セキュリティエンジニアリングとソフトウェア工学

－現状と課題－

北陸先端科学技術大学院大学　落水浩一郎

1. はじめに

本稿は Ian Sommerville 著 “Software Engineering – 9th ed.”の Part 2 (Chapter10 - Chapter15, pp.261-422, Addison-Wesley, 2011)を要約しつつ、参照論文の要約および筆者なりの解説を加えることにより、セキュアソフトウェア工学の現状と課題についてサーベイを試みるものである。10 章から 15 章の構成は表１（次ページ）に示す通りである。

表１の内容を筆者なりにまとめると、図１のようになる。

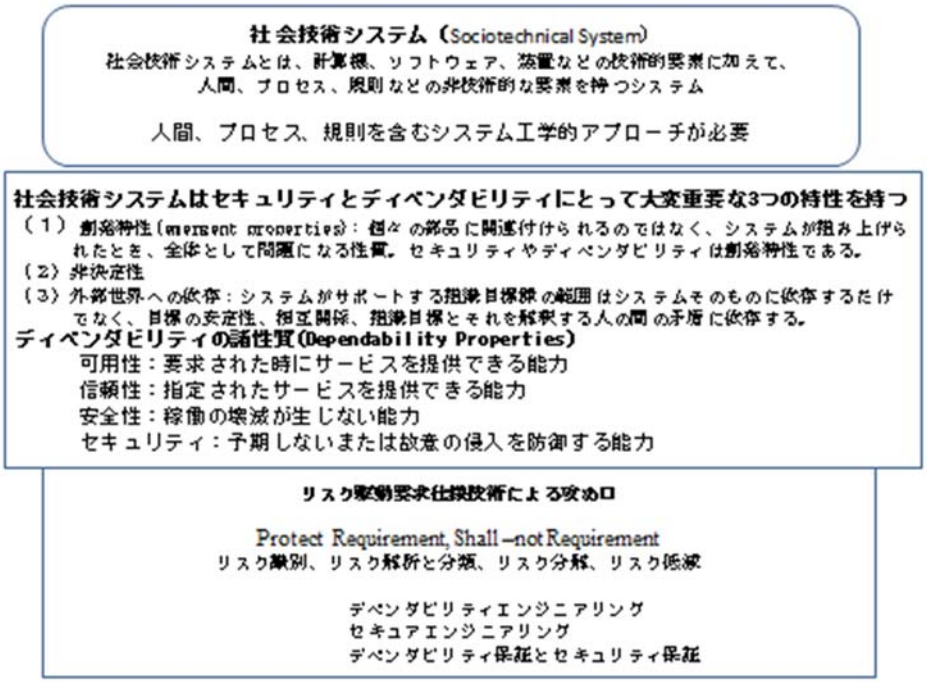


図１　セキュアソフトウェア工学の概要

図１において、

(1) 人間、プロセス、社会規則を含むシステムを社会技術システムと呼ぶ。
(2) 社会技術システムにおいては、創発特性(emergent property)を取り扱うことが重要である。
(3) セキュリティは創発特性の一つである。
(4) 可用性、信頼性、安全性、セキュリティの総称としてディペンダビリティがある。これらは従来ソフトウェア工学の世界では非機能要求として取り扱われてきた。
(5) この課題に対する攻め口としてセキュアエンジニアリングがある。
(6) セキュアエンジニアリングはソフトウェアエンジニアリングにおいて研究されてきたリスク駆動要求仕様技術を適用することが可能である。

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎

2 章以降において、各話題について詳細に考察する。

表 1　Software Engineering – 9th ed.の Chapter10 - Chapter15 の構成

10　社会技術システム(Sociotechnical System)
- 10.1　複雑系システム(Complex System)
- 10.2　システムエンジニアリング(Systems Engineering)
- 10.3　システム獲得(System Procurement)：システムの目的の決定
- 10.4　システム開発(System Development)
- 10.5　システム運用(System Operation)

11　ディペンダビリティとセキュリティ(Dependability and Security)
- 11.1　ディペンダビリティの諸性質(Dependability Properties)
- 11.2　可用性と信頼性(Availability and Reliability)
- 11.3　安全性(Safety)
- 11.4　セキュリティ(Security)

12　ディペンダビリティ仕様とセキュリティ仕様(Dependability and Security specification)
- 12.1　リスク駆動要求仕様(Risk-driven requirement specification)
- 12.2　安全性仕様(Safety specification)
- 12.3　信頼性仕様(Reliability specification)
- 12.4　セキュリティ仕様(Security specification)
- 12.5　形式仕様(Formal specification)

13　ディペンダビリティエンジニアリング(Dependability Engineering)
- 13.1　冗長性と多様性(Redundancy and diversity)
- 13.2　ディペンダブルプロセス(Dependable processes)
- 13.3　ディペンダブルシステムアーキテクチャ(Dependable system architecture)
- 13.4　ディペンダブルプログラミング(Dependable programming)

14　セキュリティエンジニアリング(Security Engineering)
- 14.1　セキュリティリスク管理(Security risk management)
- 14.2　セキュリティのための設計(Design for security)
- 14.3　システム耐破壊性(System survivability)

15　ディペンダビリティ保証とセキュリティ保証(Dependability and Security assurance)
- 15.1　静的解析(Static analysis)
- 15.2　信頼性テスト(Reliable testing)
- 15.3　セキュリティテスト(Security testing)
- 15.4　プロセス保証(Process assurance)
- 15.5　安全性事例とディペンダビリティ事例(Safety and dependability cases)

2. 社会技術システム(Sociotechnical System)

社会技術システムとは、計算機、ソフトウェア、装置などの技術的要素に加えて、人間、プロセス、規則などの非技術的な要素を持つシステムである。そのため、組織のポリシーや手順や構造に影響されるという特徴を持つ。図 2 に社会技術システムの構成を示す。

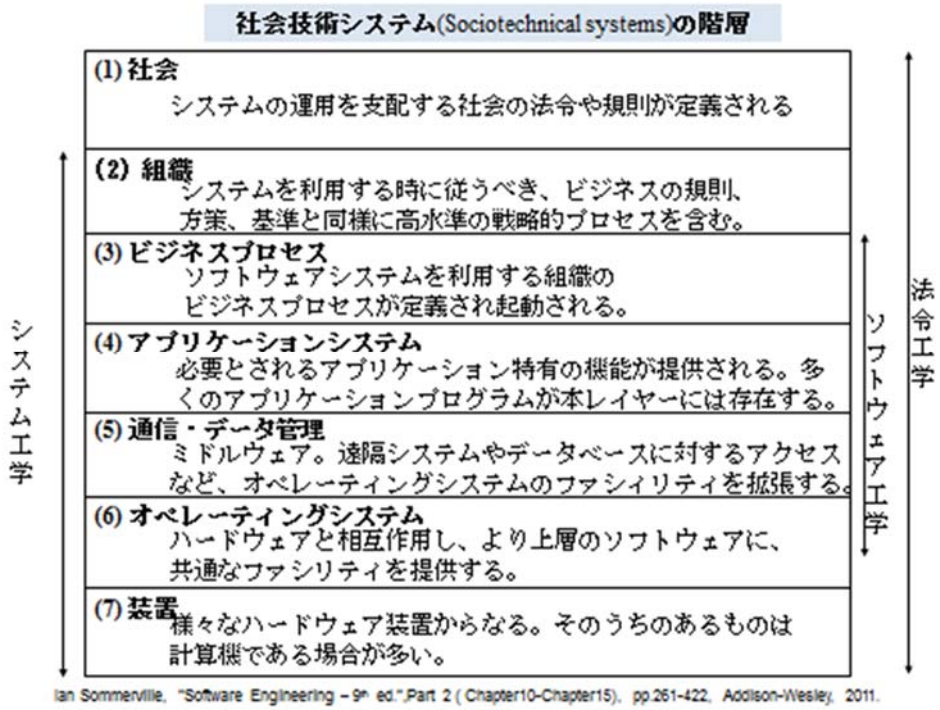


図 2　社会技術システム

図 2 に示すレイヤーにおいて、(3) ビジネスプロセスから (6) オペレーティングシステムまでがソフトウェア工学のカバー範囲であり、(2) 組織から (7) 装置までがシステム工学のカバーする範囲である。なお、片山が開始した法令工学は(1)から(7)すべてをカバーしている。社会技術システムはセキュリティとディペンダビリティにとって大変重要な 3 つの特性を持つ

(1) 創発特性(emergent properties): 個々の部品に関連付けられるのではなく、システムが組み上げられたとき、全体として問題になる性質であり、セキュリティやディペンダビリティは創発特性である。

(2) 非決定性(nondeterministic): 人間の操作員は同じ入力に対して、いつも同じようにふるまうとは限らないので、出力が同じになるとは限らない。

(3) システムがサポートする組織目標群の範囲はシステムそのものに依存するだけでなく、目標の安定性、相互関係、組織目標とそれを解釈する人の間の矛盾に依存する。

システムエンジニアリング(Systems Engineering)は、社会技術システムの獲得、仕様、設

計、実現、検証、配置、運用、保守のすべての活動を対象とする。システム獲得(System Procurement)はシステムエンジニアリングの最初のステップであり、システムのスコープを決定する。決定に必要な情報は、他の組織システムの状況、外部規則の順守、外部との競争、ビジネスの再構成、使用可能予算などである。また、ディペンダビリティとセキュリティの立場からは、システムの変更は問題点と脆弱性の源泉となる。

さらに調査すべき論文リスト

1. R.H. Thayer, “Software system engineering”, IEEE Computer, April 2002.
   要件定義、ソフトウェア設計を Software system engineering、詳細設計・コーディングをソフトウェア工学と位置づけた論文。あまり内容のない論文である。
2. K.Clarke, G.Hardstone, M.Rouncefield, I. Sommerville, “Trust in Technology: A Socio-technical Perspective”, Springer, 2006.
3. “Fundamentals of System Engineering”, in NASA Systems Engineering Handbook, 2007.

