DM6-03-051

別紙１：雛形

ソフトウェア開発における情報セキュリティ対策実施規程

（雛形）

2006 年 2 月

内閣官房情報セキュリティセンター

【本書の利用に当たって】

ソフトウェアにおけるセキュリティの実現については、開発ライフサイクル（SDLC:Software Development Life Cycle)におけるセキュリティ対策が重要な意味を持つ。このことから本書では、府省庁内部でソフトウェア開発を行う場合の「政府機関統一基準」の遵守事項の実装手法について解説する。
なお、ソフトウェアは、開発規模、開発期間、予算的制約等の相違によって、開発手法も多様化している。このため、本書が想定している開発手法ではなく、それぞれの組織がこれまで利用してきた開発手法を用いる方が適切な場合もあり得る。よって、この場合は本書に記載されたソフトウェア開発におけるセキュリティ対策実施規程を整理し、組織独自の開発手法の中に統合する必要がある。

本書に記載する要件は、完全かつ網羅的なものではないが、高いセキュリティが求められるソフトウェアを開発する際の出発点、及びISO/IEC15408（Common Criteria）導入の事前準備として有効に利用できる。また、ソフトウェア開発ライフサイクルにおいて考慮すべきセキュリティに関する事項について理解を深める際に役立つ参考文献を、本書中でその出所を記した上で紹介している。参考文献は、国際標準、又はインターネットにおいて入手が比較的容易であるものを優先して掲載している。

なお、本書においては、別途作成される策定手引書及び参考文献との併用を意図していることから、以下の内容に関する詳細な説明は本文に含めていない。

・情報の格付けに関する事項
・コーディングに関するセキュリティ事項
・ソフトウェア開発を委託する場合の調達・契約に関するセキュリティ事項
・セキュリティ設計仕様書（Security Target）の評価、確認に関する事項

本書の位置付け

本書は、各府省庁のソフトウェア開発手順書にセキュリティに関する事項を追加し、改善を促すための手引書の雛形であり、「ソフトウェア開発におけるセキュリティ対策実施規程　策定手引書」の２に示す実施手順に記載すべき事項を、同３に示す文書構成例の枠組みの中に盛り込み作成したものである。

手直しポイント

各府省庁において政府機関統一基準で規定する遵守事項を盛り込んだソフトウェア開発手順書を作成するには、大別して、新規で作成する場合と既存の手順書を修正する場合とがあるが、そのどちらの場合でも、以下の事項を踏まえて作業を行う必要がある。

(1) ソフトウェアは、その開発規模、開発期間、予算的制約等によって、最適な開発方法が異なることから、雛形の全要求事項を画一的に適用することは好ましくない。特に、比較的小規模なソフトウェアについては本書の要求事項の適用が現実的ではない場合が多い。このため、適用させる前に要求事項に所要の変更を加える必要性の有無を検討する必要がある。なお、雛形では、要求事項を箇条書きにして「●」記号を付加している。各府省庁においては、自組織における業務の内容と省庁基準にかんがみ、適宜、要求事項を追記又は削除する。また修正が必要となる箇所等には、以下の記号を付加している。

(a) 雛形中に、*[・・・]* 形式で明記される部分（省庁名、担当者等）については、各府省庁内の定めに合わせる。

(b) 雛形中に、【・・・の場合】形式で明記される記述については、想定される案を記したものであり、各府省庁の判断により適宜、選択又は修正する。

(c) ≪図・・・≫は、開発手順書の策定者向けの解説資料であり、適宜判断の上、修正又は削除する必要がある。

(d) ≪参考文献≫は、セキュリティの高いソフトウェアを開発するための参考資料であり、適宜判断の上、修正する必要がある。

(2) ソフトウェア開発における役割分担については、組織や開発プロジェクトによって様々であるため、各府省庁の開発手順書では、自組織の構成や各担当者のソフトウェア開発に関する責務を考慮した上で、主語の追記又は変更を検討する。

(3) 雛形において使用しているソフトウェア開発に関わる用語等については、府省庁において既に用いられている用語と平仄を揃える。例えば、ソフトウェア開発の工程を意味する「要件定義」「設計」という用語等は共通化された呼称ではなく、組織やプロジェクトによって定義や利用方法が異なっているため、必要に応じて

修正を加える。

(4) 既存の情報セキュリティ関係規程との整合性を考慮し、適切な分割、統合、相互参照を検討する。

(5) 情報セキュリティ対策の観点以外の一般的な記述について、雛形の内容では不足があると思われる場合には、適宜、補う。

改訂履歴

改訂日	改訂理由
2006/2/17	初版

商標について

本資料に記載されている会社名、製品名は、それぞれの会社の登録商標又は商標です。

目次

1　本書の背景と目的

1.1　本書の背景

近年、業務全体のうち、何らかのソフトウェアを用いるものが占める比重は増大しており、これに比例するようにソフトウェアは大規模化、複雑化の一途をたどっている。これに伴いソフトウェアの品質を確保することが困難となったことから、この課題を解決する様々なソフトウェア開発の方法論が検証され、その標準化が進められている。代表的なものとして、ソフトウェアのライフサイクルの観点から検討された共通フレーム 98 や CMMI、あるいはマネジメントの観点のから検討された PMBOK や ISO9001 などが挙げられる。

また、一時代前においてはソフトウェアが満足すべき品質の一要素でしかなかったセキュリティは、高度化する情報通信ネットワークへの業務の依存度が著しく増大したことによって、リスクマネジメントの観点から、コストや納期、又は保守性や操作性といった品質に優先して組織が取り組むべき最優先の課題へと押し上げられることとなった。こうしたことから、ソフトウェア開発において、『セキュリティを考慮した開発方法論』が必要とされてきている。

セキュリティ面からのソフトウェア開発へのアプローチとしては、代表的なものとして、IT 関連製品のセキュリティ評価手法である ISO/IEC15408（Common Criteria：CC）が存在する。しかしながら、CC は高い網羅性を持つ万能のセキュリティ標準として作成されていることから、一般のセキュリティ管理者にとっては、個々の情報システムに適用させるのが難解であるという一面もあるため、十分に普及しているとはいえない状況である。

1.2　本書の目的

本書は、セキュリティの高いソフトウェアを開発するために、*[○○省]*で採用しているソフトウェア開発手順を、*[政府機関統一基準]*が定めるセキュリティの観点から補完し、改善を促すことを目的としている。

