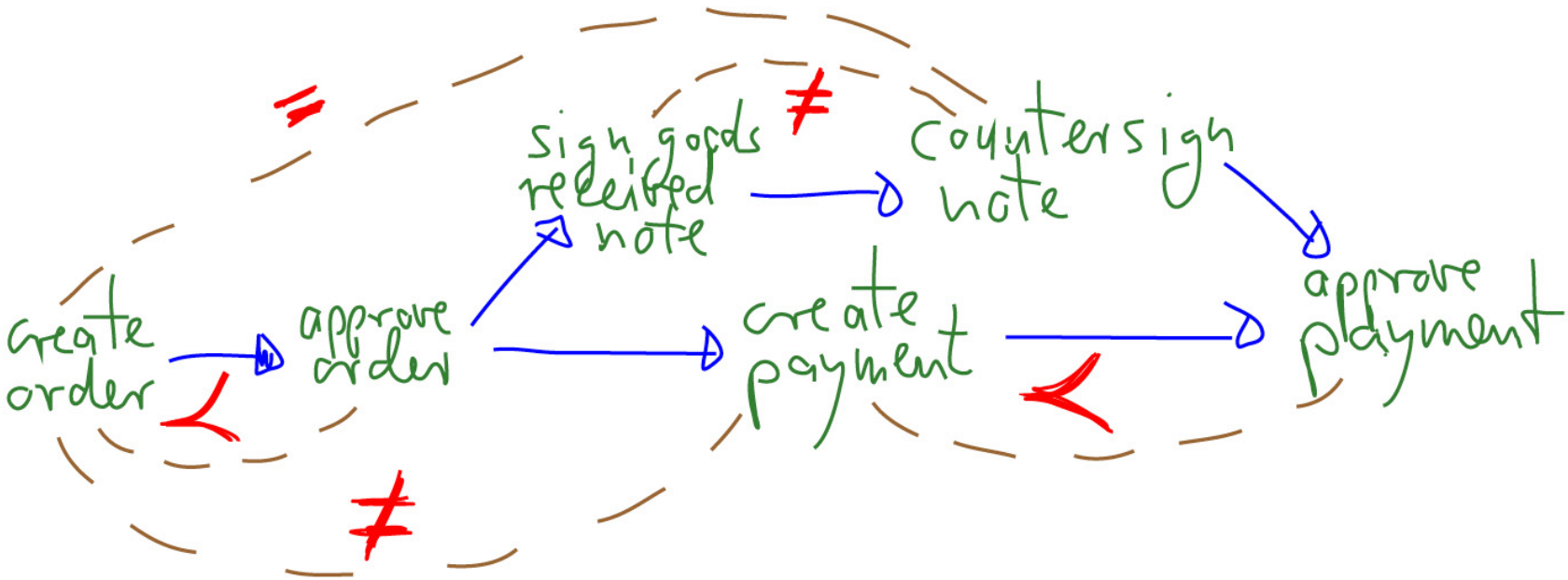
- Plans or schedules that map users or resources to tasks
- Such mappings may be constrained, e.g. Binding of Duty
- Security policy may prevent some user/task combinations
- Business objectives or legal requirements may further constrain workflow
- Temporal order of tasks may be constrained

A workflow is such a plan that meets all constraints.

Why are workflows interesting?

- Important technology, e.g.
  - Business process management systems
  - Cloud-based collaboration services, e.g. `inkspotscience.com`
- Industrial practice of workflows is
  - often flawed and uses ad hoc methods
  - rarely takes into account security considerations
- Academic methods brittle under change of models
- Most problems NP-hard
- Model-based approaches to design and analysis of workflows have potential impact

Example workflow specification



Blue edges: temporal constraints. Binding of users to tasks constrained by $=$, $\neq$, and seniority $\prec$.