3. ディペンダビリティとセキュリティ(Dependability and Security)

3.1 ディペンダビリティの諸性質(Dependability Properties)

ディペンダビリティ(Laprie 1995)は、可用性、信頼性、安全性、セキュリティの総称である。可用性、信頼性、安全性、セキュリティの定義はそれぞれ以下のとおりである。

(1) 可用性(Availability)：要求されたときにサービスを提供できる能力
(2) 信頼性(Reliability)：指定されたサービスを提供できる能力
(3) 安全性(Safety)：壊滅的な稼働が生じない能力
(4) セキュリティ(Security)：予期しないまたは故意の侵入を防御する能力

ディペンダビリティを保証することは以下の点で重要である。

(1) システムの障害(failures)は大変多くの人に影響を与える
(2) 信頼性がなく、安全でなく、セキュアでないシステムを利用者は拒否する
(3) システムの障害(failures)は巨大な費用を費やす
(4) ディペンダブルでないシステムは情報の損失をもたらす。

ディペンダビリティのその他の諸性質としては以下のものがあげられる。

(1) 修復可能性
(2) 保守性
(3) 耐久性(survivability): 外部からの攻撃中もサービスを継続できる能力
(4) 誤り許容性

ディペンダブルなソフトウェアを作るためには以下の事柄が重要である。

(1) システムの仕様化や開発の際に、偶発的な誤りの混入をさけること
(2) システムのディペンダビリティに影響を与えるエラーを発見できるV&Vプロセス

を設計すること

(3) 可用性やセキュリティを保証する、外部からの攻撃を防御できる保護機構を設計すること

(4) 稼働環境上にシステムのその支援ソフトウェアを正しく配置すること

(5) 回復機構を準備すること

3.2 システムの信頼性を改善する技術

Fault Avoidance, Fault detection and removal, Fault torelance

3.3 安全性(Safety)

Safety-Critical System には以下の 2 種類がある。

(1) Primary Safety-Critical System: システムのコントローラに埋め込まれたソフトウェアの不調はハードウェアの不調、人間の傷害を連鎖的に引き起こす

(2) Secondary Safety-Critical System：CAD の不調は製品の不備をもたらす。

システムの信頼性と安全性は相互に関係する。しかし、信頼性のあるシステムが安全であるとは限らない。

3.4 セキュリティ(Security)

セキュリティはディペンダビリティの最も大事な側面であり、ネットワークに結合されたシステムの出現により外部からの攻撃が可能になった。可用性、信頼性、安全性はシステムが初期の意図どおりに設置されていることを仮定している。攻撃されたソフトウェアはその前提をくずしてしまう。このような脅威（セキュリティ上の脅威）には三つのタイプがある。

(1) システムとデータの秘密性(confidentiality)に対する脅威：不当なアクセスに対して情報を見せてしまう

(2) システムとデータの整合性(integrity)に関する脅威：ソフトウェアやデータに損害を与えたり、破壊したりする

(3) システムとデータの可用性に関する脅威：権利のある利用者のアクセスを制限してしまう。

システムのセキュリティを高めるには

(1) 脆弱性の回避：ネットワークに接続しない。暗号化する。

(2) 攻撃の検出と中和：攻撃を検出し追い払う。異常な動作を監視・検出する。

(3) 限度的影響(Exposure limitation)と回復：自動バックアップ、ミラーリング

配列オーバーや不測の入力への無応答などは、セキュリティ逃げ道(loopholes)を作ってしまう。

さらに調査すべき論文リスト

1. R.Cumming, “The evolution of Information assurance”, IEEE Computer, 35(12).
2. W.R, Dunn, “Designing Safety Critical Systems”, IEEE Computer, 36(11).
3. B.Schneier, “Secrets and Lies: Digital Security in a Networked World”, John Wiley&Sons , 2004.

4. ディペンダビリティ仕様とセキュリティ仕様(Dependability and Security specification)

本章では、(1) リスク駆動アプローチが安全性要件、信頼性要件、セキュリティ要件を識別し解析するのに有用であること、複雑系システムに必要とされるセキュリティ要件の異なるタイプについてのみ詳細にサーベイする。

1993 年にポーランドのワルシャワ空港に着陸した飛行機が滑走路をオーバーランし、堤に衝突して炎上する事故が発生した。計算機制御のブレーキングシステムが稼働しなかったことが直接の原因である。ただ、ブレーキングシステム自体は仕様どおりに作成されていたしプログラムに誤りはなかった。真の原因は、仕様がまれなケースを考慮していない不完全なものであった。ブレーキングシステムは航空機の着陸を検知できずまだ飛行中であると認識していた。その結果、飛行機の安全システムが飛行機の逆噴射システムを止めた。この事件は，要件について細心の注意を払うべきことを示唆している。

ディペンダビリティ要件とセキュリティ要件は二つのタイプを持つ。

(1) 機能要求
検査と回復、攻撃に対する防護の機能を定める

(2) 非機能要求
必要な信頼性基準と可用性基準を定める

ディペンダビリティ要件とセキュリティ要件に関する機能要求を生成する出発点は、ビジネスやドメインのルール、ポリシー、規則である。

これらは高レベル要求であり、Shall not Requirements として定義される。

Shall not Requirements の例を以下に列挙する。

1. アクセス権限の変更や生成されてないファイルへのアクセスをすべての利用者に許可しない（セキュリティ）。
2. 飛行中に逆噴射を許さない（安全性）
3. 3 つ以上のアラーム信号の同時起動を許さない（安全性）

4.1 リスク駆動要求仕様(Risk-driven requirement specification)

ディペンダビリティ要件とセキュリティ要件は防護要件(protect requirements)として考え得る。防護要件とは、内部故障から自身を防御する、環境への損害を引き起こすシステム故障を止める、システムに損害を与えるシステムの外部環境からの事故や攻撃を止める、

故障の場合機能を回復させるなどである。

防護要件を発見するには、システムと環境に対するリスクを理解する必要がある。危険な状況を認知し、発生確率と損害にいたる確率を識別し、損害の範囲を特定する必要がある。

リスク駆動仕様化は安全でセキュアなシステムの開発者によって広く利用されてきた保手段である。リスク駆動仕様化は最も多く損害を引き起こすか、頻発するであろう事象に焦点を当てる。安全クリティカルなシステムでは、リスクは事故にいたるハザードと結びつけられる。セキュリティクリティカルなシステムでは、リスクは、可能な脆弱性を利用しようとする内部や外部からの攻撃から生じる。

リスク駆動仕様の手順は以下のとおり

(1) リスク識別

システムに対する潜在的なリスクが識別される。これらはシステムが利用される環境に依存する。リスクは環境との相互作用や、運用環境におけるまれな条件から発生する。ワルシャワ空港における事故は、雷雨による横風が飛行機を傾け、その結果、一つの車輪で着陸したことに起因するものである。

(2) リスク解析と分類

各々のリスクは別々に検討される。潜在的に重大なリスクと not implausible がさらなる解析によって選択される。

(3) リスク分解

それぞれのリスクは、リスクの潜在的根本原因を発見するために解析される。根本原因とはシステムがフェーイルする理由である。それらは、ソフトウェアやハードウェアの誤り、システムの設計上の意思決定から生じる固有の脆弱性である。

(4) リスク低減

識別されたリスクを低減させ消滅させるための手段を提案する。

大規模システムの場合は、以下のフェーズに分けるとよい。

(1) 初期リスク解析

システム環境から発生する主要なリスクが識別される。システム開発に利用される技術とは独立している。初期リスク解析の目的は、セキュリティ要件とディペンダビリティ要件の初期集合を開発することである。

(2) ライフサイクルリスク解析

システム開発時に発生し、システムの設計上の意思決定に関連して発生するリスクである。異なる技術やシステムアーキテクチャはそれぞれ結合されるリスクを持つ。この時点でセキュリティ要件をリスクを防御するものに拡張しなければならない。