セキュリティの高いソフトウェアを開発するために実施すべき事項を概観すると、以下のとおりとなる。

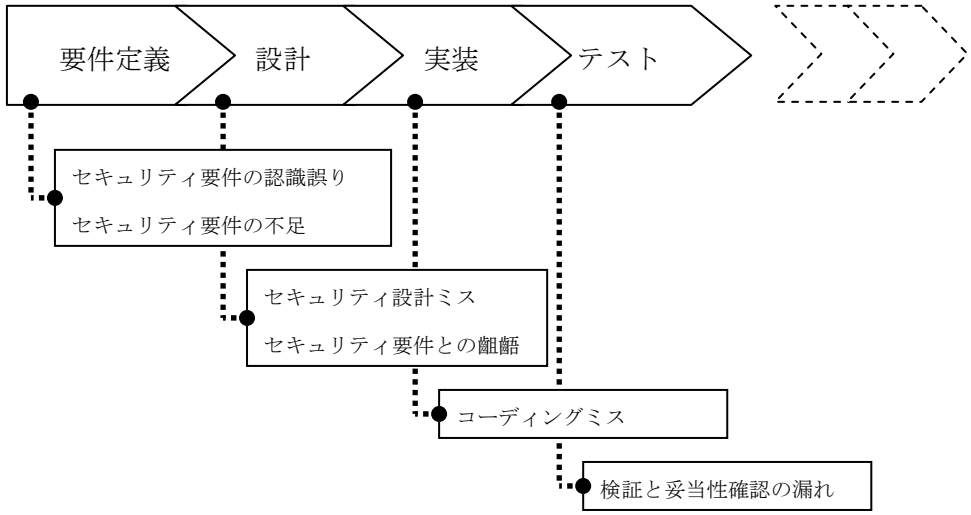
(1) 開発作業、場所、情報資産、要員に関する適切なセキュリティの管理
(2) 適切なセキュリティ機能の設計及び実装
(3) セキュリティの脆弱性の排除

そして、これらの事項を達成するためには、組織のソフトウェア開発手法をセキュリティの観点から修正し、セキュリティを確実に実現できる開発手法に改善する必要

がある。

具体的には、ソフトウェア開発のライフサイクル（要件定義、設計、実装、テスト）の各工程について、混入されるおそれのある脆弱性を排除し、セキュリティの品質を向上させるために実施すべき対策を開発手法に含めることにより、これを実現する。

≪図 1-1：ソフトウェア開発ライフサイクルにおいて混入する脆弱性≫



本書はセキュリティの高いソフトウェアを開発するに当たって実施すべき標準的な事項を*[○○省]*内部でソフトウェアを開発する場合を想定して記載したものである。この趣旨から、外部委託によりソフトウェアを開発する場合においても、[情報システムにおける情報セキュリティ対策実施規程及び外部委託における情報セキュリティ対策実施規程と併用することで]発注者として外部委託事業者を適切に管理するための資料として有効に活用できる。

≪参考文献≫

- SLCP-JCF／共通フレーム 98（ISO/IEC 12207）
  ソフトウェア開発及び取引を明確化し、利用者と開発者が共通の尺度を持つための枠組みを解説したもの。ソフトウェア構築時の作業手順や役割分担を明確化すること、契約時の相互の認識の不一致を防止することを目的として策定されている。
- NIST Special Publication 800-64「Security Considerations in the Information System Development Life Cycle」
  情報システムの開発ライフサイクル(System Development Life Cycle)においてセキュリティを確保するための枠組みを解説したもの。
- Microsoft：「信頼できるコンピューティングのセキュリティ開発ライフサイクル」

（http://www.microsoft.com/japan/msdn/security/general/sdl.asp）

重大な脅威にさらされるソフトウェア向けにマイクロソフト社が採用しているセキュリティ開発ライフサイクル(Security Development Lifecycle)を示したもの。

2　本書の対象者

本書は、セキュリティの立場からソフトウェア開発に携わる*[情報システムセキュリティ責任者]*を対象とする。

3　開発体制の構築及びソフトウェア・情報資産の保護

【趣旨】

セキュリティの高いソフトウェアを開発するためには、そのために必要となる措置を確実に実現できる体制の確保が必須である。

また、開発されるソフトウェア及び開発において使用される情報を適切に保護するためには、開発に用いる施設や環境、開発に要する情報資産に関するセキュリティの管理も考慮しておく必要がある。

本項では、セキュリティを実現するためのソフトウェア開発における体制の構築やこれに係る情報資産の保護について解説する。

3.1　開発体制に係るセキュリティ

開発担当者にセキュリティに関する責務が正式に割り当てられていなかったり、担当者の能力が不足している場合、セキュリティの高いソフトウェアの開発は不可能である。セキュリティの高いソフトウェアを開発するために必要とされる作業を明確化した上で、専門的な知識を持った要員の配置や当該要員に対する教育等、セキュリティにかかわる対策事項を満たすことが可能な開発体制を構築しなければならない。

(1) ソフトウェア開発に係る役割の明確化

- ソフトウェア開発の責任者に対して、セキュリティの高いソフトウェアを開発するために必要となる作業を特定し、十分な人員を割り当てることを要求すること。

セキュリティの高いソフトウェア開発を確実に実現するために、ソフトウェアのセキュリティ要件に応じて必要な各作業（要件定義→設計→実装→テスト）についてその担当者を明確にし、その的確な実施を求めなければならない。

(2) 開発担当者へのセキュリティ教育

- ソフトウェア開発の責任者に対して、開発担当者に対する適切なセキュリティ教育を行うことを要求すること。

ソフトウェアを開発する作業を行う担当者に、割り当てられたセキュリティに関する役割を確実に達成させるために、知識や能力を向上させるためのセキュリテ

ィ教育を求めなければならない。

3.2 ソフトウェア・情報資産の保護

開発場所の入退室管理や開発に要する情報資産の管理に不備があれば、開発に関する機密性の高い情報が外部に流出したり、不正なプログラムがソフトウェアに埋め込まれるおそれなどが高まる。このため、セキュリティの高いソフトウェアを開発する前提として開発を行う場所や開発に要する情報資産に係るセキュリティを確保しておかなければならない。

(1) 施設と環境のセキュリティ

- *[政府機関統一基準]*に準拠して、ソフトウェア開発を行う施設と環境を安全区域として保護すること。

【運用中の情報システムとの分離が必要な場合】

- 運用環境とは物理的・論理的に分離された環境で、ソフトウェアの開発・試験を実施すること。また、開発担当者が運用環境において作業を行う場合は、責任者の承認を受け、作業内容を記録すること。

