Vol.2017-DC-107 No.2
2017/11/30

# IT システム運用手順書における構造化手法の提案

波田野 裕一[†1]

**概要**：社会的重要性を増す IT システム運用において，サービス品質を担保するために必要とされるドキュメント群の中でも運用手順書は利用頻度が高く，その作成・維持に相当の工数を要している．しかし，IT システム運用現場において運用手順書の作成手法が広く確立されているとは言えず，各現場で定められた独自フォーマットに従って各担当者が自己経験に基づいて作成しているのが現状である．そのため，運用手順書の作成に工数が掛かる割には，不適切な手順によるトラブルの発生やその再発を防げないという事象が多発している．
本論では，現場の業務に合致した運用手順書を効率的かつ効果的に作成するために，必要となる要素および作成過程を分析し，適切に構造化するための手法について提案する．

**キーワード**：システム運用，運用手順書，運用ドキュメント，テクニカルライタ

## 1. はじめに

インターネット及びパブリッククラウドの急激な普及及び発展により，多くの IT 情報基盤は社会基盤(インフラ)としての性格を色濃く帯びてきており，IT 情報基盤を提供する企業や担当部署における運用管理業務の維持及び拡大が企業の事業活動に大きく影響を与えるようになってきている．

しかし，日本国内の IT システム運用及び IT サービス運用の現場(以下「運用現場」)においては，要員に対する恒常的な高負荷，属人的な運用，トラブルの多発等により，事業継続性の面からもコストの面からもリスクを内在させ，効率面での課題を抱えつつも，運用現場の個々人の過大な努力により日々の運用を維持しているのが現状である．そして，現状の維持に追われているため，昨今のクラウド普及などの急激な社会及び技術の変化に迅速に対応する余力を有する運用現場は国内ではまだまだ少ない．

運用現場が抱えている問題点の多くは，1.対応負荷が高い，2.属人的で暗黙知が多い，3.費用対効果が見えにくい，という 3 つの要因が複合化した結果発生していると考えられる．そしてこれら 3 つの問題要因を極小化するためには，1.「運用への期待」の明確化，2.「運用設計」の確立，3.「期待に対する消費リソース」の見える化という 3 つの「運用業務の見える化」を行ない，運用現場における業務の「負荷平準化」，「非属人化」，「費用対効果の説明可能化」を実現する必要があると考えられる[1][2]．しかし，「運用現場ごとの個別事情によりやり方が異なるため標準化が難しい」と従来言われてきたそれぞれの運用業務に対して，標準的な「運用業務の見える化」の技法が運用現場には存在しないため，結果として運用現場の「見える化」は個々人の経験と知識に依存している．

このような状況下において，それぞれの運用現場では，日々発生する多様な運用要求に対して一定品質以上の運用成果を提供するために必要となる「非属人化」，及び非属人化がもたらす「負荷平準化」を目的として多くの IT システム運用手順書(以下「手順書」)が作成され利用されている．しかし，運用現場における手順書の作成についても広く確立されている手法が有るとは言えず，各運用現場で定められた独自フォーマットに従って各担当者が自己の経験に基づいた独自の手法で作成しているのが現状である．

本論文は，運用現場における各種の手順書を業務の目的に適合するように効率的かつ効果的に作成するために必要となる要素及び作成過程を分析し，適切に構造化するための手法について論じる．

## 2. IT システム運用手順書における課題

### 2.1 読みにくい手順書

運用現場には多くの手順書が存在し日常の業務において利用されているが，読み手である作業者(以下「作業者」)にとって読みにくくわかりにくい手順書が少なくない．

一例として，作業者にとって読みにくい手順書には以下のようなものがある．

**1. 過少型**

作業を正確に行うには内容や記述が足りない手順書である．以下のようなものが過少型に属する．

- **「自己満足型」手順書**: 作成者がもともとは自分用のメモとして作成したものを，あまり手を加えずに共有したために他人には非常にわかりにくくなっている手順書である．
- **「言葉足らず型」手順書**: 共有用途で作成したがハイコンテキストな記述になっているために，コンテキストを共有している作業者以外には非常にわかりにくい手順書である．

**2. 過剰型**

作業を正確に行うために必要な内容や記述が含まれているが必要以上に冗長な手順書である．以下のようなものが過剰型に属する．

- **「ミス回避型」手順書**: かつて発生した作業ミスを回避・防止するために追加手順や詳細な解説が追加されることにより内容や記述が過剰になった手順書である．

†1 北陸先端科学技術大学院大学

- **「初心者解説型」手順書:** 「誰でも手順を実施できるように」という意識が過剰に働いた結果，初心者向けの解説までもが含まれている手順書である．

過剰型の手順書は一見丁寧で作業者にとって親切に見えるが，その手順書においてどこが重要なのかをわかりにくくし，読み取り工数の増加にも繋がるため作業者にとって読みにくい手順書になるのである．