Representative specification formalism

Specification of authorization system $\mathcal{AS}$ comprised of:

- $(T, \leq)$ finite partial order of tasks:
  $t < t'$ means t has to precede t'
- U set of users
- $A \subseteq T \times U$ where (t, u) in A means:
  u authorized to execute task t
- C set of entailment constraints of form $(D, t \rightarrow t', \rho)$
- $D \subseteq U$ and $\rho \subseteq U \times U$
- meaning: if u in D and assigned to task t, then user u' assigned to t' such that (u, u') is in ρ
- e.g. $=$ as ρ and D as U gives Binding of Duty

Synthesizing secure workflows in LTL(F)

- Translate a workflow specification $\mathcal{AS}$ into formula $\phi_{\mathcal{AS}}$ of NP-complete linear-time temporal logic fragment
- Show: authorized workflow translates into model of $\phi_{\mathcal{AS}}$
- Conversely, show that any model of $\phi_{\mathcal{AS}}$ translates into authorized workflow
- So we can synthesize authorized workflows for $\mathcal{AS}$ by
  - generating $\phi_{\mathcal{AS}}$ from $\mathcal{AS}$
  - running a model checker on the fully connected model . . .
  - . . . with the negation of $\phi_{\mathcal{AS}}$ as query

Temporal logic LTL(F)

- Syntax where p is from set of atomic propositions AP:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \mathsf{F}\,\phi$$

- F temporal connective "Future", and "Globally" $\mathsf{G}\,\phi$ is defined as $\neg\mathsf{F}\neg\phi$
- Semantics via infinite sequence of states $\pi = s_0 s_1 \ldots$ where each s_i subset of AP:

$\pi \models p$ iff $p \in s_0$

$\pi \models \neg\phi$ iff not $\pi \models \phi$

$\pi \models \phi_1 \wedge \phi_2$ iff $(\pi \models \phi_1$ and $\pi \models \phi_2)$

$\pi \models \mathsf{F}\,\phi$ iff there is $i \geq 0$ with $\pi^i \models \phi$,
where π^i is the infinite suffix $s_i s_{i+1} \ldots$ of π

Formula $\phi_{\mathcal{AS}}$ for model checker

$$\phi_{\mathsf{F}T} = \bigwedge_{t \in T} \mathsf{F}\, t \qquad \phi_{\mathsf{G}T} = \mathsf{G}\left(\bigvee_{t \in T} t\right) \qquad \phi_{\mathsf{G}U} = \mathsf{G}\left(\bigvee_{u \in U} u\right)$$

$$\phi_{\leq} = \bigwedge_{t \in T} \mathsf{G}\left(t \rightarrow \mathsf{G}\left(\bigvee_{t' \not\leq t} t'\right)\right)$$

$$\phi_{seU} = \bigwedge_{u \in U} \mathsf{G}\left(u \rightarrow \bigwedge_{u' \in U \setminus \{u\}} \neg u'\right)$$

$$\phi_{seT} = \bigwedge_{t \in T} \mathsf{G}\left(t \rightarrow \bigwedge_{t' \in T \setminus \{t\}} \neg t'\right)$$

$$\phi_A = \bigwedge_{t \in T} \mathsf{G}\left(t \rightarrow \bigvee_{(t,u) \in A} u\right) \qquad \phi_C = \bigwedge_{(D, t \rightarrow t', \rho) \in C} \phi_{(D, t \rightarrow t', \rho)}$$

$$\phi_{(D, t \rightarrow t', \rho)} = \bigvee_{u \in D} \left(\mathsf{F}\left(t \wedge u\right)\right) \rightarrow \mathsf{G}\left(t' \rightarrow \bigvee_{(u,u') \in \rho} u'\right)$$