悪意のある者が容易に接触できる状況においては、セキュリティを維持しつつソフトウェアを開発することが困難である。このため、重要なセキュリティ要件を持つソフトウェア開発を行う施設については、*[政府機関統一基準]*に準拠して、入退室管理を含めた安全区域の物理的管理を実施する必要がある。また、運用中の情報システムへの悪影響を防ぐため、開発・試験環境と運用環境は物理的・論理的に分離し、開発者の運用環境へのアクセスを制限しなければならない。

(2) 開発に要する情報資産のセキュリティ

- ソフトウェア開発時に必要となる要保護情報及び関連する情報資産は、責任者の承認を受けた上で利用し、利用状況を記録すること。また、利用が許可された情報等は適切に保護すること。
- コンパイラ、エディタ、その他ユーティリティ等の開発に要する資産は、責任者の承認を受けた上で利用し、利用状況を記録すること。また、利用が許可された資産等は適切に保護すること。
- 本番運用データは原則テストデータとして使用しないこと。やむを得ず使用する際は要機密情報を消去した上で使用すること。
- ソースコードついて、許可された利用者以外のアクセス（閲覧・変更）を制限し、滅失、き損等に備えたバックアップの取得を行うこと。

ソフトウェア開発の担当者は、*[○○省]*の所有する要保護情報及び関連する情報資産の取扱いを任され、当該情報等を通常とは異なった運用に用いることになる。このような場合においては、情報資産を盗難、改ざん、滅失等から防ぐために使用に際しての申請、承認及び記録並びに使用後の返却及び消去を徹底するなど、当該情報等について厳格な管理を行わなければならない。

【セキュリティ要件を明確にする必要がある場合】

4　セキュリティの要件定義

【趣旨】

要保全情報を取り扱い、不特定多数の人間が接続するインターネット上で利用されるソフトウェアについては、外部からの攻撃等による改ざんが生じないセキュリティ品質が要求される。また、要機密情報を取り扱うソフトウェアについては、情報の漏えいが生じないセキュリティ品質も要求される。このように、個々のソフトウェアの重要性や利用環境に応じて必要とされる要求事項は異なるため、それぞれの場合について、情報資産を守るために求められるセキュリティの要求事項が何であるかを検討しなければならない。この要求事項を「セキュリティ要件」と呼ぶ。
本項では、ソフトウェアのセキュリティ要件について解説する。

4.1　セキュリティ要件の定義

ソフトウェアのセキュリティ要件は、開発するソフトウェアが運用される際に関連する情報資産に対して想定されるセキュリティ脅威の分析結果、及び当該ソフトウェアにおいて取扱う情報の格付けに応じて決定される。セキュリティ機能の必要性を認めた場合、セキュリティ要件をセキュリティ要件定義書として文書化する。

(1) セキュリティ要件定義書の作成

- ソフトウェアのセキュリティ要件定義書を作成し、これに基づきセキュリティの設計と実装を行なうこと。

ソフトウェアのセキュリティ要件について、共通の認識を持ち、組織としてセキュリティに取り組むためには、要件定義書として文書化しなければならない。セキュリティ要件定義書はセキュリティ機能の設計と実装を求めるための根拠となる。
なお、セキュリティ要件定義書の作成に当たっては、開発対象となるソフトウェ

アのセキュリティを確保する観点のみならず、直接・間接に結びついた他の情報システムへの影響も考慮に入れておく必要がある。これは、例えば対象となるソフトウェア自体の重要度が低い場合であっても、不正侵入の脅威が大きい環境に設置されるのであれば、踏み台にされ、他の情報システムを脅威にさらす可能性があるからである。

(2) *[政府機関統一基準]*の遵守との整合性

- *[政府機関統一基準]*と整合性を持つセキュリティ要件定義書を作成すること。

*[○○省]*が遵守すべきルールや前提条件に準拠したソフトウェアを開発するため、セキュリティの要件定義書は、*[○○省]*のセキュリティの遵守事項をまとめた*[政府機関統一基準]*と整合性を持つ必要がある。

【セキュリティ機能が必要な場合】

5　セキュリティ機能の設計・実装・構成管理

【趣旨】

「ソフトウェアが行うべきこと（What)」を定めたセキュリティの要件定義に則って、「ソフトウェアが行う方法(How)」を設計し、この設計に基づいて確実な実装を行なわなければならない。

また、ソフトウェアの大規模化、複雑化が進むにつれて、開発の初期段階において要件や設計を完全に定義することは困難となり、仕様の変更は避けられない事象となりつつある。こうした変更は、セキュリティの脆弱性を生じさせる原因となるおそれがあるため、適切な構成管理が重要である。

本項では、セキュリティ機能の設計と実装及び構成管理について解説する。

5.1　セキュリティの設計

ソフトウェアのセキュリティ要件定義により明確にされた要求事項を満たすソフトウェアを開発するためには、その要求を満たすための具体的な機能を設計しなければならない。この際、セキュリティの実装の段階で齟齬が発生しないように、設計書を正確に具体化していく必要がある。

また、あらゆる脅威を想定し、それらに対して完全な措置を講ずることは、実際には困難であることから、セキュリティが侵害された場合の対策をあらかじめ設計をしておく必要がある。

(1) セキュリティの設計

- *[政府機関統一基準]*とセキュリティの要件定義に準拠してセキュリティの設計を行うこと。

セキュリティの要件定義を精査した上でセキュリティに関する設計を行なう。セキュリティの設計は、セキュリティ要件定義を満たす網羅的な構成を取る必要がある。
加えて、セキュリティの設計は政府機関が遵守すべきセキュリティ事項を網羅している政府機関統一基準を前提条件とした内容となっていることを確認しなければならない。これによって、強力な認証やアクセス制御、妥当性の検証など、強固なセキュリティの設計を行なう必要がある。また、客観性や網羅性を保つ観点から、政府機関統一基準に加えて、その他の信頼できる文献を併用することも有益である。

≪参考文献≫

- ISO/IEC 15408 Common Criteria(JIS X 5070)
  IT 製品及びシステムのセキュリティ機能の開発を対象とした評価規格を示したもの。概説（Part1)、セキュリティ機能要件（Part2)、セキュリティ保証要件（Part3）の三部から構成される。CC の Part2 は、セキュリティ機能要件のセットであり、セキュリティ設計・実装の参考書として有効に活用できる。
- ISO/IEC 17799（JIS X 5080）
  組織の情報セキュリティマネジメントに基づいて推奨されるべきセキュリティコントロールを体系的に定めた規格を示したもの。
- NIST Special Publication 800-53「Recommended Security Controls for Federal Information Systems」
  米政府の各省庁における情報システムの分類された結果（Low/Moderate/High）に基づき実施すべきセキュリティコントロールのベースラインセットを示したもの。