**3. 複合型**

過少型と過剰型の双方の性質が複合することで作業者に読みにくいと感じさせる手順書である．一つの手順書に多くの作業を詰め込もうとすると過少型もしくは複合型になりやすい．

いずれの型の手順書も独特な読解力を要求するため，読解に時間がかかるだけでなく，作業者の理解や習熟を妨げ，作業上のミスを誘発しやすくなる．これらの手順書は，一定品質以上の運用成果を提供するために作成されたにもかかわらず，その意図に反して作業ミスに繋がってしまうのである．

### 2.2 読みにくい手順書の特徴 ～ 作業者視点から考える

読みにくい手順書を作業者の視点から見ると，一般的に以下のような特徴を有している．

**1. 複雑である**

読みにくい手順書は多くの場合，記述が複雑である．この複雑さには，量的な複雑さと質的な複雑さの 2 種類がある．

量的な複雑さは，1 つの手順書で多くの作業をやろうとして，手順書そのものの分量が多くなってしまうことで生じる．その結果，手順書を読み取るための工数が増大し，手順書が実現したいことの全体像を把握するまでに時間がかかる．

一方，質的な複雑さは，1 つの手順書で特性の異なる複数の作業を同時にやろうとして，手順書から簡潔さが失なわれることで生じる．その結果，手順書で最も実現したいことや個々の手順を行う理由が不明瞭になり，作業に対する深い理解を妨げる．

**2. 曖昧である**

読みにくい手順書は多くの場合，記述が曖昧である．この曖昧さには，ハイコンテキストがもたらす曖昧さと非構造的な文章構造がもたらす曖昧さの 2 種類がある．

ハイコンテキストがもたらす曖昧さは，前提や置かれている状況，作業を行うことによる変化等の重要な記述が省略されることで生じる．その結果，過去に同様の作業をしたことがある作業者には利用可能だが，作業経験が少ない作業者にとっては利用困難な手順書となる．

一方，非構造的な文章構造がもたらす曖昧さは，論理的な整合性の確認が行なわれないことによって生じる．その結果，手順書の到達点が不明瞭になり，確認手順が確認としての実効性を持たなかったために「手順は正常に完了したが，作業の結果としては異常状態を発生させてしまった」という事態を引き起こす．

**3. 作業者保護観点に欠ける**

読みにくい手順書は多くの場合，作業者保護観点に欠けている．欠けている作業者保護観点として，作業ミスの発生を前提とする視点と手順書が作業者に与える負の影響を極小化する視点の 2 点がある．

作業ミスの発生を前提とする視点が欠けていることは，作業ミスを防止する手立てが手順全体に組み込まれず，作業ミスの発生確率の増加及び発生時の影響を大きくすることに繋がる．例えば，あまり一般的ではない省略記法や略語の多用や，一般名詞に特殊な意味を持たせた記述等は，高い頻度で誤読や誤解を引き起こし作業ミスに繋がる．誤読や誤解による作業ミスを意識したレビューを行なっていない場合，複数の手順書で同様の作業ミスが発生する可能性が高くなる．

手順書が作業者に与える負の影響を極小化する視点が欠けていることは，不自然な手順や作業動線等により作業者の注意を散漫にさせ，作業ミスの発生確率の増加及び発生時の影響を大きくすることに繋がる．例えば，手による入力や手順の一部を作業者に読み替えさせる記述，手順書とチェックリストが分離し頻繁に切り替えながら実施する手順等は，作業者の精神的な負荷が高いだけでなく作業ミスにも繋がりやすい．

### 2.3 読みにくい手順書が作られる理由 ～ 作成者視点から考える

読みにくい手順書が作成される背景として，多くの場合，作成者に以下のような理由が存在する．

**1. 作成のために確保できる時間が足りない**

多くの運用現場は日常業務に忙殺され，かつギリギリの人員で業務を回していることが少なくない．そのような中で，業務時間内に手順書を含む文書の整備に割ける時間はわずかである．

**2. 作成したことが評価されない**

多くの運用現場では，手順書を作成することよりも実作業を数多くこなすことの方が評価される傾向にある．そのため，手順書を個々人で作成・管理しているケースや，共有されているとしてもあくまでもボランティアベースであり正規の業務としては評価されていないケース等もある．

1 の「作成のために確保できる時間が足りない」ことが起こるのは，「作成したことが評価されない」ことにより，手順書を作成する時間を確保するための名分が立ちにくいことも大きな要因となっている．