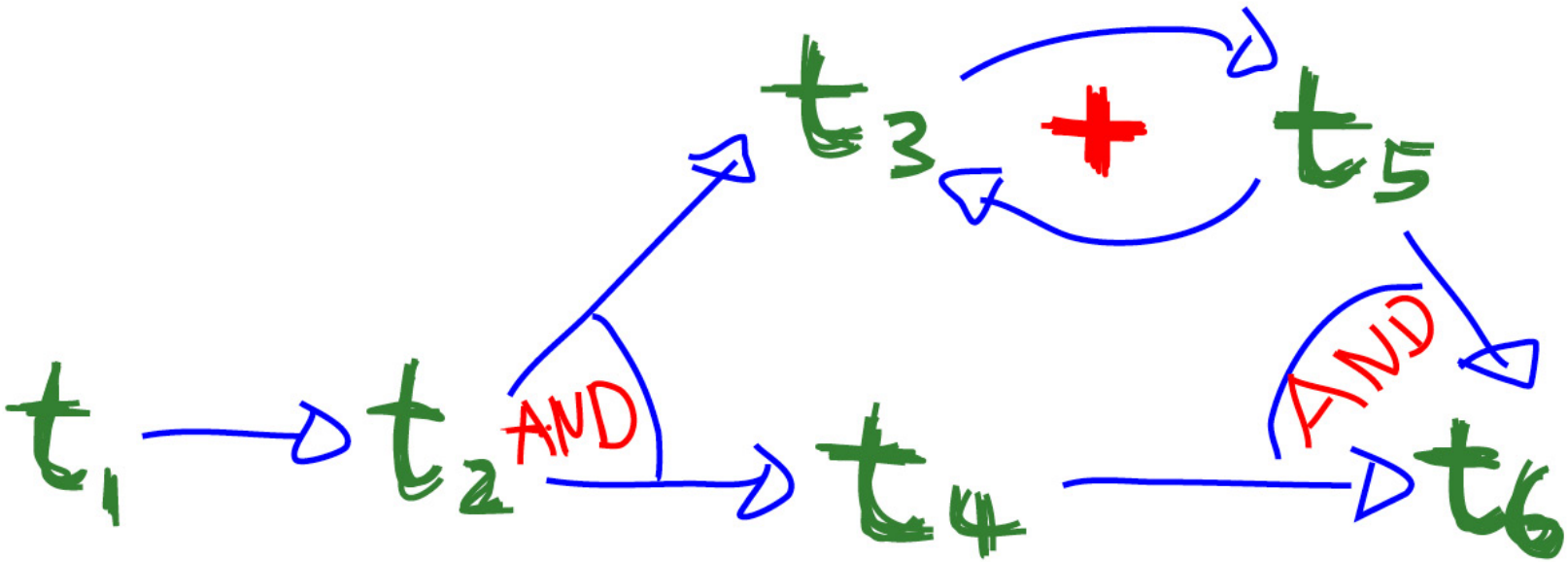
$$\phi_{\mathcal{AS}} = \phi_{\mathsf{F}T} \wedge \phi_{\mathsf{G}T} \wedge \phi_{\mathsf{G}U} \wedge \phi_{\leq} \wedge \phi_{seU} \wedge \phi_{seT} \wedge \phi_A \wedge \phi_C$$

Parameterized analysis tool

- Model-checking algorithm works for all formulas of LTL(F)
- No need to invent new analyses, if written in LTL(F), e.g.
- Schedulability with constraints across workflow instances:
  - write $\phi'_{\mathcal{AS}}$ for $\phi_{\mathcal{AS}}$ with each p replaced by p'
  - check two instances of workflow are realizable where . . .
  - . . . task t executed by different users in each instance:

$$\phi_{\mathcal{AS}} \wedge \phi'_{\mathcal{AS}} \wedge \bigwedge_{u \in U} (\mathsf{F}\,(t \wedge u) \rightarrow \mathsf{G}(t' \rightarrow \neg u'))$$

More expressive workflows



Would like to support AND/OR joins and forks as well for authorization frameworks.