(2) 設計書におけるセキュリティ機能の具体化

- セキュリティの設計を設計書において具体化し、記述すること。

ソフトウェアの設計は、作業段階に応じて基本設計から詳細設計へと具体化され、設計書として文書化される。
ソフトウェアの開発規模、複雑性等によって、どのレベルまで詳細な設計書を必要とするかは異なるが、実装段階での齟齬を防止するため、セキュリティ機能を具体化し、明確に記述しておく必要がある。

【セキュリティ機能として、セキュリティの管理機能が必要な場合】

(3) セキュリティの管理機能の設計

- 開発するソフトウェアが運用される際に利用されるセキュリティ機能の管理機能を設計すること。

「管理機能」とは、真正確認、権限管理等のセキュリティ機能を管理するための機能のほか、証跡保全の機能等の故障、事故、障害等の発生時に行う対処及び復旧にかかわる機能等を指す。これらの機能をソフトウェアのセキュリティ要件に応じて、セキュリティの管理機能をソフトウェアに組み込む必要がある。

【開発するソフトウェアに重要なセキュリティ要件がある場合】

(4) ST 評価・ST 確認の実施

- 重要なセキュリティ要件があるソフトウェアに対して、セキュリティ設計仕様書（Security Target）の評価・確認を実施すること。

開発するソフトウェアに重要なセキュリティ要件があると判断されたソフトウェアについては、セキュリティ要件定義の結果を受けて、セキュリティ設計仕様書（Security Target）を策定し、セキュリティ機能の設計において、第三者機関による ST 評価・ST 確認を受ける必要がある。

【権限管理の設計が必要な場合】

5.2 設計のポイント - 権限管理

ソフトウェアにおいて要機密情報の漏えいなどを防止する手法のうち一般的なものとして、権限管理機能がある。『権限管理』とは、識別と真正確認の結果である『認証』に対して、『アクセス制御』すなわち、権限の割当てや制限を強制することである。

(1) 権限管理対象とする情報の識別

- ソフトウェアで権限管理を行うべき情報を識別すること。

ソフトウェアのセキュリティ要件を定義する際に実施した当該ソフトウェアで取り扱う情報の格付けの結果に基づき、「権限管理」すべき情報を識別しなければならない。その上で、必要とされる権限管理の強度を慎重に判断する必要がある。この際、ソフトウェアが取り扱う行政事務情報以外の重要な情報にも注意する必要がある。特に識別子、パスワード、暗号鍵、ソースコード、パラメータ、セッ

ション情報、監査ログ等のシステム構成情報については、慎重な検討を要する。

(2) 権限管理の設計

- *[政府機関統一基準]*に準拠して、ソフトウェアのセキュリティ要件に応じた識別（特定された一意の識別子など)、認証（知識・所有・生体による認証）に基づいた権限管理機能及びそれを支援するための暗号、電子署名、証跡管理等の機能を設計すること。
- *[政府機関統一基準]*に準拠して、識別された要保護情報に適切な権限管理を行なうこと。
- *[政府機関統一基準]*に準拠して、プログラムが所有する特権を特定し、必要最小限の権限に制限すること。

権限管理対象の識別結果に基づいて、権限管理の設計を行う。この際、政府機関統一基準　4.1　情報セキュリティについての機能に準拠した十分な強度の権限管理を設計し、識別された情報を保護する必要がある。特に、システム構成情報はソフトウェアのセキュリティにかかわる情報であり、システム構成情報のセキュリティが侵害された場合、セキュリティ機能が迂回されたり、無力化される可能性があるため、慎重な権限管理の設計が必要である。
また、権限管理の設計においては、必要な情報に適切な権限管理を行い、不要な権限を割り当てないことが重要である。

【情報の妥当性の検証に係る設計が必要な場合】

5.3　設計のポイント - 情報の妥当性の検証

ソフトウェアは、入力された情報を正確に処理し、その結果を出力することをその基本的な目的としている。この目的を確実に達成するため、妥当性の検証によって、処理の誤りや悪用される可能性のある情報などから、情報と処理方法の完全性を保護する必要がある。
特に悪意のある情報の入力によってソフトウェアの完全性を侵害する攻撃は、現在ソフトウェアに対する最も重大なセキュリティ脅威となりつつあり、その対応に注意が必要である。

(1) 妥当性検証を行うべき情報の識別

- ソフトウェアで妥当性検証を行うべき情報を識別すること。
- 利用者及び外部プログラムとやり取りするものを含めてすべての入力される情報について、悪意ある情報が含まれると仮定した上で、妥当性の検証を行

うこと。

「妥当性の検証」を行うべき情報を識別しなければならない。
この際、特にソフトウェアに「入力」される情報には注意を払う必要がある。入力される情報は常に信頼できるものとは限らない。すべての入力される情報には悪意ある情報が含まれると仮定した上で、識別子、パスワード、入力フォームといった開発者が「入力を想定している情報」だけではなく、「入力を許可している情報」に対しても識別を行わなければならない。
なお、外部の利用者による入力だけでなく、当該ソフトウェア以外の外部のプログラム等とやり取りされる情報についても、ソフトウェアの内部で検証を行う必要がある。

(2) 情報の排除

- 悪用される可能性のある情報を識別し、排除すること。

情報の排除とは、悪用される可能性のある情報の型等を定義し、当該情報が含まれる場合に不正な部分の除去、置換等の処理を行なう手法である。
悪用される可能性のある情報としては、プログラム内部や最終的な出力において特別な意味を有する特殊文字がある。情報の排除は、可能な限り手作業を排し、自動化されたプロセスを実装することが望ましい。
なお、処理を OS やミドルウェアなど外部のプログラムに引き渡す場合は、情報を使用するプログラムと検証するプログラムが異なることから、作業漏れが発生しやすい。こうした場合においても悪用される可能性のある情報を確実に排除しておく必要がある。

(3) 入力の制限

- 入力を許可する情報の型等を定義して、合致した情報のみの入力を許可すること。
- 悪用される可能性のある情報の型等を定義して、その定義に合致する情報の入力を拒否すること。
- 許可されない入力に対しては、リスクを想定して適切なエラー処理を実施すること。

入力の制限とは、許可した情報以外の入力を拒絶し、リスクに応じて適切なエラーの処理（再確認、終了、警告等）を行う手法である。
これには、入力を許可する情報の型等を定義して、その定義に合致した情報のみ