**3. 作成が得意ではない**

多くの運用現場では，手順書作成について深い見識と経験を有している人材が少ない．これは，国内の業務教育が OJT に深く依存しており，文書作成について正規の教育を受けていないためであると考えられる．

また，2 の「作成したことが評価されない」ことにより，文書作成能力を伸ばそうという動機が運用現場には存在しないことも，文書作成に精通した人材が増えない要因となっている．

## 3. IT システム運用手順書に求められる 3 つの要件

読みにくい手順書が氾濫している現状から脱却するためには，手順書に求められる要件を明確にする必要がある．本論では，「2. IT システム運用手順書における課題」における課題分析から，以下の 3 点が手順書に求められる要件だと捉える．

**1. 要件 1: 作業者が読みやすい**

手順書は，一定品質以上の運用成果を提供することを目的として作成される文書であることから，運用成果の品質に最も影響を与える作業者にとって過不足なく読みやすい文書であることが手順書の最初の要件となる．

これはごく当たり前の要件に見えるが，実際の運用現場の手順書は作成者の都合が優勢になりがちで，作業者が隠れた工数を負担していることが少なくないのが現実である．

**2. 要件 2: 作成者が書きやすい**

手順書は，一定品質以上の運用成果を提供することを目的として作成される文書であることから，新規に生まれる業務に迅速に対応し，時代や状況の変化に適切に対応した更新が行なわれることが常に求められる．これを実現するためには，その内容に大きな影響を与える作成者にとって書きやすい文書であることが手順書の 2 つ目の要件となる．

**3. 要件 3: 運用現場にとって価値がある**

作業者にとって読みやすく，作成者にとって書きやすい手順書であっても，運用現場が求められる価値や目指す価値を実現することに貢献していなければ，手順書の活用による効果は限定的であり，作成や維持に必要となる工数を確保することもできない．

運用現場にとっての価値の一例として，以下の 2 つの種類がある．

**サービス価値**

運用現場が，顧客や他の組織，部署等に対して提供している価値である．一般的に運用現場は，何らかの課題解決を直接・間接的に行うことで外部に対して価値を提供している．運用現場のサービス価値を一定品質以上で提供することに貢献することが手順書の価値の源泉となる．

**エンジニアリング価値**

運用現場が，サービス価値を継続的，反復的に提供する能力であり，その能力が産み出す複合的な価値である．仮にサービス価値が高くとも，継続的・反復的に提供できなければ，運用現場の事業継続性を確保することができない．運用現場のサービス価値を反復的及び再現的に提供することに貢献することで手順書の価値は増大する．

上記のサービス価値及びエンジニアリング価値のように運用現場にとって何らかの明確な価値に貢献することが手順書の 3 つ目の要件となる．

## 4. 「作業者に効果的な IT システム運用手順書」を実現するために必要な要素

手順書に求められる 3 つの要件のうち「要件 1: 作業者が読みやすい」及び「要件 3: 運用現場にとって価値がある」を満たし，作業者にとって効果的な手順書を実現するためには以下の要素が必要となる．

### 4.1 平易であること

どんな作業者が手順書を利用するかの想定を明確にし，手順書が作業者にとって平易であることが求められる．作業者を特定することは，一般的な文書で求められるいわゆる 5W1H のうち Who(誰)を明確にすることに該当する．

想定される作業者にとって複雑もしくは曖昧な手順になることを避け，手順書が平易になるためには以下のような配慮が必要となる．

**1. 平易な記述**

作業者のレベルにあわせた平易な記述を心掛ける．一文を長くし過ぎず，主語や述語を把握しやすいように配慮する．

**2. 平易な語彙**

作業者のレベルにあわせた平易な語彙を利用する．必要以上に難しい言葉の使用を避け，作業者が混乱や誤解をしないように配慮する．

**3. 書き過ぎない**

平易に書こうとするとつい詳細に記述したくなるが，長い文は作業者の読み飛ばしを誘発するため，書き過ぎないように配慮する．これは，量的な複雑さを回避するためにも重要な視点となる．

### 4.2 単純であること

手順書には，作業に必要な要素が不足なく記述されているとともに，想定される作業者にとって適切に単純化されていることが求められる．5W1H のうち When(いつ)，Where(どこで)，What(何)を，Why(なぜ)，How(どうする)を明確にすることは，作業に必要な要素を不足なく記述することに繋がる．