(3) 運用リスク解析

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎

システムのユーザインタフェースとオペレータの操作ミスに関連するリスクである。ユーザインターフェスに関する決定がなされた場合、それに対する防護要件が追加されるべきであろう。

セキュリティ要件とディペンダビリティ要件は、技術の選択と設計上の決定に特に影響される。

4.2　安全性仕様(Safety specification)

安全性仕様は防護要件であり、正常なシステム動作には関係しない。ハザードという用語を使用する。手順は以下の通り。

(1) リスク識別：ハザードの識別
(2) リスク解析：ハザードの評価
(3) リスク分解：ハザードを引き起こす事象の発見
(4) リスク低減：ハザード解析の結果を踏まえ、安全性要件にいたる。

故障木の手法が使える

4.3　信頼性仕様(Reliability specification)

省略する。

4.4　セキュリティ仕様(Security specification)

セキュリティ仕様は安全性仕様とある意味で共通する。それらを定量的に仕様化することは非現実的であり、セキュリティ要件は、必要とされるシステムの機能よりもむしろ、受理しがたいシステムの振る舞いを定義する Shall not requirements である。しかしながら、セキュリティは安全性よりも以下の理由によりもっと挑戦的である。

(1) 安全性を考察するときは、システムが設置された環境が敵対的でないと仮定できる。
(2) 安全性に対するリスクを引き起こす故障が発生した時、故障を引き起こした誤りを探すことになる。攻撃がシステム故障を引き起こし場合、根本原因を発見することはより困難になる。
(3) 安全性に関係する故障が起きたときは、システムをシャットダウンしたり、サービスの度合いを下げることで対応することは受け入れ可能である。しかし、サービス否定と呼ばれる攻撃は、システムをシャットダウンすることが目的である。
(4) 安全性に関する事象は知的な敵対者によって生成されない。攻撃者は一連の攻撃中に、システムの応答を通じて、攻撃を修正しつつ、システムの防御を探知する。

これらの違いは、セキュリティ仕様は安全性仕様よりより拡張的であることを意味する。10 個のタイプのセキュリティ要件がある。

(1) 識別(Identification)要件：利用者と相互作用する前に、ユーザを識別する必要があるかどうかを仕様化する。

(2) 認証(authentication)要件：どのように利用者が識別されるかを仕様化する。
(3) 権利(Authorization)要件:識別された利用者に対する特権とアクセス権を仕様化する。
(4) 免疫(Immunity)要件：ウイルス、ワーム、その他同様の脅威からシステム自体をどのように防護するのかを仕様化する。
(5) 整合性(integrity)要件: データ破壊をどうやって回避するのかを仕様化する。
(6) 侵入(intrusion)要件:システムに対する攻撃を検出するために利用される機構を仕様化する。
(7) 非拒否(Non-repudiation)要件:トランザクションの一部分がそのトランザクションに含まれることを否定できないことを仕様化する。
(8) プライバシー(privacy)要件：どのようにしてデータのプライバシーが維持されるべきかを仕様化する。
(9) セキュリティ監査(security auditing)要件：システムの利用がどのように監査され検査されるのかを仕様化する。
(10) システム保守(system maintenance)セキュリティ要件：アプリケーションが、セキュリティメカニズムの偶発的な破壊から権利あるアクセスをどうやって妨害できるかを仕様化する

リスク駆動セキュリティ要件プロセスを図 3 に示す。

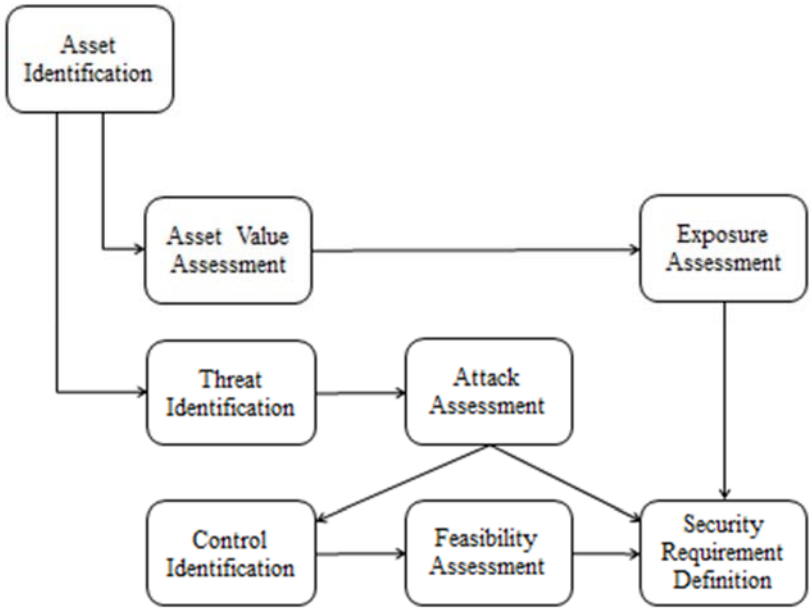


図 3　リスク駆動セキュリティ要件仕様化プロセス

図 3 において各フェーズのタスクは以下の通りである。

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎

(1) **Asset 識別**:防御の必要な財産を識別する。システムに結合されたデータと同様に、システム自体や特別なシステム機能も財産として識別する（リスク識別)。

(2) **Asset 価値評価**：識別された財産の価値を評価する（リスク解析)。

(3) **Exposure 評価**：財産の紛失に伴う損失を評価する。直接的な損失のコストに加えて、回復コストや風評コストも勘定にいれる必要がある（リスク解析)。

(4) **脅威識別**：財産にたいする攻撃を識別する（リスク解析)。

(5) **攻撃評価**：脅威を攻撃と攻撃の手段に分解する（リスク分解)。故障木が利用可能である。

(6) **制御識別**:財産を防御するための制御方法(暗号化など)を提案する(リスク低減)。

(7) **実現性評価**：防御手段の実現可能性とコストを評価する（リスク低減)。

(8) **セキュリティ要件定義**：(3)、(4)、(6）に基づき、基盤とアプリケーションに対するセキュリティ要件を定義する（リスク低減)。

4.5 形式仕様(Formal specification)