の入力を許可する手法と悪用される可能性のある情報の型等を定義し、その定義に合致した情報の入力を拒否する手法がある。これらのうち、適切な手法について検討し、実施する必要がある。

5.4 セキュリティの実装を支援するための枠組み

セキュリティの実装は、個々のプログラマのスキルによって差異が生じがちである。責任者の視点から重要な点は、プログラマ個々の知識に依存せず、コーディング規約の統一等によって開発方法を標準化することで、実装時に混入する脆弱性を最小限に抑制する点にある。

(1) コーディング規約におけるセキュリティ

- 最新のセキュリティ技術を反映したセキュアコーディングに関する注意事項を記載したコーディング規約を作成すること。
- セキュアコーディングに関する注意事項について、定期的にプログラマに教育を実施し、その浸透を図ること。

プログラミング作業の品質と保守性を向上させるために、「コーディング規約」を作成する。「コーディング規約」は命名、スタイル、コメント、改行・インデント、関数・クラス・変数の取扱い等、様々な事項をルールとして網羅的に定めたものである。
ソフトウェアを開発する組織は、実装段階における脆弱性の混入を防ぐため、コーディング規約にセキュリティに関する注意事項を記載し、教育などを通してその運用を徹底し、個々のプログラマの知識に依存しない仕組みを確立することによって、確実なセキュリティの実装を図る必要がある。
組織がコーディング規約を策定する際のセキュリティ事項については、文献を参考にできるが、参考文献のみでは最新の攻撃手法に対応することが難しいため、外部の専門家の支援や専用ツールの購入などの手法で、コーディング規約を更新していくことも必要である。

≪参考文献≫

- IPA「セキュア・プログラミング講座」
  （http://www.ipa.go.jp/security/awareness/vendor/programming/）
  セキュリティの脆弱性を発生させないようなプログラミングテクニックを示したもの。

5.5 構成の管理

不十分な構成管理は、開発者によるバックドアや隠れチャネルの埋め込みなどのセキュリティ上の深刻な問題を引き起こす原因となる。また、現在のソフトウェア開発においては、開発の過程で、仕様の変更が発生することが多い。これらの問題に対処するため、ソフトウェアの構成要素の変更、追加、削除を管理し、ソフトウェアの完全性を確保するための手続である構成管理を適正に実施する必要がある。

(1) 構成要素の識別

- ソフトウェア開発に関する構成要素を識別した上で、各構成要素を一意に識別し、バージョン番号を付与し管理すること。

セキュリティの高いソフトウェアを開発するためには、各工程において作成される仕様書、ソースコードその他管理の対象とすべき構成要素のセキュリティを保護しなければならない。
こうしたソフトウェアの各構成要素は、一覧を作成する等の手法で明確に識別して、バージョン管理を行う必要がある。また、識別された構成要素に対して、アクセス制御やバックアップの取得などの十分な保護対策を実施しなければならない。構成要素として、下記のようなものが想定できる。

(a) 開発プロジェクトの計画関連のドキュメント
(b) 実装関連ドキュメント
(c) 要件・設計ドキュメント
(d) テスト関連ドキュメント
(e) 運用ガイダンス
(f) 構成管理ツール

(2) 構成変更の管理

- 構成要素の変更管理を行う前提として、開発関係者の合意の結果をベースラインとして定義し、許可された利用者以外のアクセス（閲覧・変更）から保護すること。
- ベースラインの設定以降に構成要素の変更を行う場合には、正式な変更申請手続を行うこと。変更作業の内容は記録として保持すること。
- 構成要素の変更申請に対して、ソフトウェア開発を行う責任者はその影響を検証すること。変更作業は、緊急時も含めて承認された後に実行すること。

ソフトウェアの開発過程における仕様変更や発見された欠陥に対処するためのプログラム・モジュール構成の修正又は当該修正に伴うドキュメントからの乖離は、セキュリティの脆弱性を発生させる原因となる。
このため、各構成要素について、ある時点での開発関係者の合意の結果をベースラインとして定義し、ベースラインが設定された以降の変更は、正式な手続を通して厳格に管理する必要がある。
なお、変更が発生する要因としては、仕様が変更された場合と欠陥が発見された場合の修正が考えられる。仕様変更は、さらに要求変更、設計変更、実装変更に分類することができる。ソフトウェア開発を行う責任者はこうした変更の必要性と影響を検証した上で変更内容を承認し、かつ修正に関する記録を残しておかなければならない。

(3) 構成管理の自動化

- 構成管理ツール等の利用により、正確かつ効率的な構成管理を行うこと。

構成変更は、開発するソフトウェアの規模が大きく複雑になると、大量に発生する可能性がある。また、特定の構成要素の修正によって、他の構成要素に変更が生ずることも想定される。
このようなことに対処し、構成管理作業の正確性や効率性を確保するために、構成管理ツール等を利用しなければならない。

6　セキュリティの検証と妥当性確認

【趣旨】

ソフトウェアは、その開発ライフサイクルの各工程で的確な作業がなされることで、本来求められるセキュリティをはじめて確保することができる。そして、各工程での作業の的確性を判断し、ソフトウェアの脆弱性を可能な限り減少させるためには、厳格な検証と妥当性確認が不可欠である。
本項では、セキュリティの視点からの検証と妥当性確認、及びその際のポイントとなる既知の攻撃手法について解説する。

6.1　セキュリティの検証と妥当性確認

高い品質のソフトウェアを開発するためには、各工程が前の工程の成果物を受けて正しい作業が行われているかを検証し、かつ各工程の作業結果が、上流工程で定められた目的や要件に合致しているかの妥当性の確認を徹底する必要がある。この検

証と妥当性確認は、セキュリティの品質を保証する意味でも、中核的な位置付けとなる作業である。

(1) 開発工程へのセキュリティの検証と妥当性確認の組込み

- セキュリティの検証と妥当性確認を開発工程に組み込むこと。

ウォーターフォールモデルでもスパイラルモデルでも基本的な開発の流れは「要件定義→設計→実装→テスト」の順序で進行する。いずれのモデルを採用するにせよ、各工程にセキュリティの検証と妥当性確認を計画的に組み込む必要がある。

【セキュリティの専門家による確認が必要と判断した場合】

(2) セキュリティの専門家による検証と妥当性確認

- セキュリティの検証と妥当性確認を行うための専門家が、開発者による設計や実装作業が適正であるかどうかを確認すること。

脆弱性は、開発者の思いこみや盲点を原因として発生する場合が多い。この対策として、セキュリティの検証と妥当性確認を行う専門の担当者を配置する必要がある。

なお、検証と妥当性確認を行う担当者（レビューを行うレビュアとテストを担当するテスタ）は、セキュリティに関する十分な知識、経験を備えた専門家として、開発者による設計や実装作業が適正であるかどうかをセキュリティ面から確認する必要がある。

