

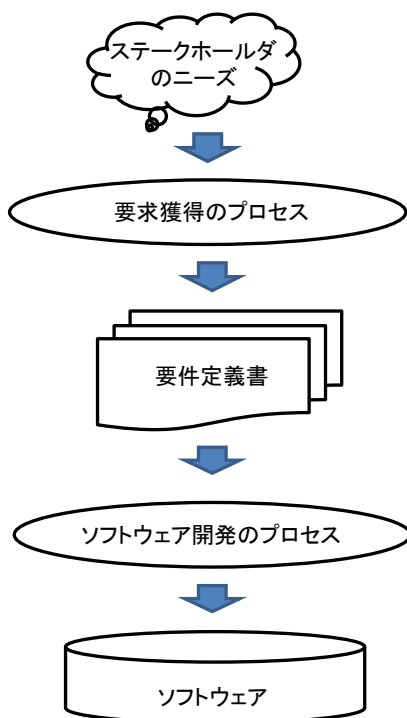
第 21 章 要件定義書の作成

要件定義書とは

ソフトウェアの開発で、要件定義書（SRS : Software Requirement Specification）はその最初の段階で作成される非常に重要な文書である¹。要件定義書の作成法についてすでに多くの優れた本が出版されているが、それらの本の著者の 1 人である清水吉男氏は、要件定義書について次のように述べている。

「これから開発するソフトウェアの『作業のゴールとしての要件』を明らかにするものであり、顧客や開発関係者間でこれから作るものについて合意するための文書である[SIM05]。」

この要件定義書によって、それまで曖昧模糊としていた情報システムに対するステークホルダ（利害関係者）の要求が明確にされ、文書化される。そしてこの文書を元に、これから開発する情報システムの詳細が決定されてゆくことになる。この関係を、図表 21-1 に示す。（図表 21-1 は、図表 6-1 で示したものと同一のものである。）



図表 21-1 要件定義書の位置づけ（[SAN94]を元に一部修正）

現時点での「品質」についての国際的な公式の定義は、2015年版のISO 9000の規格（ISO 9000 : 2015）にある[JIS15a]。それによると品質とは、「本来備わっている特性の集まりが、要求事項を満たす程度」とある。いかにも国際規格らしいたいへん難しい表現だが、品質に関

¹ 「要件定義書」に似た文書に、「要求仕様書」がある。要求仕様書と要件定義書の違い、要求仕様書の作成については、第 20 章で記した。

わるオーソリティの一人であるフィリップ・B. クロスビー (Philip B. Crosby) は、これを端的に「ユーザの要求への適合度」と表現している[CRO79]。つまり、「ユーザのニーズに適合している割合が高い製品ほど、その製品の品質が高い」というわけである²。

ここでクロスビーは「ユーザ」という言葉を使用している。しかし最近はこの「ユーザ」という言葉に代えて、前述のように「ステークホルダ」、または「利害関係者」という表現が使われる。ユーザはステークホルダの一部であるが、全てではない。しかし議論を簡単にするために、以下では「ユーザ」という言葉をそのまま使い続けることにする。

ソフトウェアは「ユーザの要求により多く適合しているほど品質が高い」訳であるから、図表 21-1 を基に考えると、まず高品質のソフトウェアを作るためには、「要求獲得のプロセス」で曖昧模糊としたユーザ (利害関係者) のニーズを的確に把握し、その結果を明確に要件定義書に記述することが必要である。その上でその要件定義書に記述された内容を、「ソフトウェア開発のプロセス」を通してソフトウェアに仕上げてゆくことになる。このように見ると高い品質のソフトウェアを作る上で、要件定義書はたいへん重要な文書であることが分かる。

「要求工学」という言葉

この曖昧模糊としたステークホルダの要求を過不足無く的確に引き出し、適切に文書化し、レビューするということは、容易なことではない。高いレベルの技術が要求される、難しい作業の 1 つである。

図表 21-1 で「要求獲得のプロセス」と呼んだこの領域をカバーする仕組みを、ソフトウェア工学の立場からは「要求工学 (Requirement Engineering : RE)」と呼んでいる。「ソフトウェア工学」という 1 つの「工学」の範囲内に「要求工学」という別の名前がついた「工学」があることになる訳だが、それだけこの分野が重要であることを示している。