Plan-based ソフトウェアプロセスでは形式仕様技術が利用できる。

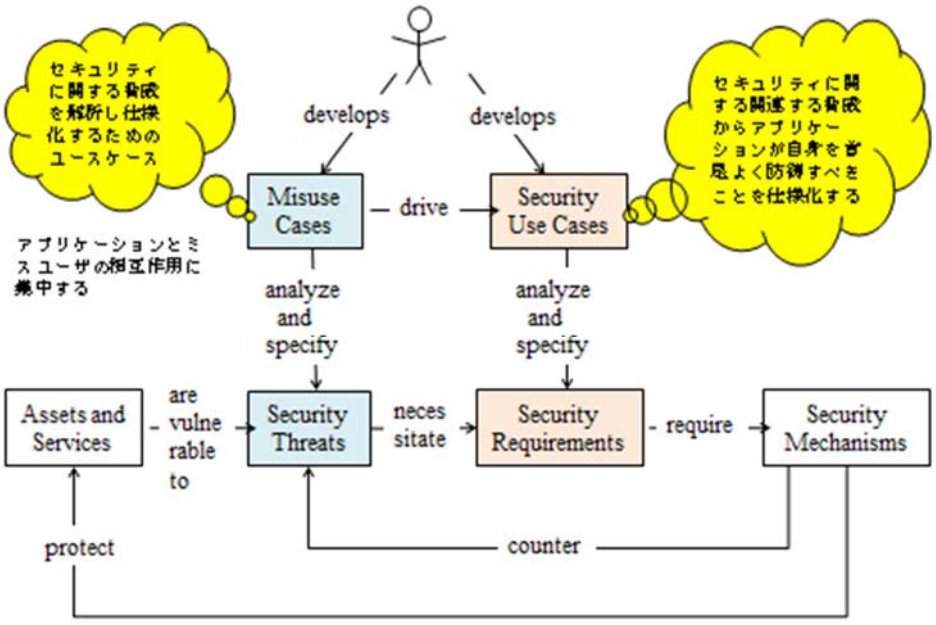
B メソッドの利用。

さらに調査すべき論文リスト

1. D.G Firesmith, “Security Use Cases”, Journal of Object Technologies, 2(3), 2003.

セキュリティ要件とミスユースケースの違いと役割を指摘した論文である。

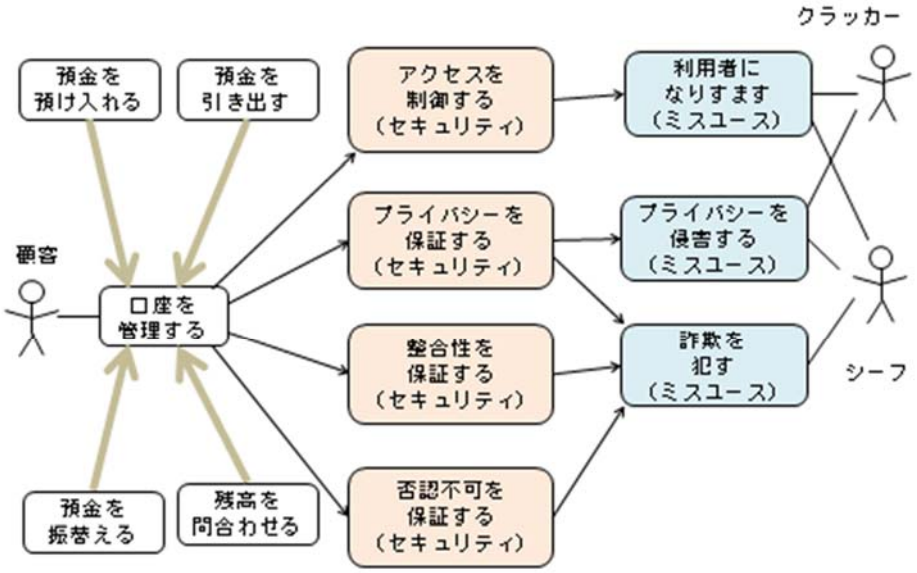
せクリティ要件とミスユースケースの役割の違いを下図に示す。



ミスユースケースとセキュリティユースケースの関係

ATM を例にした、ユースケース定義の例を下図に示す。

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎



ミスユースケースとセキュリティユースケースの例

論文ではさらに、アクセス制御、整合性保証、プライバシー防御に関するセキュリティユースケースの一般形を示している。

2. J.P. Bowen and M.G. Hinchey, “Ten Commandments of Formal Methods…Ten Years Later”, Trust in Technology: A Socio-technical Perspective”, IEEE Computer, 39(1), Jan. 2006.
3. I.A. Tondel, M.G. Jaatun, P.H. Meland, “Security Requirements for the Rest of Us”, IEEE Software 25(1), Jan.and Feb. 2008.

セキュリティ要件エンジニアリングに対する異なる研究が
要件フェーズのタスクをどのように取り扱っているか

研究	定義	目標	ミスユース/脅威	財産	コーディング標準	分類と優先付	検査と承認	プロセス計画
SQUARE	✔	✔	✔			✔	✔	
Charles Haley & colleagues		✔	✔	✔			✔	
Gustav Bostrom & colleagues			✔	✔	✔	✔		
CLASP		✔	*	✔			✔	
Microsoft		✔	✔†	✔†				✔
Axelle Apvrille & Makan Pourzandi		✔	✔			✔		
Eduardo Fernandez			✔					
Kenneth van Wyk & Gary McGraw			✔					
Gunnar Peterson			✔					

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎

セキュリティ要件の獲得(elicitation)技術の開発が重要であるという現状認識のもとに、まず関連研究をサーベイした（上図)。これらのサーベイの結果として、セキュリティ要件の関する統一的な定義がないとの結論にいたった。脅威、財産、対象物の識別に焦点を当てた以下のようなアプローチを定義した。

(1) セキュリティ対象物(security objective)

利用者にとって最も重要なハイレベルの要求またはゴール、または、関連する法律、ポリシー、標準などを順守するために満たされるべき要求。以下の 2 ステップで実行する。①セキュリティに関する利用者のニーズを識別する。②システムやモジュールに適用される関連する法律、ポリシー、標準、ベスト・プラクティスを識別する。

(2) 財産の識別

セキュリティ要件を獲得し、優先順位付けをおこなうには、開発者は、「どの財産のどの性質が防御すべきものとして最も重要であるか」を知る必要がある。

(3) 脅威解析

すでに定義されている脅威の分類 STRIDE(spoofing　なりすまし, tampering 改竄, repudiation 否認, information disclosure　情報開示（不正読み込み）, denial of service　サービス拒否（DOS 攻撃）, elevation of privilege 特権の昇格)を利用して、最も重要な財産に焦点を当てる。攻撃木の手法を使って最も重要な脅威を分析する。

(4) セキュリティ要件の文書化

どうやって達成するのかではなく、何が達成されるべきかに焦点を当てて要件を記述する

4. 情報システムに係る政府調達におけるセキュリティ要件策定マニュアル

日本政府においても、2011 年 3 月 30 日に「情報セキュリティを企画・設計段階から確保するための方策に係る検討会（内閣官房情報セキュリティセンター)」において、政府機関の情報システムにおいて情報セキュリティ対策を適切に講じるために、情報システムのライフサイクル（企画・設計、開発・運用・廃棄）における企画段階（調達段階）から情報セキュリティの観点を意識し、その際に必要となる調達仕様にセキュリティ要件を適切に組み込むためのマニュアルが作成されている。

(1) 政府調達における利用のタイミング

情報システムの調達プロセスにおける発注側の調達担当者が、調達仕様書にセキュリティ要件を記載するための手順を定め、作業を支援する。

(2) 調達指針が定める調達仕様書に記載する事項

本マニュアルが策定の対象とするセキュリティ要件の記載箇所は、主に表中赤字の部分である。また、セキュリティ要件の策定過程で明らかになる業務

要件には、セキュリティ要件を満たすセキュリティ機能を提案する際に不可欠となる情報となる可能性が高いものが含まれるため、そのような情報は「1 調達件名」、「2　作業の概要」、および「7　情報システム稼働環境」に記載する。

項目		主な記載内容
1	調達件名	情報システムに係る工程名
2	作業の概要	(1) 目的、(2) 用語の定義、(3) 業務の概要、(4) 情報システム化の範囲、(5) 作業内容・納入成果物
3	情報システムの要件	(1) 機能要件、(2) 画面要件、(3) 帳票要件、(4) 情報・データ要件、(5) 外部インタフェース要件
4	規模・性能要件	(1) 規模要件、(2) 性能要件
5	信頼性等要件	(1) 信頼性要件、(2) 拡張性要件、(3) 上位互換性要件、(4) システム中立性要件、(5) 事業継続性要件
6	情報セキュリティ要件	(1) 権限要件、(2) 情報セキュリティ対策
7	情報システム稼働環境	(1) 全体構成、(2) ハードウェア構成、(3) ソフトウェア構成、(4) ネットワーク構成、(5) アクセシビリティ要件
8	テスト要件定義	要求仕様の適合性を検証するためのテストに係る要件
9	移行要件定義	(1) 移行に係る要件、(2) 教育に係る要件
10	運用要件定義	(1) システム操作・監視等要件、(2) データ管理要件、(3) 運用施設・設備要件
11	保守要件定義	(1) ソフトウェア保守要件、(2) ハードウェア保守要件
12	作業の体制および方法	(1) 作業体制、(2) 開発方法、(3) 導入、(4) 瑕疵担保責任
13	特記事項	その他、特記すべき要件
14	妥当性証明	調達仕様書の妥当性を確認した調達担当課室の長の氏名

※下線部分はセキュリティ要件又はその関連情報の記載が想定される箇所

(3) 手順の全体像

①. 業務要件の検討

調達担当者は、情報システムを調達する「目的」および対象とする「業務」を洗い出した上で、「主体」、「情報」、「利用環境・手段」の 3 つの観点（誰が何をどのように）を意識して業務の特徴を整理することによって、情報

システムに係る「業務要件」を抽出する。

主体	業務	業務(細分化後)	業務（細分化後）の概要	情報	利用環境・手段
国民	政策に関する意見・コメントの投稿	利用登録	個人情報を登録して、サービスの利用資格を得る。	「個人情報」（氏名、ニックネーム、性別、年齢、職種、連絡先等）	PC，携帯電話、スマートフォン、インターネット
		意見・コメントの投稿、投票	新規の意見や他者の意見に対するコメントの投稿および投票を行う。	「意見」「コメント」（タイトル、本文、投稿者名、投稿日時等）	
		意見・コメントの閲覧	意見やコメントを検索し、閲覧する。	「意見」「コメント」（タイトル、本文、投稿者名、投稿日時等）	
事務局	政策に関する情報提供および利用者の管理	意見に対する回答、修正	意見に対する事務局回答の投稿及び不適切な意見の削除を行う	「回答」（本文、投稿者名、投稿日時等）	PC，内部ネットワーク
		事務局からの周知	サービス停止や注意事項等の利用者に対する周知を行う。	「お知らせ文面」	
		利用者の管理	利用者の登録情報の確認、修正、等の管理業務を行う。	「個人情報」（氏名、ニックネーム、性別、年齢、職種、連絡先等）	
システム管理者	情報システムの利用状況の把握および管理	利用状況の把握	利用者の登録状況、アクセス状況、意見やコメントの集計を行う。	「利用統計」（全体および意見ごとのアクセス数、利用者の登録数等）	PC，管理用LAN
		不正利用および障害の監視、追跡	アクセス状況の監視およびログ等を元にした原因究明を行う。	「履歴」（アクセス、認証、利用ログ等）	
		システムのバックアップと復旧	システムのデータを定期バックアップおよび障害時の復旧を行う。	「システムデータ」	