【セキュリティベンダのサービスの利用が必要と判断した場合】

(3) セキュリティベンダを利用した検証と妥当性確認

- セキュリティ検証と妥当性確認のために外部のセキュリティベンダによるサービスを利用すること。

本来は、自組織内でセキュリティの専門家を育成し、配置することが望ましい。しかし、現実的には十分なスキルを持つ専門家を自組織内で配置することは極めて困難である。このような場合においては、必要に応じて外部のセキュリティベンダを利用することも有効な解決策となり得る。

セキュリティベンダによる検証と妥当性確認は、自組織内での作業では不足することが想定される検査項目の網羅性や最新技術への対応が期待できることから、ソフトウェアのセキュリティ要件に応じて、実施の是非を検討する必要がある。

【ツールの利用が必要と判断した場合】

(4) セキュリティツールを利用した検証と妥当性確認

- コード検査ツール等の利用により、正確かつ効率的なセキュリティの検証と妥当性確認を行うこと。

セキュリティの検証を全部手作業で実施すると、膨大な時間を要すると同時に作業ミスも発生しやすくなる。このため、検証と妥当性確認の作業は、セキュリティツール等の利用によって可能な限り自動化すべきである。

例えば、ソースコードを検査するツールの中には、危険な関数や変数の利用をスキャンし、不具合を修正する機能等を有するものがあり、高いセキュリティ品質を備えたソフトウェアの開発において不可欠となりつつある。また、最新の攻撃手法等を使用して脆弱性を検査するツールもある。

ただし、こうしたツールは技術的に発展途上の状態にあり、誤検出等のおそれも高いことから、十分に熟練した開発者の作業を支援する手段として利用する必要がある。

6.2 セキュリティに関するレビューとテスト

ソフトウェアの検証と妥当性確認を行うための方法論はレビューとテストであり、セキュリティのレビューとテストの徹底によって脆弱性は減少する。

レビューは、テストに比して軽微な工数で済むとされ、効率的な欠陥予防・除去の観点から、極めて重要なプロセスである。しかし、レビューのみではすべての欠陥を排除することは難しいため、実際にプログラムを動かして確認するテストを実施する必要がある。

なお、ソフトウェア開発における一般的な動作確認のレビューやテストが、「想定される行動が行われた際に要求された機能が確実に動作する点」に比重を置くのと対照的に、セキュリティのレビューやテストは、「想定外の行動が行われた際に問題のある動作が発生しない点」に重点を置いている。本項では、セキュリティのレビューとテストについて解説する。

(1) セキュリティに関するレビュー

- 実装を開始する前に*[定められた各工程の開発関係者]*において、セキュリティの設計に関するドキュメント（要件定義書、基本設計書、詳細設計書等）のレビューを実施し、セキュリティ要件定義に基づく設計が行われていることを確認し、その合意した内容について記録すること。

- 実装を終了する前に*[定められた各工程の開発関係者]*において、セキュリティの実装ドキュメント（ソースコード等）に関するレビューを実施し、セキュリティ設計に基づく実装が行われていることを確認し、その合意した内容について記録すること。
- 各セキュリティドキュメント（要件定義書、基本設計書、詳細設計書、ソースコード等）間における要件の対応関係を明確化し、セキュリティ要件が正確かつ完全に具体化されていることを確認すること。

レビューとは、各工程の成果物としてのドキュメント（要件定義書、基本設計書、詳細設計書、ソースコード等）について開発関係者間の合意を得る手続であり、セキュリティ面からのレビューも実施する必要がある。レビューの手法は、インスペクション、ピアレビュー、ウォークスルー等が一般的である。
ソフトウェア開発は、下流工程に進むにつれて、要件の抜けや認識の誤り等の齟齬が発生することが多い。特に実際の開発作業のスタートラインである要件定義から基本設計までの工程でミスや抜けが生じていた場合は、多大な手戻り工数が発生してしまう可能性があることに留意し、上流工程におけるレビューは特に慎重に実施する必要がある。

【セキュリティテストが必要な場合】

(2) セキュリティに関するテスト

【ホワイトボックス形式でのセキュリティテストが必要な場合】

- セキュリティ問題を想定し、それに対する対策を実践できるセキュリティ知識を備えたテスタが、ホワイトボックス検査によるセキュリティテストを行うこと。

【ブラックボックス形式でのセキュリティテストが必要な場合】

- 使用するプログラミング言語に対する既知の攻撃手法を熟知し、それに対して新たな脆弱性を発見できるセキュリティ知識を備えたテスタがブラックボックス検査によるセキュリティテストを行うこと。

テストとは、作成したプログラムを実際に稼働させ、その処理の結果が想定したものと一致しているかどうか等を検証・確認する手続であり、単体テスト、結合テスト、統合テストといった工程において、性能、信頼性、負荷テスト等に加え、セキュリティ面からのテストも実施する必要がある。
テストの手法は、ホワイトボックス検査とブラックボックス検査に大別される。このうち、ホワイトボックス検査は、主に単体テストや結合テストで使用される手法であり、プログラムの内部構造が開示された状態で検査を行うことから、入

力に対するプログラムの挙動を確実に把握することができる。一方、ブラックボックス検査は、主に統合テストで使用される手法であり、プログラムの内部構造を確認せずに仕様やインタフェースに基づきテストを行う。これらのうち、適切な手法について検討し、実施する必要がある。

【セキュリティテストが必要な場合】

6.3 既知の攻撃

既知の攻撃を想定した脆弱性の検証は、セキュリティのレビューとテストの効果を高める重要な要素となる。なぜなら、多くの攻撃が同様の手法によって行われるからである。加えて、既知の攻撃を想定した脆弱性の検証は、レビューとテストに客観性をもたらし、作業漏れを防止する効果もある。
なお、攻撃手法は一定ではないため、常に最新の動向を把握しておくことが重要である。本項では、例示として代表的な攻撃手法の概要について解説する。