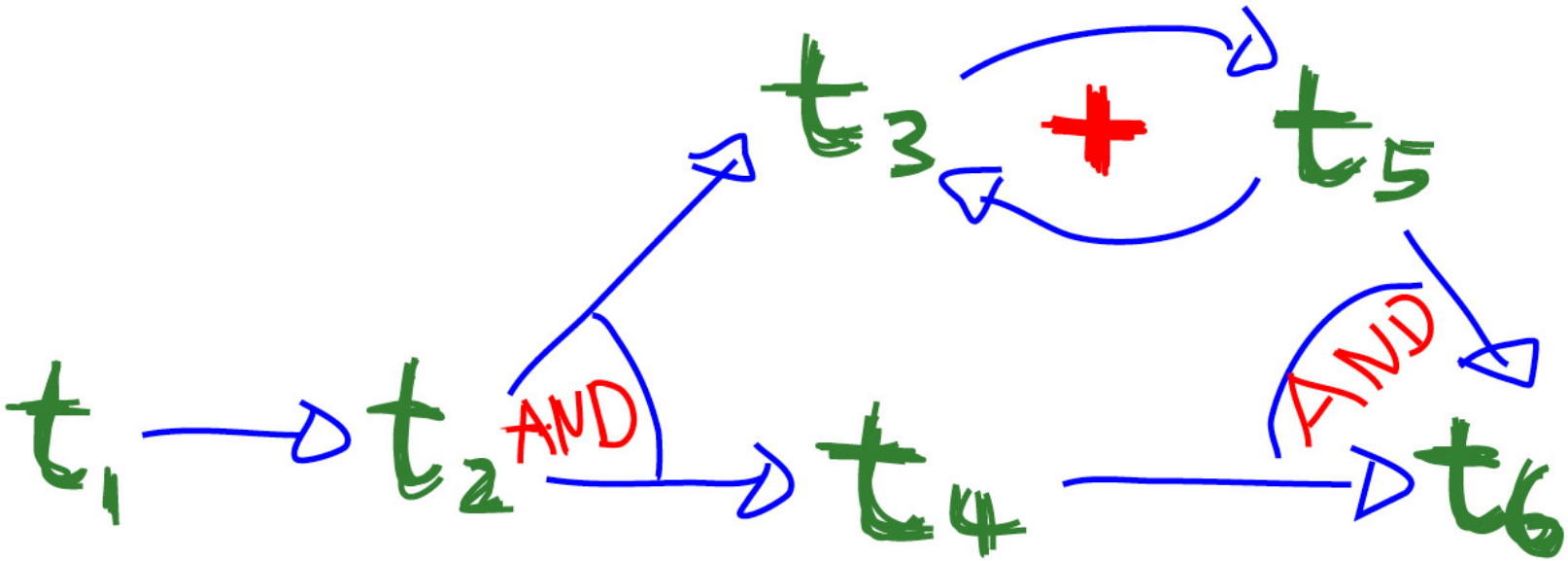
Toy declarative specification language

V, W	$::=$	Workflows
	t	(Atomic Workflow)
	W^+	(Unbounded Iteration)
	$V; W$	(Sequential Composition)
	choose k from $\mathbf{W}$	(Threshold Choice)

- choose k from $\mathbf{W}$ means exactly k workflow specifications from set $\mathbf{W}$ scheduled
- gives OR-fork and OR-join for $k = 1$
- gives AND-fork and AND-join for $k = |\mathbf{W}|$

A declaration and its graphical representation

$(t_1; t_2)$; choose 2 from $\{(t_3; t_5)^+, t_4\}$; t_6



Challenges

- LTL(F) satisfiability checks practical approach?
- iteration (W^+): temporal logics cannot count that well
- automata can count, but don't like constraints too well
- can constraint satisfaction solvers, SMT solvers, or planning tools deal with temporal expressiveness?
- van der Aalst's approach originally used Petri nets:
  - very good expressiveness
  - but core analysis problem Reachability EXPTIME-hard
  - their declarative language uses linear-time temporal logic

Opportunities

- Pattern-Based Modeling and Verification,
  e.g. parameterized modeling and analysis:
- Knowledge and tool transfer:
  Software verification of domain specific languages (DSL)
- Empirical Evaluation of Formal Methods for such DSLs
- Workflow and Collaborations in the Cloud:
  leverage work on multiple agents and imperfect information
- Novel forms of analysis,
  e.g. “dead task” detection, run-time synthesis

References

- Rozier, K. Y., and Vardi, M. Y. (2010) LTL satisfiability checking. Software Tools and Technology Transfer 12:123-137
- van der Aalst, W., Pesic, M., Schonenberg, H. (2009) Declarative workflows: Balancing between flexibility and support. Computer Science - R & D 23(2):99-113
- van der Aalst, W., and ter Hofstede, A. (2005) YAWL: yet another workflow language. Information Systems 30(4):245-275
- Sistla, A. P., and Clarke, E. M. (1985) The complexity of propositional linear temporal logics. Journal of the ACM 32:733-749

Q & A

Q & A

Q & A

Q & A

Q & A

Q & A

Q & A

Q & A

Q & A

Q & A