抽出した業務要件を「システム概要図」と呼ぶ図に表して（見える化）

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎

俯瞰することにより業務要件に不足や矛盾点がないことを確かめるとともに、「定型設問」に回答することによって業務要件の詳細化を図る。

②. セキュリティ要件の策定

調達担当者は①にて検討した業務要件を踏まえ、「対策要件集」から調達する情報システムにふさわしいセキュリティ要件を選定して、「調達仕様書」に記載する。対策要件集とは、情報システムの標準的なセキュリティ要件をまとめたものである。下表に対策要件集を示す。

対策区分	対策方針	対策要件	判断条件 対応関係	実施レベル有無		
				低位	中位	高位
侵害対策 (AT:Attack)	通信回線対策(AT-1)	通信経路の分離(AT-1-1)	A or F		有	有
		不正通信の遮断(AT-1-2)	A		有	
		通信のなりすまし防止(AT-1-3)			有	有
		サービス不能化の防止(AT-1-4)			有	有
	不正プログラム対策(AT-2)	マルウェアの感染防止(AT-2-1)	-	有		
		マルウェア対策の管理(AT-2-2)	A or B			有
	セキュリティホール対策(AT-3)	構築時の脆弱性対策(AT-3-1)	-	有		
		運用時の脆弱性対策(AT-3-2)	A	有	有	
不正監視・追跡(AU: Audit)	証跡管理(AU-1)	証跡の蓄積・管理(AU-1-1)	B or C	有	有	
		証跡の保護(AU-1-2)		有	有	有
		時刻の正確性確保(AU-1-3)	-	有		
	不正監視(AU-2)	侵入検知(AU-2-1)	A		有	有
		サービス不能化の検知(AU-2-2)				有
アクセス・利用制限 (AC:Access)	主体認証(AC-1)	主体認証(AC-1-1)	D		有	有
	アカウント管理(AC-2)	ライフサイクル管理(AC-2-1)	D		有	
		アクセス権管理(AC-2-2)	E			有
		管理者権限の保護(AC-2-3)	-	有		
データ保護 (PR: Protect)	機密性・完全性の確保(PR-1)	通信経路上の盗聴防止(PR-1-1)	B or C		有	
		保存情報の機密性確保(PR-1-2)			有	有
		保存情報の完全性確保(PR-1-3)				有
物理対策 (PH:Physical)	情報搾取・侵入対策(PH-1)	情報の物理的保護(PH-1-1)	-	有		
		侵入の物理的対策(PH-1-2)		有		
障害対策(事業継続対応) (DA:Damage)	構成管理（DA-1）	システムの構成管理(DA-1-1)	B	有	有	
	可用性確保(DA-2)	システムの可用性確保(DA-2-1)	-	有		

ここで、A は外部アクセスの有無、B は情報の重要度、 C は情報保存時の安全性、 D は利用者の限定要否、 E はアカウントの多様性、 F は複数部局による利用である。

また、低位の実施レベルの仕様書記載例の採用を検討する場合は、調達コスト等を勘案し検討する。中位また高位の実施レベルの仕様書記載例の採用を検討する場合は、中位と高位のどちらを採用すべきかを検討する。費用対効果の観点からは中位レベルの採用が望ましい。

最後に調達担当者は、仕様記載例を決定した後は、対策要件集の「仕様書記載時の注意事項」の解説および以下の点に留意して、調達仕様書の該当部分に記載する。

- 記載内容のさらなる具体化
- 対策要件の記載箇所
- 既存設備の利用を想定した仕様の調整
- 記載内容の妥当性の点検

調達指針の記載例

大項目		小項目	記載内容
1	調達件名	情報システムに係る工程名	(※　ステップ 1 の検討結果のうち「名称」を反映)
2	作業の概要	(1) 目的	(※　ステップ 1 の検討結果のうち「目的」を反映)
		(2) 用語の定義	
		(3) 業務の概要	(※　ステップ 2 およびステップ 4 の結果を反映)
		(4) 情報システム化の範囲	
		(5) 作業内容・納入成果物	
3	情報システムの要件	(1) 機能要件	
		(2) 画面要件	
		(3) 帳票要件	
		(4) 情報・データ要件	
		(5) 外部インタフェース要件	
4	規模・性能要件	(1) 規模要件	
		(2) 性能要件	
5	信頼性等要件	(1) 信頼性要件	(※　対策要件 DA-2-1 を求める場合に記入)
		(2) 拡張性要件	
		(3) 上位互換性要件	
		(4) システム中立性要件	
		(5) 事業継続性要件	
6	情報セキュリティ要件	(1) 権限要件	(※　対策要件 AT-1, AT-2, AU-1, AU-2, AC-1, AC-2 , PR-1, DA-1-1 を求める場合に記入)
		(2) 情報セキュリティ対策	

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎

7	情報システム稼働環境	(1) 全体構成	(※　ステップ 3 にて作成したシステム概要図を記載)
		(2) ハードウェア構成	
		(3) ソフトウェア構成	
		(4) ネットワーク構成	
		(5) アクセシビリティ要件	
8	テスト要件定義	要求仕様の適合性を検証するためのテストに係る要件	(※　対策要件 AT-3-1 を求める場合に記入)
9	移行要件定義	(1) 移行に係る要件	
		(2) 教育に係る要件	
10	運用要件定義	(1) システム操作・監視等要件	
		(2) データ管理要件	
		(3) 運用施設・設備要件	(※　対策要件 PH-1 を求める場合に記入)
11	保守要件定義	(1) ソフトウェア保守要件	(※　対策要件 AT-3-2 を求める場合に記入)
		(2) ハードウェア保守要件	
12	作業の体制および方法	(1) 作業体制	
		(2) 開発方法	
		(3) 導入	
		(4) 瑕疵担保責任	
13	特記事項	その他、特記すべき要件	
14	妥当性証明	調達仕様書の妥当性を確認した調達担当者課室の長の氏名	

5.　リスク定量化手法の概要（岡本卓馬「情報セキュリティにおけるリスクの定量化手法」, UNISYS TECHNOLOGY　REVIEW　第86号,　Aug.　2005より引用）

本稿ではリスクを定量化するための手法として，Annual Loss Expectation（以下ALE）によるリスク定量化の手法を採りあげる．ALE とは，何らかのセキュリティ事故によって，企業が1 年間に被ると予測される損失額のことである．この手法では，まず自社の現状に基づいて識別された個別のリスクに対して，以下の計算式で年間の予想損失額を算出する．

ALE（年間予想損失額）＝年間発生頻度×予想損失額

予想損失額とは，個別のリスクが1 回顕在化するごとに被る被害額の予測値である．また，年間発生率は，リスクが1 年間に顕在化すると予測される回数を指す．リスクが顕在化する確率及び予想される損失額は，実施する情報セキュリティ対策によって変化する．ALE を活用して情報セキュリティ対策の投資対効果を計るためには，修正ALE という数値を算出する必要がある．修正ALE とは，対策実施によるリスク顕在化の確率及び予想損失額の変化を反映させて，ALE を算出し直した数値である．ALE と修正ALE を比較することで，特定の情報セキュリティ対策の実施によってどの程度の年間予想損失額の低減が見込めるのかを明確にすることが可能となる．

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎

（1）リスク定量化の算出方法例

(1.1) リスク定量化のステップ

リスクの定量化をおこなうためにはまず，リスクアセスメントを実施し，自社が保有しているリスクを明確にすることが不可欠である．ALE によるリスク定量化では，洗い出した個別のリスクに対して，リスク顕在化の確率，及びリスクが顕在化した場合に被ると予想される予想損失額を見積もり，その乗算から年間予想損失額を算出する．次節以降で，リスク顕在化の確率及び予想損失額の算出方法例について記述する．

(1.2) リスク顕在化の確率の算出

リスクが顕在化する確率を正確に算定することは困難であるが，統計的データや経験的データからある程度の数値を算出することは可能である．具体的には，自社における過去の実績データ（システム障害や社内犯罪など），システム利用者の状況（情報リテラシー，利用者数，利用頻度など），また，公的機関が発表している情報などを利用する．

リスク顕在化の確率を算出する上で使用するデータは，想定するセキュリティ事故によって異なってくる． 例えば，現在，社会の関心が最も高いと言える個人情報の漏洩について考えてみると，一般的に個人情報の漏洩は個人情報データベースに対して正規のアカウントを持つユーザによる漏洩と，不正アクセスによる漏洩との2 種類に分けられる．

正規のアカウントを持つユーザによる個人情報の漏洩については，情報セキュリティに関する犯罪発生率を表す適切なデータが現在のところ存在しない．そのため，犯罪白書といった公的データを参考に推察する必要があるであろう．例えば，法務省の「犯罪白書平成15 年版」[5] によると，日本における窃盗の発生率は，10 万人当たり1437 件であり，1 年あたりの発生確率は1.4%である．あくまでデータからの推察であるが，利用者が100 人いれば，年間およそ1 回の割合で個人情報が漏洩してしまう計算になる．