【アクセス制限の回避に関するテストを実施する場合】

(1) アクセス制限の回避

- アクセス制限の回避に関する脆弱性の検証を行うこと。

攻撃者は、識別や認証、アクセスコントロールの抜けや誤りによって生ずる権限管理の脆弱性を突いて、機密性の高い情報に許可されないアクセスを行う可能性がある。

【パスワード推測に関するテストを実施する場合】

(2) パスワード推測

- パスワード推測に関する脆弱性の検証を行うこと。

知識による認証であるパスワードは、十分な時間さえあれば理論的には解読は可能である。パスワード推測の手法としては、予測され得るパスワードを推測する辞書攻撃、トライアンドエラーを繰り返す総当たり（Brute force）攻撃などが代表的である。
攻撃者は、パスワードを自動で推測するツール等を利用することで、権限管理の脆弱性を突いて、パスワードを不正に取得する可能性がある。

【権限昇格に関するテストを実施する場合】

(3) 権限昇格

- 権限昇格に関する脆弱性の検証を行うこと。

攻撃者は、アプリケーションにアクセスを行った後、権限管理の脆弱性を突いて、一般利用者権限から管理者権限への昇格を試みる可能性がある。権限を昇格させた攻撃者はソフトウェアの完全な制御が可能となる。

【パラメータの改ざんに関するテストを実施する場合】

(4) パラメータの改ざん

- パラメータの改ざんに関する脆弱性の検証を行うこと。

攻撃者は、ソフトウェアと利用者との間で送受信されるパラメータの権限管理や情報の妥当性の検証の脆弱性を突いて、パラメータを改ざんし、ソフトウェアの誤動作を発生させる可能性がある。代表的な例として、電子商取引サイトでの価格の改ざんやパラメータを使ってファイルをオープンするプログラムにおけるパラメータ改ざんによるディレクトリトラバーサル攻撃等がある。

【セッション管理への攻撃に関するテストを実施する場合】

(5) セッション管理への攻撃

- セッション管理への攻撃に関する脆弱性の検証を行うこと。

ウェブアプリケーションでは、アプリケーションレベルでセッション管理を行なっているものがある。攻撃者は、セッション情報の権限管理の脆弱性を突いて、利用者のセッション情報を取得することで、利用者と同レベルの権限でアクセスを行う可能性がある。代表的な例として、利用者とアプリケーションとのセッション情報を盗聴や推測によって強奪するセッションハイジャック等がある。

【コマンドインジェクションに関するテストを実施する場合】

(6) コマンドインジェクション

- コマンドインジェクションに関する脆弱性の検証を行うこと。

攻撃者は、入力データを利用したコマンド処理を実行しているアプリケーションにおいて、情報の妥当性の検証の脆弱性を突いて悪意あるコマンドを混入し、アプリケーションの背後にあるデータベースやシェルなどを不正に操作する可能性がある。代表的な例として、SQL インジェクション、OS コマンドインジェクション等がある。

【クロスサイトスクリプティングに関するテストを実施する場合】

(7) クロスサイトスクリプティング

- クロスサイトスクリプティングに関する脆弱性の検証を行うこと。

攻撃者は、入力データを外部に出力する処理を実行しているアプリケーションにおいて、情報の妥当性の検証の脆弱性を突いて、悪意あるスクリプトを利用者のWeb ブラウザ上で実行する可能性がある。これによって正規の利用者のセッション情報を盗んだり、偽のページを表示させてフィッシング詐欺等に利用することがある。

【バッファオーバフローに関するテストを実施する場合】

(8) バッファオーバフロー

- バッファオーバフローに関する脆弱性の検証を行うこと。

攻撃者は、不適切なデータ長の定義によって生ずる情報の妥当性の検証の脆弱性を突いて、確保したメモリ領域を超えてデータを入力することで、データをあふれさせてプログラムを暴走させ、開発者の意図しない動作を実行させる可能性がある。この代表的な例として、スタックオーバフロー、ヒープオーバフロー等がある。

【エラー処理からの情報の取得に関するテストを実施する場合】

(9) エラー処理からの情報の取得

- エラー処理からの情報の取得に関する脆弱性の検証を行うこと。

権限管理や情報の妥当性の検証の処理において、許可されないデータについては可能な限り入力を拒否し、エラー処理を返す必要があるが、その際、ソフトウェアの内部処理が露呈する可能性がある情報をエラーメッセージとして通知すると、攻撃者は、当該エラーメッセージを利用して、機密情報を取得したり、攻撃に関する有用な手掛かりを得る可能性がある。

≪参考文献≫

- IPA：「脆弱性関連情報に関する届け出状況」
  （http://www.ipa.go.jp/security/vuln/report/press.html ）
  ソフトウェアの脆弱性関連情報に関する最新の届け出状況をまとめたもの。

【セキュリティテストが必要な場合】

6.4　セキュリティテストの計画と管理

問題のある副作用や誤動作が発生しないことを網羅的に検証するためにセキュリティに関するテストでは、適正なテスト計画の準備が極めて重要となる。
また、テストにおいて発見された脆弱性は、速やかに改善されなければならないが、その過程で従来存在していたセキュリティ機能が弱められたり、新たな脆弱性を発生させる可能性があるため、改善の実施状況を適切に管理する必要がある。

(1) セキュリティのテスト項目

- セキュリティテストに当たって、既知の脆弱性やコーディング規約をベースとしたテスト項目を作成し、テストを実施すること。

網羅的かつ効率的なテストのためにはテストのスケジュールや体制、テスト項目や検証手法等に関するテスト計画の作成が重要である。テスト計画には、ソフトウェアのセキュリティ要件に応じて、必要なセキュリティのテスト項目を記載しなければならない。
セキュリティのテスト項目は、ソフトウェアの設計や実装に応じて、既知の攻撃手法やコーディング規約などをベースに作成しなければならない。

≪参考文献≫

- IPA：「消費者向け電子商取引サイトにおける注意点」
  （http://www.ipa.go.jp/security/vuln/20050304_ec_security.html）
  電子商取引サイトにおいて発生しうるけるセキュリティ上の問題点とそれらの対策方法をまとめたもの。
- IPA：「セキュアな Web サーバーの構築と運用」
  （http://www.ipa.go.jp/security/awareness/administrator/secure-web/）
  Web サーバを構築、運用するにあたってのセキュリティ対策の手引き
- Microsoft：「Web アプリケーション セキュリティ強化」
  （http://www.microsoft.com/japan/msdn/security/guidance/secmod71.mspx）
  セキュリティ保護された ASP.NET Web アプリケーションを構築する際のガイドラインを示したもの。

