Evaluating the Risk of Information Leakage in Security-Sensitive Software Process

9th COE Postdoctoral and Doctoral Researchers
Technical Presentation
December 22nd , 2004

Yuichiro Kanzaki
Software Engineering Laboratory,
Graduate School of Information Science,
Nara Institute of Science and Technology

Background

Some software development process have security-sensitive information.

Example of security-sensitive information:

- Secret Source code
- Personal data included in work products (e.g. test data)
- Secret keys of DRM (Digital Rights Management) application

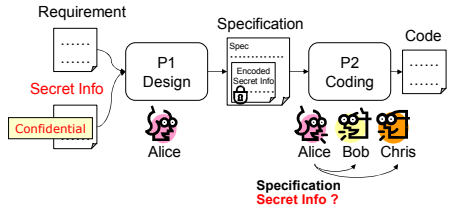
A slipshod management of secret information causes leakage of personal data[1] or source code[2].

[1] "Firms struggling to plug customer information leaks", Mainichi Shimbun, March 2, 2004.
[2] "Microsoft's code leakage", *Conrante Tech News.*,http://www.corante.com/openmind/archives/001884.php

Information leakage in Software Process

Information leakage in a software process can incur the problems.

Information Leakage (in a software process):
knowledge transfer with irrelevant work products



There are no methods to evaluate the risk of information leakage.

Goal and Approach

Goal

To propose a framework to evaluate *the risk of the information leakage* in software developing process quantitatively.

Approach

- We formulate the problem of information leakage by introducing a formal software process model.
- We present a method to compute a probability that each developer knows each product.

The probability reflects the risk that someone *leaked* the product to the developer.

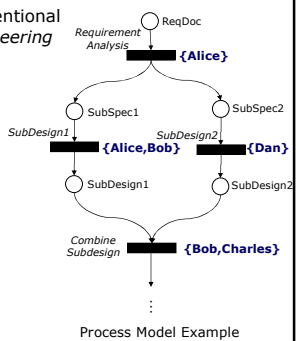
Definition: Software Process Model

The model is based on the conventional *process-centered software engineering environment*[1].

- A software process consists of a series of sub-processes.
- Each sub-process has a set of input products and a set of output products.
- Each sub-process has a set of developers.

Functions:

I(*p*) :input products of *p*
O(*p*) :output products of *p*
AS(*p*): developers engaging in *p*



Process Model Example

[1] P. K. Garg and M. Jazayeri, Process-Centered Software Engineering Environments, IEEE Computer Society Press, 1995.

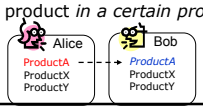
Definition: Knowledge of Developers

knowledge: idea , mechanism or the product's document itself

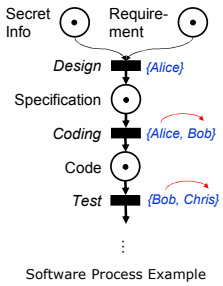
Knowledge of developers changes when a sub-process is performed.

knowledge acquisition:
All developers ∈AS(P) know both I(P) and O(P).

knowledge leakage:
All developers ∈AS(P) tell others the knowledge of product *in a certain probability*.



Knowledge Transfer Example



Software Process Example

Developers' Product Knowledge

	Alice	Bob	Chris
Require-ment	○	△	△
Secret Info	○	△	△
Specificati on	○	○	△
Code	○	○	○

○ : know
△ : have a chance to know

We want to derive a probability that each developer knows each product.

Product Knowledge

Product Knowledge Pkn(u,p,w):
Probability that a developer u knows a work product w at the completion of a process p

$$Pkn(u,p,w) = \begin{cases} 1.0 & (\cdots \textbf{if} \quad w \in Know(u,p)) \\ Pkn(u,pred_u(p),w) + C(u,p) * \{1 - \prod_{u_i \in AS(p)-\{u\}} \{1 - Pkn(u_i,pred_{u_i}(p),w) * leak(u_i,w,u)\}\} & (\cdots \textbf{if} \quad w \notin Know(u,p)) \end{cases}$$

Know(u,p) : A set of product that u knows at the completion of a process p
$pred_u(p)$: A process that u performed just before performing p
leak(ui, w, u): Probability that u_i leaks w to u
C(u,p) = 1 iff $u \in$ AS(p), otherwise C(u,p) = 0

Example(1/3)