正規のアカウントを持たない者の不正アクセスによって個人情報が漏洩してしまう確率を算出する際には，現在公表されているデータが参考になる．例えば，独立行政法人情報処理推進機構（IPA）のセキュリティセンターが毎月公表している不正アクセス届出状況が挙げられる．これによると，2005 年上半期（1 月～6 月）では，不正アクセスの届出件数は319 件であり，その内実際に侵入やメール不正中継などの被害に遭ったケースは89 件に及び，前年比約2.5倍の増加となっている．届出は任意のため，直接このデータから個人情報が漏洩する確率を計算するのは難しいが，自社の状況に照らし合わせて大まかな傾向をつかむことは可能である．また，最近では，「1 ： 29 ： 300 の法則」を使用して，情報漏洩の発生確率を推定する方法も挙げられている．「1 ： 29 ： 300 の法則」は米国のハインリッヒ氏が労働災害の事例の統計を分析した結果導き出されたもので，「ハインリッヒの法則」とも呼ばれている．これによれば，1件の重大災害の裏には29 件のかすり傷程度の軽災害があり，その裏にはケガはないがヒヤリとした300 件の体験があるとしている．つまり，不正アクセスに当てはめると，330 件の悪意ある攻撃があるとすると，そのうちの30 件が事故につながり，その中の1 件は重大な事故であるという試算である．不正アクセス検知を行っている企業であれば，この法則を参考に，情報漏洩事故の発生確

率を概算することは可能であると言える．このように，想定するセキュリティ事故の発生原因を考慮して，それに適合するデータや分析手法を利用することで，リスク顕在化の確率を見積もることが可能である．

(1.3) 予想損失額の算出

予想損失額を算出するためには，個別のリスクが顕在化した際に被ると予想される損失の構成要素を洗い出す必要がある．損失には，復旧などに要した人件費や業務停止による逸失利益，損害賠償費用など様々な要素が考えられるが，本稿ではこれらの構成要素を大きく二つに分類する．まず一つ目として，復旧に要するコスト，事故対応に要するコスト，システム停止による逸失利益を，リスクが顕在化することで生じる1 次的な損失額である「直接損失額」とする．次に二つ目として，各種の補償や損害賠償請求費用などの2 次的な損失額を「間接損失額」とする．実際にセキュリティ事故が発生してしまった場合，一般的に，企業には再発防止策を実施することが求められる．この際，外部にコンサルティングを依頼するコストや対策実施にかかるコストなどが発生すると考えられるが，ALE によるリスク定量化は，あくまで対策を実施することによって，どの程度年間予想損失額を軽減できるかを計るための手法である．そのため，再発防止対策コストについては予想損失額を算出する上では対象外とする．直接損失額は，以下のような式で算出することができる．

直接損失額＝逸失利益＋復旧に要するコスト＋事故対応に要するコスト

＝（年間想定売上高÷365×業務停止日数）＋（復旧対応をおこなう人数×原価×復旧にかかる日数＋ハードウェア・ソフトウェア費用）＋（事故対応をおこなう人数×原価×事故対応にかける日数）

復旧にかかる日数については，想定するリスクを，自社の過去の事例やシステム構成などと照らし合わせて見積もる必要がある．また，暫定的対応としてハードウェア・ソフトウェアが必要となる可能性にも留意すべきである．事故対応については，例えば，個人情報が漏洩した際の顧客からの問い合わせ窓口設置や，取引先への営業的対応のそれぞれに必要とされる人数及び日数を見積もる必要がある．間接損失額を算出するためには，2 次的に発生すると想定されるコストを洗い出す必要がある．2 次的なコストの例としては，補償，損害賠償費用，謝罪広告費用などがある．間接損失額については以下のような式で算出できる．

間接損失額＝補償・補填費用＋損害賠償費用＋謝罪広告費用

補償，補填費用の具体例としては，個人情報漏洩事故事例に見られるような，お詫び金が挙げられる．昨今の個人情報漏洩事故事例では，個人情報が漏洩した企業がその被害者に対して，お詫び金として500 円～1000 円の金券を配布するケースが見られる．このような事例を参考にして費用を見積もることが有効である．損害賠償費用の見積もりについても個人情報漏洩事故事例から考える．個人情報漏洩事故から損害賠償請求訴訟に発展した場合，企業が被る損失は莫大なものになると予想される．想定損害賠償額を見積もる上でよく参考にされるのは，京都府宇治市の住民情報漏洩事件である．この事件では，「基本4 情報（氏名，住所，性別，生年月日）を漏洩した場合，慰謝料は1 人につき1 万円とする」という判例が出ている．漏洩し

た個人情報の内容によって想定される損害賠償額は異なってくるため，判例から想定損害賠償額を見積もるには，自社の保有する個人情報の内容と合致する個人情報漏洩事故の判例を参考にする必要がある．想定損害賠償額を見積もる手法としては日本ネットワークセキュリティ協会から「2004 年度セキュリティインシデントに関する調査報告書ver 1.0」[6] が公開されている．この報告書では，個人情報漏洩事故を漏洩個人情報価値，情報漏洩元組織の社会的責任度，事後対応評価の三つの要素で分析してレベル分けし，想定損害賠償額をこの3 要素の乗算で求めている．また，顕在化した場合の影響が甚大であると推測されるリスクについては，謝罪広告費用についても考慮する必要がある．

このようにして算出した直接損失額と間接損失額の和が予想損失額となる．

予想損失額＝直接損失額＋間接損失額

(1.4) ALE の算出と修正ALE

算出したリスク顕在化の確率と予想損失額から，現状での年間予想損失額（ALE）を算出する．ALE は1 年間にリスクが顕在化する頻度と，リスクが顕在化した場合に被ると予想される予想損失額の乗算で求められる．当然のことながら，リスクが顕在化する確率及び予想される損失額は実施する情報セキュリティ対策によって変化する．ALE を活用して，情報セキュリティ対策の投資対効果を計るためには，対策実施による変化を反映させて，修正ALE を算出する必要がある．

修正ALE の算出には，実施する情報セキュリティ対策を，リスクが顕在化する確率を低減させる対策，リスクが顕在化した際の被害を低減させる対策，両方の効果が見込める対策のように分類することが有効である．例えばリスクアセスメントの結果からリスクを分析したところ，リムーバブルメディアを利用した内部からの個人情報漏洩というリスクの予想損失額が5000 万円，1 年間にリスクが顕在化する頻度が0.5，つまり2 年に1 回であるすると，ALE は以下のように計算できる．

ALE＝5000 万円×0.5＝2500 万円

このリスクに対して，暗号化などによるデータ保護ソフトを導入した場合，リスク顕在化の頻度を0.1，つまり10 年に1 回まで低減させることができるとすると，修正ALE は，

修正ALE＝5000 万円×0.1＝500 万円

となる．この場合，データ保護ソフトの導入に必要な費用が1000 万円だと仮定すると，実際には，可用性や運用方針なども考慮に入れる必要があるが，投資対効果の面では導入を検討する余地は十分にある．

このように，年間予想損失額としてリスクを定量化することは，情報セキュリティ対策への投資対効果を計る上で有効である．しかし，リスクアセスメントが的確におこなわれていなければ，情報資産の脅威，脆弱性からリスク顕在化の確率や予想損失額を信頼できるデータとして算出することは難しい．リスクを定量化するためには，リスクアセスメントは必要不可欠なのである．

（1.5）リスク定量化の留意点

年間予想損失額としてリスクを定量化することは，情報セキュリティ対策への投資額を考える上で有効である．しかし，リスクアセスメントの結果洗い出した全てのリスクに対して定量化を実施することは，必要となる人的コストから考えても現実的ではない．リスクを定量化する目的は，経営者が情報セキュリティ対策の投資対効果を計るための指標を提供することである．そのため，定量化を実施するリスクの対象は，リスクアセスメントの結果を踏まえ，自社に重大な損失を与える可能性があるリスクに絞るべきである．

また，一般的に投資対効果を計るための指標を算出する際には，より精密な数値を算出しようとする傾向が強い．しかしながら，情報セキュリティにおけるリスクは日々刻々と変化している．精密な数値を算出するために時間を費やしている間にリスクが変化してしまう可能性もある．予想損失額が1000 万円であるか，1100 万円であるかというデータはあまり重要ではない．重要なのは精密な数値ではなく，自社の環境に即した正確な数値を経営者に提供し，より迅速な意思決定を可能とすることなのである．

5. ディペンダビリティエンジニアリング(Dependability Engineering)

5.1　冗長性と多様性(Redundancy and diversity)

5.2 ディペンダブルプロセス(Dependable processes)

5.3 ディペンダブルシステムアーキテクチャ(Dependable system architecture)