(2) セキュリティテストの管理

- セキュリティテストにおける項目、実施結果、実施時に判明した不具合及び当該不具合の修正内容等を記載した記録を作成し、管理すること。
- セキュリティテストの終了後に、テスタはテスト報告書を作成し、責任者はテスト報告書の内容を検証した上で承認すること。
- セキュリティテストの終了後に脆弱性の修正を行う場合、原因や修正内容などを検討し、関係者に連絡すること。連絡を受けた責任者は、この内容を検証し、妥当と認めた場合にこれを承認すること。
- 責任者の承認を得た後に修正を行なうこと。また、責任者は修正された結果について、再度検証を行うこと。
- テストドキュメント（報告書やテスト用ソフトウェア等）について適切に保護し、保存期間終了後は直ちに廃棄すること。
- セキュリティテストの終了後に、仕様の変更、又は脆弱性の修正をした場合、コードレビューや回帰テスト（リグレッションテスト）を実施すること。

重大な脆弱性を見逃すことがないように、また運用業務において発生するトラブルの原因究明及び保守業務作業に備えるため、テストの実施状況を管理しなければならない。

具体的には、試験項目、実施結果、不具合、修正結果等を記載したセキュリティテストの記録を作成する。作成するテストの記録は、その実施数、正常終了数、修正数などをまとめ、作業の進捗状況を確認する必要がある。

また、セキュリティテストの結果、要件定義書、設計、プログラム等に欠陥が発見された場合は修正を行う必要がある。上流工程において修正が行われた場合は、脆弱性が正しく修正されていること及び修正による副作用（新たなバグの発生）が発生していないことを確認するため、コードの再レビューや回帰テスト（リグレッションテスト）を実施する必要がある。

7　運用環境への移行におけるセキュリティ

【趣旨】

検証と妥当性確認の結果が良好であれば、ソフトウェアを開発環境から運用環境に切り替えることが可能となる。

本項では、検証と妥当性確認を終了した新規開発ソフトウェアを実運用環境に移行する作業に関するセキュリティ面からの留意事項を解説する。

7.1　運用ガイダンスにおけるセキュリティの考慮

運用環境とは、実際の業務が行われている環境であり、移行作業におけるセキュリティ上のリスクは可能な限り極小化される必要がある。このため、開発されたソフトウェアの適切な運用を行うためのガイダンスの準備が必要である。

【運用ガイダンスの作成が必要な場合】

(1) 運用ガイダンスにおけるセキュリティの考慮

- 運用ガイダンスにおいて、セキュリティ機能の特質、設計及び実装を含めたソフトウェアの構成及び機能運用について明確に記述すること。
- 運用ガイダンスにおいて、管理者及び利用者がセキュリティ面で正しい運用を行うことを支援するための操作手順について明確に記述すること。
- 運用ガイダンスにおいて、セキュリティ上の留意事項を含む保守運用体制、インストール及びアンインストール方法について明確に記述すること。
- 運用ガイダンスにおいて、脅威に基づいたセキュリティインシデント定義、インシデント発生時の連絡先及び一次対応手順、予兆検知のためのログの収集・解析方法等を含めた事故・障害の対応手順について明確に記述すること。
- 運用ガイダンスにおいて、セキュリティに関する一般的問い合わせとそれに対する回答（FAQ）について明確に記述すること。

運用ガイダンスとは、ソフトウェアの確実かつ効率的な運用を支援するため、ソフトウェアの機能・設計、操作方法、導入・保守の手順、事故・トラブル対応を説明したドキュメントである。ガイダンスには、セキュリティ面からの考慮事項を併せて記載しておく必要がある。

運用ガイダンスは、管理者向けと一般利用者向けに分けて作成されるのが一般的であり、運用環境への移行前に完成しておくことが望ましい。

7.2　導入におけるセキュリティの考慮

移行作業は切替えの方法によっては、旧システムの情報の変換や利用者側の業務停止等を伴うことがあり、業務継続の観点を重視して慎重な作業を行わなければならない。ただし、業務継続や効率性を優先するあまり、ソフトウェアの導入・移行においてセキュリティが見落とされ、脆弱性が混入されないようにしなければならない。

【導入手順の作成が必要な場合】

(1) 導入手順におけるセキュリティの考慮

- 運搬によって、ソフトウェアの配付を行う場合、安全な配送方法や開梱検出シール等で改ざんの危険性を低減すること。
- 通信によって、ソフトウェアの配付を行う場合、コード署名・ハッシュ関数等で改ざんの危険性を低減すること。
- 本番環境にコンパイルする際、デバッグモードの利用を禁止すること。
- 導入に当たって、開発時に使用した不要な識別子、認証情報、ソースコード、ユーティリティ、テストデータ、サンプルプログラム等を運用環境から除去すること。
- 本番環境への導入前に、アンチウイルスソフトウェア等によって、不正なプログラムが残留していないかを確認すること。
- 本番環境で稼働させる前にセキュリティ機能の設定を完了すること。

開発したソフトウェアを指定された運用環境に配付・移送する際は、*[政府機関統一基準]*に則り、適切な安全管理措置を行う必要がある。
また、導入されるソフトウェアには、テスト効率などの観点から開発者が作り込んだバックドアやデバッグコード、不要なテスト用ツール等が残留している状態であったり、セキュリティ機能が未設定である可能性がある。
このため、導入の事前段階でこうした点について検証した上で、安全な導入作業を行わなければならない。

【移行計画・移行手順の作成が必要な場合】

(2) 移行計画・移行手順におけるセキュリティ

- 開発者の本番環境へのアクセス及び変更作業について、正式な申請・承認・記録の手順を作成すること。
- 本番環境におけるコンパイラ、エディタ等の利用について、正式な申請・承認・記録の手順を作成すること。
- 移行前に既存システムのソースプログラムやマスターファイルなどを記録媒体に記録し、一定期間保存すること。
- 運用に使用しない開発用のデータ、ドキュメント、ソフトウェア（試作品や不良品を含む。）類は、開発終了後、廃棄すること。ただし、以降の保守・運用・復旧作業に必要になる資産については、管理責任を明確化し、適切に保護すること。

移行時には、確実かつ効率的な移行を行うことを目的として、移行対象とする資産、移行に要する期間、要員計画等を明確化した移行計画、及び移行方法、作業

手順、利用者教育等を明確化した移行手順を策定する。この際、セキュリティ面からの管理を十分に考慮しておく必要がある。
具体的には、移行作業は多くの担当者が関与する状況であり、作業の管理が困難であることから、作業についての正式な申請、承認、記録の手順を作成させたり、障害等の予期しないトラブルを想定して、移行前の環境を適切に保存させる等の対策が必要となる。
また、移行を終了し、確認期間を終了した時点で、運用に使用しない開発用の情報、関連する情報資産、及び開発に要する資産は適切に廃棄しなければならない。