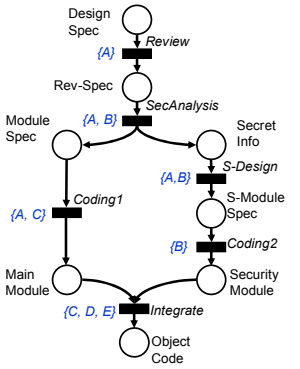
```
Developer = { A, B, C, D, E}
Product = { DesignSpec, Rev-Spec, SecretInfo,
        ModuleSpec, S-ModuleSpec, MainModule,
        SecurityModule, ObjectCode }
SubProcess = {Review, SecAnalysis, S-Design,
        Coding1, Coding2, Integrate }

I(Review)={DesignSpec}
I(SecAnalysis)={Rev-Spec}
I(S-Design)={SecretInfo}
I(Coding1)={ModuleSpec}
I(Coding2)={S-ModuleSpec}
I(Integrate)={MainModule, SecurityModule}

O(Review)={Rev-Spec}
O(SecAnalysis)={ModuleSpec,SecretInfo}
O(S-Design) = {S-ModuleSpec}
O(Coding1)={MainModule}
O(Coding2)={SecurityModule}
O(Integrate)={ObjectCode}

AS(Review)={A}
AS(SecAnalysis)={A, B}
AS(S-Design)= {A, B}
AS(Coding1)={A, C}
AS(Coding2)={B}
AS(Integrate)={C, D, E}
```

Software Process Example

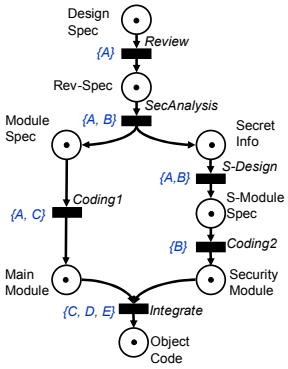


Petri-Net Representation

Example(2/3)

Product Knowledge (leak=0.01)

	A	B	C	D	E
Design Spec	1.00	0.0199	0.01	0.0001	0.0001
Rev-Spec	1.00	1.00	0.01	0.0001	0.0001
Secret Info	1.00	1.00	0.01	0.0001	0.0001
Module Spec	1.00	1.00		0.01	0.01
SModule Spec	1.00	1.00	0.01	0.0001	0.0001
Main Module	1.00		1.00	1.00	1.00
Security Module		1.00	1.00	1.00	1.00
Object Code			1.00	1.00	1.00



Example(3/3)

Product Knowledge (leak=0.01)

	A	B	C	D	E
Design Spec	○	0.0199	0.01	0.0001	0.0001
Rev-Spec	○	○	0.01	0.0001	0.0001
Secret Info	○	○	0.01	0.0001	0.0001
Module Spec	○	○	○	0.01	0.01
SModule Spec	○	○	0.01	0.0001	0.0001
Main Module	○	0.00	○	○	○
Security Module	0.00	○	○	○	○
Object Code	0.00	0.00	○	○	○

The probability reflects the possibility that someone leaked the knowledge of the *product* to the *developer*.

Case Study(1/2)

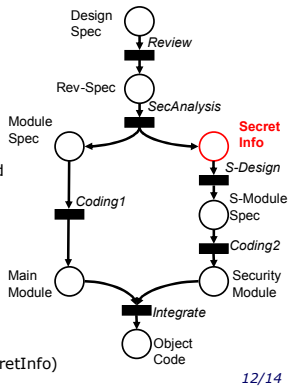
We find an optimal assignment of developers.

Assumption

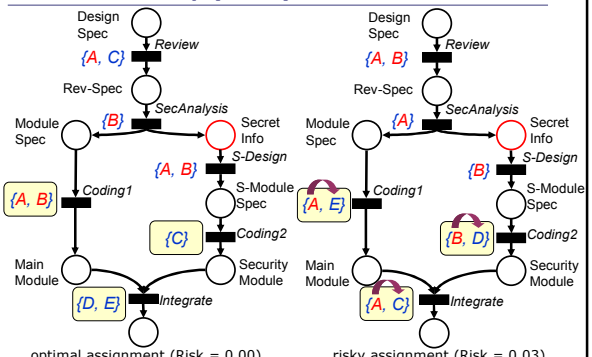
- The processes performed by 5 developers (A, B, C, D, E)
- Only developers A and B are authorized to access *SecretInfo*
- All processes must be performed by the effort of the total 10 developers
- At most 2 developers can conduct each process
- Leak = 0.01

An optimal assignment is an assignment that *Risk* is minimized.

Risk = $\sum_{u \in \{C,D,E\}}$ Pkn(u, integrate, SecretInfo)



Case Study(2/2)



optimal assignment (Risk = 0.00)

risky assignment (Risk = 0.03)

More collaboration among authorized and unauthorized developers causes the higher risk of information leakage.

Conclusion and Future Work

Conclusion

- We have presented a method to evaluate the risk of information leakage in software development process.
  - We formulated the leakage as an unexpected transfer of product knowledge.
  - We proposed a method to derive the probability that each developer knows each work product.

Future Work

- Further evaluation with more practical processes
- Investigation of the emerging application domain