想定される作業者にとって複雑もしくは曖昧な手順になることを避け，手順書が単純になるためには以下のような配慮が必要となる．

**1. 手順目的の単純さ**

手順書には，その手順の主目的が何であるかを可能な限り単純かつ明確にすることが求められる．

目的を明確にすることは，5W1H のうち Why(なぜ)を明確にすることに繋がる．手順書における多くの Why は，その作業全体の目的のために生じるからである．また，手順書の目的を明確にすることにより，その手順書の位置付けや中心的な価値が明確になり，手順書の存在価値及び利用頻度の向上に繋がる．

**2. 手順対象の単純さ**

手順書には，いつ，どんな場面で，何を，どうするための手順なのかを可能な限り単純かつ明確に特定することが求められる．

この観点からは，1 つの手順で複数のことをやらせないことが重要となる．複数の When(いつ)，Where(どこで)，What(何)を，How(どうする)が混在することは複雑さを増し，作業者にとって理解しづらい手順書になるためである．

手順書の対象の単純さを保つために，以下の配慮が必要となる．

- When は 1 つ (時間的に断絶した複数の作業を 1 つの手順書にまとめない)
- Where は 1 つ (複数の場所や場面での作業を 1 つの手順書にまとめない)
- What は 1 つ (複数の作業対象に対する作業を 1 つの手順書にまとめない)
- How は 1 つ (作成，変更，削除等の影響が大きい作業を 1 つの手順書にまとめない)

**3. 手順フローの単純さ**

手順書には，分岐せずに上から下に流れるように記述することが求められる．

手順途中での条件分岐や，以前の手順に戻ることを促す記述等は，手順書を複雑にして作業者を混乱させるため極力避けるべきである．

### 4.3 親切であること

手順書は，運用成果を提供するための活動を実際に担う作業者に対して，親切であることが求められる．

**1. 全体を把握しやすい工夫**

手順書には，全体を把握しやすい工夫が求められる．

例えば，作業単元の区切り方と見出しの付け方を工夫する等，自然な流れを表現することが求められる．見出しに規則性を持たせることは作業の理解しやすさを促進する上で効果的である．

**2. 現在位置を見失わない工夫**

手順書には，現在どの位置の作業をしているかを見失わないための工夫が求められる．

見出しに番号を付けることは現在位置を見失わない有効性の高い方法である．

**3. 作業者視点によるレビューの反映**

手順書では，作業者はだいたい同じような場所でとまどい工数増や作業ミスを発生させるため，作業者視点でのレビューを繰り返すことで，実際に工数増や作業ミスが発生する前に該当部分の解消を図ることが求められる．

作成後や更新後，一定時間を置いてから改めてレビューすることや，運用現場に配属された新人にレビューを実施させることで新たに修正すべき記述を発見することも多い．

### 4.4 思いやりがあること

作業者保護の観点から，手順書には作業者に対する思いやりが込められていることが求められる．

「平易であること」，「単純であること」，「親切であること」の 3 つについては技術論的な観点から実現は難しくないが，作業者保護の観点についてはやや精神論になるが，読みにくい手順書による苦労や作業ミスによる痛みを知る者の視点が不可欠である．

利用する作業者が同様の苦労や痛みを経験しないようするための思いやりが，手順書を読みやすく誤解を生みにくい高品質のものへと向上させていくのである．

## 5. 「作成者に効率的な IT システム運用手順書」を実現するために必要な要素

手順書に求められる 3 つの要件のうち「要件 2: 作成者が書きやすい」及び「要件 3: 運用現場にとって価値がある」を満たし，作成者にとって効率的な手順書を実現するためには以下の要素が必要となる．

### 5.1 書くことの価値が明確であること

作成者が手順書を作成する工数を確保し，積極的に手順書の作成を継続していくためには，手順書を書くことによって実現される価値が明確であることが求められる．ここでの「価値」には，「3.3. 要件 3: 運用現場にとって価値がある」で言及した「サービス価値」や「エンジニアリング価値」等が含まれる．

実現すべき価値が明確であれば，手順書に記述すべき内容は価値実現に必要かつ十分なものに限られ，手順書の対象範囲及び作業完了時に何を実現していれば正常な完了なのかが早期に確定するため，効率的な手順書作成が可能となる．

その結果，手順書の作成に要する工数が適正化されるとともに手順書の品質が向上することで，手順書の作成業務自体の価値の向上及び作成者の評価に繋がる．

### 5.2 構造的であること

本論文において「構造化」とは，「ばらばらになっているものを，一定の基準に従い論理立てて整理すること」と定義する． [3]