5.4 ディペンダブルプログラミング(Dependable programming)

6. セキュリティエンジニアリング(Security Engineering)

システム・セキュリティ・エンジニアリング技術の開発が重要である。セキュリティに関する問題を論ずる場合、アプリケーションセキュリティと基盤セキュリティの双方を考える必要がある。基盤構造に含まれるものは以下の通りである。

1. Linux や Windows のようなオペレーティングシステム
2. Web Browsers や e-mail clients のような、汎用のアプリケーション
3. データベース管理システム
4. 分散計算やデータベースアクセスを支えるミドルウェア
5. アプリケーションソフトウェアで利用される再利用コンポーネントのライブラリ

アプリケーションセキュリティと基盤セキュリティの主要な違いは以下の通りである。

(1) アプリケーションセキュリティはソフトウェア工学の問題であり、ソフトウェア技術者はシステムが攻撃に抵抗できるように設計されていることを保証しなければならない。

(2) 基盤セキュリティは管理の問題であり、システム管理者は攻撃に抵抗できるように基盤を配置する必要がある。

システムセキュリティ管理は単一のタスクではなく、利用者とアクセス許可の管理、システムソフトウェア配置と保守、攻撃モニタリング、攻撃検知と回復などの活動を含んでい

る。

(1) 利用者と許可管理
適切な利用者の認証(authentication)管理がきちんとしていること、必要とする資源にのみ利用者がアクセスできるようにアクセス許可が設定されていることを前提に、システムに利用者を登録し、除去することである。

(2) システムソフトウェア配置と保守
システムソフトウェアとミドルウェアをインストールすること、セキュリティ脆弱性が回避できるように適切に配置することを含んでいる。また、ソフトウェアを新しい版やパッチで定期的に更新することも含まれる。

(3) 攻撃監視、検出、回復
権限のないアクセスを監視、検出し、攻撃への抵抗を行い、また、攻撃後の回復のためにバックアップをとる活動。

6.1 セキュリティリスク管理(Security risk management)

効果的なセキュリティエンジニアリングのためには、セキュリティリスクの評価と管理が最も肝要である。リスク管理は、システムの財産が攻撃された結果起こる可能な損失を評価することと、これらの損失を減少させる可能性のあるセキュリティ手順のコストに対してこれらの損失を天秤にかけることに関係する。クレジットカードの業務においては、クレジット詐欺を防ぐために新しい技術を導入するより、利用者の損失を補償することの方がおうおうにして安価である。リスク管理は技術上の問題ではなく、ビジネス上の問題である。

リスク評価と管理の重要な入力は、組織のセキュリティポリシーである。セキュリティポリシーは、常に維持されるべき条件を指定することにより、リスクと脅威を識別する。セキュリティポリシーは何が許され何が許されないかを定義しなければならない。

セキュリティエンジニアリングではこれらのポリシーを実現する設計する。

リスク評価のプロセスを以下のように提案する。

(1) 初期リスク評価
この段階では、詳細なシステム要求、システム設計、システム実現はまだできていない。ここでの重要な目的は適切なコストで適切なレベルのセキュリティが達成されるかどうかを評価することである。もしそうなら、システム特有のセキュリティ要件を定義する。システムまたは再利用コンポーネントやミドルウェアに関する潜在的な脆弱性については情報がない。

(2) ライフサイクルリスク評価
リスク管理は、システム開発ライフサイクルにおいて出現する。評価結果はセキュリティ要件の変更や新しい要件の付加を導く。既知の潜在的脆弱性が識別され、これらの知識はシステム機能の実現についての意思決定、どのように実現し、テスト

し、配置するのかを通知するのに使われる。

(3) 運用リスク評価

システムは配置され利用が開始された後、リスク管理は、どのようにシステムが利用されているかと新しいまたは変更された要件に注意し続けなければならない。組織の変更があることは、システムが初期に計画されていたのとは異なる方法で利用されることを意味する。

初期リスク評価とは、適切なセキュリティ要件を導出することである。第 12 章に初期リスク評価からセキュリティ要件の集合を導き出すかを示しておいた。ここでは、ライフサイクルリスク評価と運用リスク評価に焦点を絞る。リスク評価を実行するためには、システムに対する可能な脅威を識別する必要がある。一つの手段はミスユースケースを利用することである（Alexander, 2003; Sindre and Opdahl, 2005）。Pfleeger と Pfleeger(2007)は、以下の四項目で、可能なミスユースケースを識別する出発点として利用できるとした。

(1) 横取り(interception)脅威

横取り脅威は、攻撃者が財産にアクセスすることを可能にする。MHC-PMS に対するありうるミスユースケースは、攻撃者が著名人患者の記録へのアクセスを得ることである。

(2) 妨害(interruption)脅威

横取り脅威は、攻撃者がシステムの一部を利用不可能にすることを可能にする。ありうるミスユースケースは、攻撃者がシステムデータベースサーバに対するサービス拒否の攻撃である。

(3) 変更脅威

変更脅威は、攻撃者がシステムの財産を不法に書き換えることを可能にする。MHC-PMS でありうるミスユースケースは、攻撃者が患者の記録を改変することである。

(4) 偽造(fabrication)脅威

偽造脅威は、攻撃者がシステムに誤った情報を挿入することを可能にする。バンキングシステムにおいてありうるミスユースケースは、攻撃者がシステムに誤ったトランザクションを挿入することである。

ライフサイクルリスク評価は、セキュリティに影響を与える設計と実現の詳細を識別する。このことは、新しい要件を生成したり、システムの全体設計に影響を与えながら、（初期リスク評価で決定された）既存のセキュリティ要件の解釈に影響を与える。脆弱性の幾分かは、設計上の選択に依存する。パスワード方式に基づく脆弱性は、権利のある利用者が権利のない利用者にパスワードを漏らすことである。C を実装言語に選んだときの脆弱性は C コンパイラが配列オーバーフローの検査ができないことである。

6.2　セキュリティのための設計(Design for security)

システム実現後にセキュリティを追加するのは一般に困難である。システム設計時にセクリティ要件を取り上げなければならない。以下の点を取り上げる。

(1)　アーキテクチャ設計に関する決定事項がシステムのセキュリティにどのように影響するか。

(2)　セキュアなシステムの設計において何がグッドプラクティスとして受け入れられるか。

(3)　システムが利用のため配置されるとき、脆弱性の導入を避けるためにはどのようなサポートがシステムに設計されるべきか。

ディペンダビリティとセキュリティの関係にも注目する必要がある。ディペンダビリティを達成するために、冗長と分散を採用したとき、回復の手段に影響する。セキュアなシステムを設計する作業には何等かの妥協が含まれる。成功する攻撃を防ぐためには多数のセキュリティメジャーをシステムに設計しなければならない。これは余分の計算を必要とし、当然、性能も落ちる。また、セキュリティとユーザビリティの間には緊張関係がある。セキュリティメジャーはしばしば、余分の情報を記憶することを利用者に要求する。これらを利用者が忘れることが多く、システムの利用を不可能にする。設計者は上記の事柄のバランスを保つことが重要である。

6.2.1　アーキテクチャ設計

アーキテクチャ設計はシステムの創発特性に深淵な影響を与える。不適切なアーキテクチャを採用した場合、情報の秘密性(confidentiality)と整合性を維持することやシステム可用性のあるレベルを保証することは困難である。