仮に今我々が対象としている情報システムをビジネス・アプリケーションとすると、ステークホルダの要求を獲得する方法はもっぱらインタビューと、ステークホルダが作業を行っている現場を見学し、さらに質問することである³。それ以外にビジネス・アプリケーションでは、法律や業界の取り決め、慣習などがステークホルダの作業の前提になっているのが普通であるから、それらの調査も欠かすことができない。要求の獲得方法については、後述する。

さらにビジネス・アプリケーションの話が続けるが、要件定義書でステークホルダのニーズを明らかにするに当たって、まず仕事の流れを明確にし、コンピュータと人間が作業を分担する場合にはどこまでを人間が行い、どこからをコンピュータが行うのかの作業の切り分けが不可欠である。

最近のビジネス・アプリケーションでは基本的に全ての作業をコンピュータとネットワークで行い、人間は今のコンピュータが行うことができないところだけを限定して行うというスタンスの情報システムが増えてきている。RFID やウェアラブル・コンピュータの普及などで、ビジネス・アプリケーションでのこのコンピュータとネットワークの領域が今後一層広がるのかもしれない。

要求工学知識体系

² 品質については、既に第 5 章で述べた。

³ ステークホルダが「要求仕様書」を作成している場合、これは非常に重要な要求獲得の手段となる。

この要求工学にどのような知識とスキルが含まれ、要求獲得のプロセスにはどのような作業があるのか、といったことを網羅した 1 つの標準が日本で誕生した。それを「要求工学知識体系 (REBOK : Requirement Engineering Body of Knowledge)」と呼ぶ⁴[JISA11]。

その REBOK によると、要求には 3 つのレベルがあるという。それを上から挙げると、以下のようになる。

- ① ビジネス要求
- ② システム要求
- ③ ソフトウェア要求

これまで要件定義書として作成されていたものは、3 つ目のソフトウェア要求であった。しかし最近の傾向として、前の 2 つも重要との認識が広がっている。これについても、後述する。当面「要件定義書」という言葉で表されるものは「ソフトウェア要求」を指すものとして、話を進める。

要件定義書作成についての標準

要件定義書の作成について、IEEE が 3 つ標準を持っている⁵。その 1 つが、IEEE Std 830-1998 である[IEEE98f]。

この標準によると、要件定義書には次の事項を記載しなければならないとしている⁶。

- 機能
- 外部とのインタフェース (人間、ハードウェア、他のソフトウェアとの)
- パフォーマンス (スピード、アベイラビリティ、レスポンスタイム、復旧時間、など)
- アトリビュート (移植性、正確さ (コレクトネス)、保守の容易性、安全性、など)
- 設計上の制約

この最初の「機能」に関する要求を「機能要求」、それ以外のものを「非機能要求」と呼ぶ。つまり要件定義書には「機能」に関する要求に加えて、機能に関わる要求以外のものも「非機能要求」として記載しなければならない。

ここで、「何が機能か」ということについての議論がある。ここでは、機能 (Function) とは「入力を出力に変換すること」と単純に定義し、この定義に基づいて上記の機能要求と非機能要求の区分けを行っている。

さらにこの標準では、要件定義書は以下の要件を満たさなければならないとしている[IEEE98f]。

- 正確であること (Correct)。
- 曖昧でないこと (Unambiguous)。
- 完全であること (Complete)。
- 首尾一貫していること (Consistent)。
- 重要さと安定性のためにランク付けされていること (Ranked for importance and/or

⁴ REBOK は、今はまだ国際標準として完全に認められる段階にはない。しかしそれを目指して、がんばってほしい。

⁵ 要件定義に関わる IEEE の 3 つの規格 (IEEE Std 830-1998、IEEE Std 1233-1998、IEEE Std 1362-1998) は、廃止されてしまった。