構造化による効果は以下の 3 点が考えられる．

**1. 作成者にとって手順書を書きやすくなる**

構造的に手順書を記述するには，作業対象を構造的に捉えなおさなければならないが，構造的に捉えなおした対象を構造的に記述することは容易となる．

**2.　作業者にとって手順書を読みやすくなる**

構造的に記述された手順書は作業対象を構造的に理解することを可能とし，作業者にとって作業対象の全体及び細部の把握を容易にする効果がある．

**3.　構造化されることによって構造分析が可能になる**

構造化された手順書においては，似た構造を持つ手順書や個別の手順として使いまわされている記述部分を抽出して類型化することが容易になる．抽出し類型化した手順を継続的に分析，改善することで，その運用現場における手順書の特徴を把握し，特徴に即した高品質の手順書を効率的に整備することが可能となる．

### 5.3 再利用できること

構造化した手順書から抽出した記述部分は，テンプレート化して再利用することで新規手順の作成工数を抑制することが可能となる．

例えば，インクルード機能を有するドキュメントプロセッサを利用した場合は，一度記述した作業手順部分については以後同様の記述をする必要がなくなる．また，キーワード置換機能を有するドキュメントプロセッサを利用した場合は，特定のキーワードについて作業当日の値に置換することで，特定の作業に特化した手順書を即時に生成することが可能となる．

このように，再利用可能な手順書を整備することで，手順書作成にかかる作成者の工数を最小化しながら高品質な手順書を大量に作成できるようになる．

## 6. IT システム運用手順書の作成過程

これまでに洗い出した要素を踏まえ，作業の目的に適合するように手順書を効率的かつ効果的に作成するには，その手順書全体が「5.2. 構造的であること」に従って構造的に構成及び記述されていることが必要であり，「5.3. 再利用できること」を意識した構成や記述になっていることが望ましい．

そして，構造化され再利用可能な手順書を作成するための過程については，概ね以下のような流れになると考えられる．

### 6.1 手順書が提供する価値の決定

まず，「5.1. 書くことの価値が明確であること」を満たすための作業を行う．手順書の価値を予め明確にすることで，手順書作成の工数を確保し，成果物としての手順書に対する期待値を適切に制御することができる．

**1.　作業テーマの決定**

最初にその手順書がどのような価値を実現するかの概略を明示する．

例えば，サービス価値であれば，その手順によって顧客や他の組織，部署等に対してどのような価値を提供するのかを明確にする．エンジニアリング価値であれば，その手順によって継続性反復性などがどのように向上するのかを明確にする．

そして，手順書の作業対象を特定する名詞と手順書の価値を組み合わせ，誰が見てもどんな価値がある手順書なのか理解しやすい名称に落とし込むことで，手順書が対象とする作業の主題（以下「作業テーマ」）が決定する．

**2.　作業スコープの決定**

次に，決定した作業テーマを基に，手順書が実際に対象とする範囲（以下「作業スコープ」）を明確にする．

作業スコープは，When（いつ），Where（どこで），What（何）の 3 つの要素で決定する．

作業スコープの各要素は「4.2. 単純であること」の「手順対象の単純さ」で記述した通り，以下の観点で単純であることが求められる．

- 1 つの When （時間的に断絶した複数の作業を 1 つの手順書にまとめない）
- 1 つの Where （複数の場所や場面での作業を 1 つの手順書にまとめない）
- 1 つの What （複数の作業対象に対する作業を 1 つの手順書にまとめない）

作業スコープを決定することにより，手順書に記述すべき内容は作業スコープの範囲に限られるため，効率的な手順書作成へと繋がる．

また，作業スコープの決定は「4.2. 単純であること」の「手順対象の単純さ」を実現し，作業者にとって理解しやすい手順書になるという効果が得られる．

### 6.2 実際の作業内容の決定

手順書が提供する価値が決定した後は，「4. 『作業者に効果的な IT システム運用手順書』を実現するために必要な要素」の 4 つの要素を意識しながら，作業者にとって効果的な手順書を実現していく．

**1.　事前条件・事後条件の決定**

作業者保護の観点から作業開始前に想定されている状態と作業完了後に想定されている状態を予め決定しておくことは，手順書から論理的な矛盾を排除する上でも非常に有効である．作業内容の決定においては，最初にこの事前条件及び事後条件を決める．

本論では，ホーア論理の「事前条件が成り立っている状態でプログラムを実行して，もし停止すれば事後条件が成り立つ」[4]という部分正当性の性質を参考に，手順書に「事前条件」「事後条件」を導入する．ここで，「プログラムを実行して，もし停止すれば」の部分は，「作業者が作業を実行し，混乱やミスをせずに手順を完了すれば」と置き換えて適用する．