システムアーキテクチャの設計を設計する際、以下の基本的な衝突する二つの論点を考慮すべきである。

(1) 防御
　大事な財産が外部からの攻撃から防御されるように、システムがどの　ように組織化（構造化）されるか

(2) 分散
　成功する攻撃の効果を最小にするために、財産をどのように分散させるか

階層化された防御アーキテクチャの例を図 4 に示す。

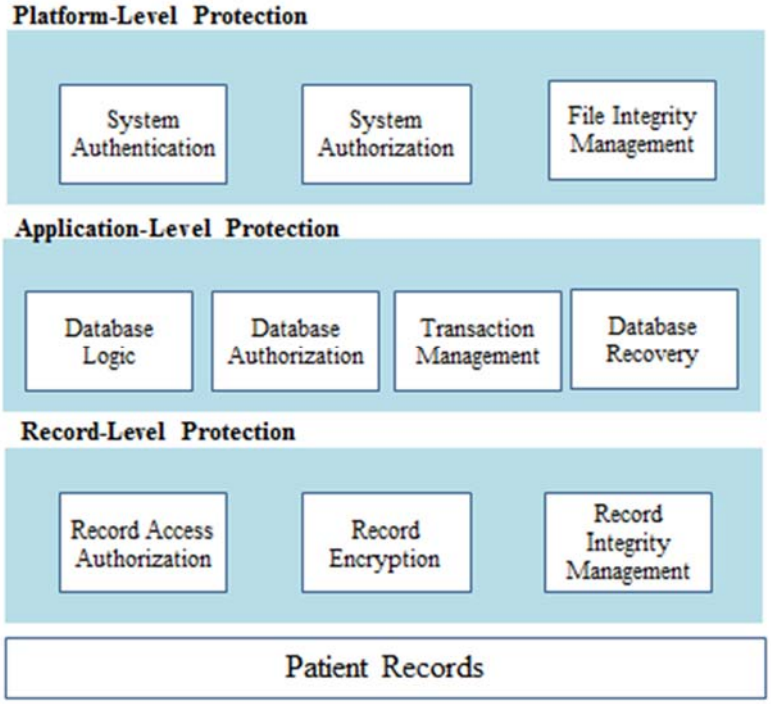


図 4　階層化された防御アーキテクチャ

6.2.2　設計ガイドライン

システムセキュリティを設計する際、一般的に通用するガイドライン 10 個を以下に示す。セキュリティ問題への気づき、またはチェックリストとしても有用である。

(1) 明示的なセキュリティポリシーに基づいてセキュリティに関して意思決定する

まず、セキュリティポリシーを定めるべきである。セキュリティポリシーは what を定め、how については言及しない。

(2) Single point failure（単一誤りがシステム全体に波及しない）を回避する

クリティカルはシステムでは、a single point failure を回避するよう試みることは、良い設計の実践である。a single point failure の回避とは、システムの一部における単一の障害がシステム全体の障害にならないようにすべきであることを意味する。セキュリティの立場からいえば、セキュリティを保証するのに単一のメカニズムに依存すべきではなく、いくつかの異なる技術を使用することを意味する。

(3) Fail securely（fail-secure にする）

システムの障害は避けがたい事柄であるので、fail-safe にすべきである。その意味でセキュリティクリティカルなシステムは **fail-secure** にすべきである。例えば、患者情報システムの場合、効率化のため、患者情報をクライアントにダウンロードして使用する場合、医療セッションの終わり故障が発生すると、クライアント中にデータが残ってしまう。fail-secure にする一つの手段は、患者データの暗号化である。

(4) セキュリティとユーザビリティのバランスを考える

セキュリティとユーザビリティに関する要求はしばしば対立する。セキュリティを厳しくすれば利用者は記憶すべきことが多くなる(impossible-to-remember 問題)、この問題に関しては、Cranor と Garfinkel の著書(2005)を参考にすべきである。

(5) 利用者活動のログをとる

可能な限り利用者の活動履歴を記録に残すべきである。少なくとも、誰が何をしたか、どの財産が利用されたか、活動の日付と時間が必要である。故障からの回復のための活動の再現、異常は活動の発見、内部からの攻撃の防止などに効果がある。

(6) リスク軽減のために冗長性と多様性を採用する

冗長性とは複数のバージョンのソフトウェアやデータを利用することである。多様性とは、異なるバージョンが同じプラットフォームに依存しないこと、同じ技術を使って実現されないことを意味する。例えば、サーバとクライアントで異なるオペレーティングシステムを採用するなどの例があげられる。

(7) すべての入力の正当性を検査する

共通する攻撃パターンに予期しない入力を投入することがあげられる。サービスの損失につながるシステムのクラッシュなどの例がある。バッファオーバーフロー攻撃、SQL ポイズニングなどはよくある攻撃である。これらの内のかなりのものは、入力の正当性を検査することで避けられる。

(8) **財産を区分けしよう**

区分け(compartmentalize)とは、システム中の情報にオールオアナッシングの形態のアクセスを提供しないことである。

(9) 環境設定（configure,構成）のための設計

多くのセキュリティ問題は、システムの運用環境に配置されるとき、システムが正しく構成されないことから生じる。

(10) 回復のための設計

どのように設計しても、セキュリティ故障は必ず発生する。そこで、回復手段が重要になる。バックアップ手段が大事である。

6.3 システム耐破壊性(System survivability)

現代の分散システムは、オフザシェルコンポーネントを含む基盤に依存する。再利用コンポーネントは異なる組織で開発される。そこでシステムのセキュリティはローカルな設計上の意思決定のみならず外部のアプリケーションによって影響される。これは、いかにセキュリティに注意を払ってもシステムが外部からの攻撃に抵抗できないことを意味する。つまり、いかにシステムの回復が早めるかを考える必要がある。耐破壊性と回復迅速性はシステムの創発特性である。クリティカルなサービスの可用性を維持するためには耐

破壊性は必須の事柄である。耐破壊性に富むシステムの設計において、Ellison とその仲間たちは以下の戦略を提案した。

(1) 抵抗（Resistance）
攻撃を拒絶する能力をシステムに組み込む。例えば、ユーザ認証のために電子証明を用いる。
(2) 認知（Recognition）
攻撃や誤り(failure)を検知する能力をシステムに組み込む。例えば、クリティカルなデータにはチェックサムを設ける
(3) 回復（Recovery）
攻撃中にも本質的なサービスを提供するための能力をシステムに組み込む。

さらに調査すべき論文リスト

1. R.J. Ellison, R.C. Linger, T.Longstaff and N.R. Mead, “Survivable Network System Analysis: A Case Study”, IEEE Software、16（4） July/August 1999.
2. J.Viega and G.McGraw, “Building Secure Software: How to Avoid Security Problems the Right Way”, Addison-Wesley, 2002.
3. R.Anderson, “Security Engineering：A Guide to Building Dependable Distributed Systems, 2nd edition”, Addison-Wesley, 2008.

6. ディペンダビリティ保証とセキュリティ保証(Dependability and Security assurance)

クリティカルシステムに対するディペンダビリティ保証のためのV&Vプロセスをサーベイする。

クリティカルシステムに対しては、以下の二つの理由により、厳格なテストと解析を必要とする。

(1) 故障のコスト
故障が起きる前に、欠陥を発見し除去することは全体としてのコストを下げる。
(2) ディペンダビリティ属性の妥当性確認
運用以前に、外部の監査人に、システムがディペンダビリティ要件を満たすことを保証する必要がある。

6.1 静的解析(Static analysis)

形式証明、モデル検査、自動静的解析

6.2 信頼性テスト(Reliable testing)

6.3 セキュリティテスト(Security testing)

Web-based システムに対する攻撃は、年々増加しており、セキュリティ評価に焦点をあてた Web-based システムの V&V プロセスは重要である。セキュリティテスティングが困難で

ある理由は以下の二つにある。

(1) セキュリティ要件は、shall-not requirements である。望ましくない振舞を定義するのはそれほど簡単なことではない。システムが何かをしないことを証明するのは不可能である。とくに想定外の攻撃に対しては要求を導きだすことさえできない。

(2) システムを攻撃する人間は知性が高くシステムの脆弱性を探し続けている。

脆弱性が攻撃の標的になる。故に、静的解析ツールはセキュリティテスティングのツールとして特に有用である。システムのセキュリティを検査するためには、テストとツールによる解析と形式証明を組み合わせて行う必要がある。

(1) 経験に基づくテスト

システムはすでに経験済みの攻撃に対して解析される。SQL poisoning やバッファオーバーフローに対してはテストすることができる。ツールによる解析と組み合わせて実施する。セキュリティ・チェックリストの例を以下に示す。

表 2　セキュリティ・チェックリストの例

アプリケーションで生成されるすべてのファイルは、適切なアクセス許可を持っているか？
システムが非活動の期間のあと、利用者セッションは自動的に終了されるか？
システムが配列境界の検査機能を持たない言語が書かれたとしたら、バッファオーバーフローが利用される状況が存在するか？
パスワードは強力か？　強力なパスワードは文字、数字、句読点からなり辞書には載っていない。
システム環境からの入力は入力仕様に基づいて常にチェックされているか？

(2) タイガーチームズ

開発チーム以外から、テストに関する経験を流用することは可能である。さらにいえば、システムセキュリティを破る目的でタイガーチームを設定する。彼らはシステムに対する攻撃をあらゆる方法でシミュレートする。タイガーチームのメンバーはテストやセキュリティ脆弱性の経験者である必要がある。

(3) ツールを利用したテスト

静的解析ツールやパスワードチェッカーなどを利用する。経験に基づくテストの拡張である。

6.4 プロセス保証(Process assurance)

主に安全性の保証に関する話題であり、詳細な解説は省略する。

ハザードロギングと監視、安全性レビュー、安全性保証

6.5 安全性事例とディペンダビリティケース(Safety and dependability cases)

主に安全性に関する事例であり、詳細な解説は省略する。

北陸先端科学技術大学院大学　高信頼組込みシステム教育研究センター　落水浩一郎

- 構造化ドキュメント：アシュアランスケース
- Structured Arguments: クレイム、エビデンス、オーギュメント

さらに調査すべき論文リスト

1. B.Goodenoughm J.J. Hudak, “Dependability cases”, CMU/SEI-2004-TN-016, 2004.
2. M. Andrews and J.A. Whittaker, “How to Break Web Software: Functional and Security Testing of Web Applications and Web Services”, Addison-Wesley, 2006.