**事後条件**

手順書の完了時に何が実現されていれば正常なのかを記述する．事後条件を明確にすることで，「手順は正常に完了したが，作業の結果としては異常状態を発生さ

せてしまった」という事象の発生を避けることが可能となる．

**事前条件**

手順書通りに正確に作業を進めて，事後条件を満たすためには，正確な事前条件を記述する必要がある．事前条件の例として，操作対象リソースの状態や存在もしくは不存在，操作権限の有無などがある．

事前条件，事後条件を明確にすることにより，手順書には事前条件と事後条件の差異を埋めるために必要かつ十分な作業だけを記述すれば良いことになるため，副次的な作用として「4.2. 単純であること」の実現にも繋がる．

**2. 作業前提の決定**

手順書が想定する作業者，事前条件と事後条件を満たすために必要となるリソースなど作業を円滑に進めるための前提を明確にしていく．

**作業者**

「4.1. 平易であること」に従い，手順書をどんな作業者が利用するかを明確にする．これにより，作業内容に記述すべき作業手順の内容や記述が決まる．

**作業に必要なリソース**

「4.1. 平易であること」に従い，作業の事前条件と事後条件を十分に満たすために必要となるリソースを明確にする．

作業に必要なリソースの例としては，作業環境(OS，ブラウザ，ホスト名，利用アプリケーションのバージョン)などや操作対象のリソース(ホスト名，サービス名，機器名，アカウント名，操作対象の要素名)などがある．

**3. 実施する作業の階層化**

「4.2. 単純であること」に従い，実際に行う作業としてHow（どうする）を階層化していく．

多くの手順書では，1 つの手順書内で作成，変更，削除等の影響が大きい作業が複数行なわれている．このことは「4.2. 単純であること」の「手順対象の単純さ」に反しているため作業者にとってはわかりにくい手順書となりやすい．しかし，実務的にはこれら影響が大きい作業が同居することは避けられないのも事実である．

この相反する要求に対応するためには，作業を階層化することが有効である．本論では，手順書における作業の階層として，手順書と同じ粒度の「作業」と作業を細分化する「タスク」という 2 階層で階層化することとする．

タスクには，「4.2. 単純であること」に従った以下の特性が「作業」よりも厳密に要求される．

- 1 つの When （時間的に断絶した複数の作業を 1 つの手順書にまとめない）
- 1 つの Where （複数の場所や場面での作業を 1 つの手順書にまとめない）
- 1 つの What （複数の作業対象に対する作業を 1 つの手順書にまとめない）
- 1 つの How （作成，変更，削除等の影響が大きい作業を 1 つの手順書にまとめない）

「タスク」の設計においては「一つのことを上手にやってもらう」という視点が「4.2. 単純であること」を実現する上で極めて重要となる．

また、タスクをテンプレート化することで，特定のタスクを複数の作業で共有することができるようになり，「5.3. 再利用できること」の実現に繋がることになる．

階層化には多くのパターンがあるため，一つの手順書（1 作業）で実施するタスクが 1 つになる場合もある．

**6.3 タスク内容の決定**

タスクは作業のサブセットとなるため，タスクの内容をは作業に求められる以下の項目の内容をより限定したものとなる．

- 作業の目的 （Why）
- 作業対象 （What）
- 想定する作業者 （Who）
- 作業に必要なリソース
- 事前条件
- 事後条件
- 作業環境条件 （Where）
- 作業開始条件 （When）

ここには，ホーア論理に準じた事前条件，事後条件，作業に必要なリソース及び作業に関する 5W が明示されているため，タスク内容の決定に必要な最後の 1 点は How（どうする）の設計だけとなる．

タスク内容の How（どうする）を決定するには，以下の 2 つの設計が必要となる．

**1. タスクにおけるパイプラインの設計**

運用を「サービスデリバリ」と捉えた場合，運用への作業要求が何らかの形で外部から入力された時点でタスクが誕生し，その作業成果を要求元へ出力した時点でタスクが完了すると考えられる．[4] このタスクのライフサイクルを，最も単純なパイプラインとして表現すると以下のようになる．

[入力] - （前処理） - 主処理 - （後処理） - [出力]

タスクにおいては，作成，変更，削除等の影響が大きい作業は主処理で行ない，主処理のために必要なチェックを前処理や後処理で行なう．前処理の例としては，入力の形式検証や実施に必要な承認などがあり，後処理の例としては，出力前の形式検証や作業記録の保存などがある．ホーア論理に準じる場合は，前処理では事前条件を満たしていることの確認と主処理に必要な補助的な手順，後処理では事後条件を満たしていることの確認と出力に必要な補助的な手順がそれぞれの内容となる．

**2. 動作の設計**

パイプラインの設計後は,各パイプラインを構成する「動作」を設計していく.

ここで,「動作」とは,作業における最小単位であり,1つの入力と1つの出力の組み合わせである.

例えば,コマンドラインインタフェース（CLI）においてはコマンド入力とその実行結果の組み合わせが1つの動作となり,グラフィックユーザインタフェース（GUI）においては,マウス操作とその操作結果の組み合わせが1つの動作となる.

これらの動作をテンプレート化することで,特定の動作を複数のタスクで共有することができるようになり,「5.3. 再利用できること」の実現に繋がることになる.

### 6.4 作成及び公開

作成する手順書の価値,作業及びタスクの内容の決定により「4.2. 単純であること」に求められる要素の多くと作業者保護の観点の一部を手順書に反映することが完了する.

次は,「4.1. 平易であること」及び「4.3. 親切であること」を意識して記述していくことで,手順書の細部を作り込んでいく.

手順書の完成後は,実際に利用を開始するまでに以下を実施することになる.

**1. 作業標準時間の決定**

手順書が想定する作業者がその作業を実施した場合に,実際に必要とする作業標準時間を算出し,手順書に明記する.

作業標準時間は,作業者の負荷や習熟度を測定する上での指針となるだけでなく,手順書によって実現される運用成果をQCDで評価するときのD（Delivery）の値として利用することが可能である.

**2. レビュー**

作業標準時間が決定した後,整合性チェックのための机上レビューや実作業による実地レビューを行い,手順書の精度を向上させていく.このときに「4.4. 思いやりがあること」を意識したレビューを実施することで,手順書の品質を大きく向上させることができる.

**3. 公開**

レビュー完了後,手順書の公開を行う.

手順書の公開時には,成果物の完成と利用開始を関係者に広く明示的に伝えることが重要となる.公開された手順書の利用を促進し,期待を上まわる手順書の価値を伝えることで,引き続き手順書作成の工数を確保することが可能となる.

## 7. ITシステム運用手順書を適切に構造化するための手法の提案

作業の目的に適合するように手順書を効率的かつ効果的に作成するために,6章で整理した作成過程を踏まえ,本論では以下の構造化手法を提案する.

### 7.1 Step1:手順書の概要設計

手順書の概要設計においては,手順書の価値と前提を決定する.

**1. 手順書の価値設計**

「6.1. 手順書が提供する価値の決定」に従い,作業テーマと作業スコープを決定する.作業テーマを決定することにより,手順書の作業がもたらすサービス価値(ビジネス的な意味など)や,手順書化することによるエンジニアリング価値(定型化する意味など)を明確にする.作業スコープを決定することにより,手順書の記述範囲を限定する.

**2. 手順書の条件設計及び前提設計**

「6.2. 実際の作業内容の決定」に従い,事前条件・事後条件と作業前提(作業者,作業に必要なリソース)を決定する.重要な事前条件,事後条件や想定している前提を限定することにより,作業に関する記述の範囲を限定し,論理的な矛盾が発生するリスクを低減する.

### 7.2 Step2:手順書の詳細設計

手順書の詳細設計においては,作業を階層化してタスクに細分化し,各タスクについて条件設計,前提設計,パイプライン設計を行う.

**1. 作業の階層化(タスク)設計**

「6.2. 実際の作業内容の決定」に従い,実施する作業の階層化を行い,作業をタスクに細分化していく.各タスクについて,その目的(Why),作業対象(What),インプット,アウトプットを決定していく.各タスクに実施の前後関係がある場合は,明確化しておく必要がある.

設計した各タスクについては,個別に条件設計,前提設計,パイプライン設計を実施する.

**2. タスクの条件設計**

タスクの目的(Why),作業対象(What),インプット,アウトプットに基づいて,各タスクの事前条件及び事後条件を決定する.1作業1タスクの場合は,手順全体の事前条件・事後条件と同一の内容となる.一般的に,目的に即して事後条件を先に決定し,事前条件を後に決定する方が,各条件の精度が高くなり手戻りが少ない.

**3. タスクの前提設計**

タスクが想定する事前条件と事後条件を満たすために必要となる以下の前提を明確にしていく.

- 作業者 (Who): 手順書全体の作業者と同一になる.タスクを複数の手順書で再利用する場合は,スキルや権限に基づく作業者像を決定する.
- 作業に必要なリソース: 事前条件と事後条件を満たすために必要となるリソースを決定する.
- 作業環境条件 (Where): タスクが想定する作業環境を決定する.

- 作業開始条件 (When): タスクの開始条件を決定する．開始条件の例として時間の到来や何らかの指示などがある．

1 作業 1 タスクの場合は，いずれも手順全体の各前提と同一の内容となる．

**4. パイプラインの設計(前処理・後処理)**

タスクのパイプライン設計においては，まず事後条件を満たす後処理を設計する． これにより，「手順は正常に完了したが，作業の結果としては異常状態を発生させてしまった」という事象の発生の防止を手順面で担保することが可能となる．

次に, 事前条件を満たす前処理を設計する．これにより，事前条件と事後条件の差分が手順面で明確になる．

**5. パイプラインの設計(主処理)**

タスクにおいては，作成，変更，削除等の影響が大きい作業は主処理で 1 つだけ実施する． 主処理においては，作成，変更，削除そのものの作業だけを記述し，付随する確認作業などは全て前処理もしくは後処理に移動することが望ましい．

主処理の内容を単純にすることは，手順における重要度が際立つことにより理解を促進し，読みにくさをもたらさないことによりミスの回避へと繋がる．

**6. タスクの前提の精緻化・レビュー**

タスクの前提設計及び条件設計を見直してタスク設計を精緻化し, レビューによりタスク全体の整合性を確認する．

### 7.3 Step3:手順書の品質検査

手順書の品質検査においては，作業の概要設計を見直して作業設計の精緻化し，机上レビューにおいて全体的な整合性を確認し，実地レビューにおいて検証環境で実際に手順書の整合性や正確性を確認する．

### 7.4 Step4:手順書のリリース

手順書の公開においては，リリースノートを作成し，手順書とリリースノートを関係者に同時に公開する．手順書は，公開とともに利用が開始され，運用成果を一定品質以上かつ効率的に提供することに継続的に貢献することになる．

### 7.5 手順書の更新

構造化された手順書は，どこで不整合が発生し，どこを修正することで整合性を回復できるかを比較的容易に発見しやすい．利用開始となった手順書に不整合が発見された場合は，作業者から作成者にフィードバックし，迅速な更新を行うことで時代や状況の変化に適切に対応することが可能となる. このことは, 手順書の利用可能期間を長くし，継続的に手順書の信頼性を向上させることに繋がる．

## 8. 効果の見通し

本章では，2 章における課題分析を基に 3 章乃至 7 章で提案した各要件，各要素，作成過程及び手法の効果についての見通しを示す．

読みにくい手順書が氾濫している現状から脱却するためには以下の 3 つの要件を満たす手順書を作成する必要があることを 3 章において示した．

- 要件 1: 作業者が読みやすい
- 要件 2: 作成者が書きやすい
- 要件 3: 運用現場にとって価値がある

要件 1 及び要件 3 を満たして作業者にとって効果的な手順書を実現するために必要な要素を 4 章において整理した. 要件 2 及び要件 3 を満たして作成者にとって効率的な手順書を実現するために必要な要素を 5 章において整理した. 4 章及び 5 章で整理した要素を基に，6 章において手順書の作成過程を整理した．

以上から，6 章の作成過程に従って手順書を作成することにより，作成者個々人の独自の手法や経験に過度に依存することなく，3 章の 3 要件を概ね満たす手順書を実現することができるはずである．

## 9. おわりに

本論では，IT システム運用手順書の作成における最低限の構造化を実現するための手法を提案したが，更に精緻な構造化を実現するための手法が今後は求められていくと考えられる．

例えば物理作業とターミナル作業，GUI 作業ではそこで利用されている語彙が異なると考えられることから，手順書に求められる語彙のカタログ化や手順書への適法方法が求められていくと考えられる．また，論理構造を人力に頼らずに解析器を活用した構造設計により手順書を作成する手法も求められていくと考えられる．本論がその発展の一助になれば幸いである．

また，本論においては先行事例の研究がほとんど欠けていると言わざるを得ない．今後先達の皆様のご鞭撻をいただき発展させていきたいと考えている．

## 参考文献

[1] 波田野裕一. 商用サービス運用の現状と課題費用対効果との闘い. CTC アカデミックユーザーアソシエーション ViewPoint. 2011, vol. 11, p. 55-56.
[2] “現場視点からの運用方法論 第 1 回 見えない「運用」- 疲弊する運用現場”. https://thinkit.co.jp/story/2010/12/02/1903, （参照 2017-11-06）.
[3] 波田野裕一. 運用ドキュメントの分類と構造化. 電子情報通信学会技術研究報告. 2012, vol. 111, no. 375, p. 9-14.
[4] 萩谷昌己. 「コンピューティング」('15). 放送大学教育振興会 2015, 106p.
[5] 波田野裕一. 運用方法論の研究について. 経営情報学会 全国研究発表大会要旨集. 2009, f(0), p. 42-42.