⁶ この IEEE の標準には、何故かデータ量などのボリュームについての記載が求められていない。

stability)。

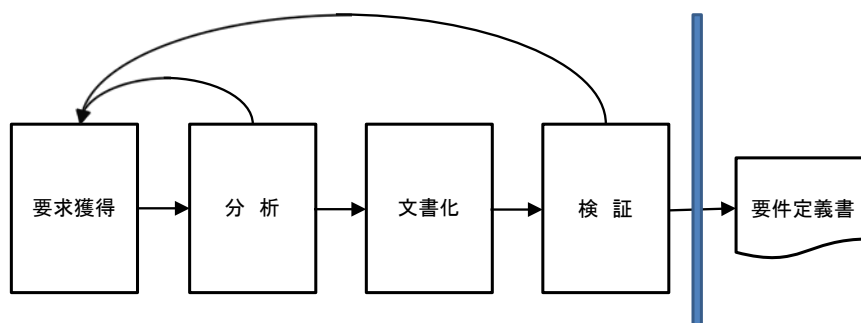
- 検証可能であること (Verifiable)。
- 修正可能であること (Modifiable)。
- 追跡可能であること (Traceable)。

その上でこの標準は、要件定義書のプロトタイプを用意しており、さらにその中の詳細な要求仕様の部分の書き方について 8 種類のひな形を用意している。全体のプロトタイプは、この章末に付 1 として添付する。

ひな形についての詳細の記述はここでは省略するが、あるものは構造化技法時代からのオーソドックスな書き方であり、あるものは新しいオブジェクト指向流のものである。以下で、それをベースにした内容の紹介を行う。

要件定義書作成の作業

シャリ・ローレンス・フリーガ (Shari Lawrence Pfleeger) はその著書の中で、図表 21-2 に示す図を掲載して要件定義書作成の作業を説明している [PFL09]。



図表 21-2 要件定義書作成の作業 ([PFL09]より)

それによると、要件定義書作成作業は以下の 4 つのフェーズで構成される。

- ① 要求獲得のフェーズ
- ② 分析のフェーズ
- ③ 文書化のフェーズ
- ④ 検証のフェーズ

分析のフェーズと検証のフェーズからは、要求獲得のフェーズに戻ることもあり得る。検証をパスすると、その文書が「要件定義書」として公開される。以下で、この 4 つのフェーズについて述べる。

要求獲得のフェーズ

要求獲得のフェーズで最初に行う作業は、要求を獲得する対象のステークホルダ（利害関係者: ソフトウェアの全ライフサイクルを通してソフトウェアに正当な利害関係を持つ人たち）を特定することである。共通フレーム 2013 では、ステークホルダとして次のような人たちをあげている [IPA13a]。

- 利用者
- 運用者
- 支援者
- 開発者
- 製作者
- 教育訓練者
- 保守者
- 廃棄者
- 取得者
- 供給者
- 外部に対して対応の責任を負う当事者
- 規制機関、並びに社会の構成員
- その他

次は、その要件を獲得する手段である。ITIL (Information Technology Infrastructure Library) はその v3 の「システムデザイン」の部分で、この要求獲得のフェーズで使える方法には以下の 10 個があるとしている[OGC08a]。

- ① インタビュー
- ② ワークショップ
- ③ 観察
- ④ プロトコル分析：ユーザにタスクを遂行させながら、各ステップを説明させる方法
- ⑤ シャドーイング：一定の期間（例えば 1 日）ユーザを追跡して、特定の作業を調査する方法
- ⑥ シナリオ分析
- ⑦ プロトタイプング
- ⑧ アンケート
- ⑨ 特定用途のレコード：特定の課題やタスクについて、ユーザに記録を付けさせる方法
- ⑩ 活動サンプリング：前記特定用途のレコードの方式に、経過時間を記録するなど定量面を加味したもの

分析のフェーズ

共通フレーム 2013 での記載に戻るが、要件定義書を作成する人は要件として獲得したものを、次のような区分にまとめる[IPA13a]。

- ① 業務要件
 - 業務内容（手順、入出力情報、組織、責任、権限、など）
 - 業務特性
 - 業務用語
 - 外部環境と業務の関係、授受する情報
- ② 組織および環境要件の具体化
- ③ 機能要件
 - 必要なシステムの機能を明確にする
 - ◆ 業務を構成するシステム間の情報（データ）の流れの明確化

- ◆ 人の作業とシステムの機能の実現範囲の定義
- ◆ 他のシステムとのインタフェース

④ 非機能要件

非機能要件の考え方については、後述する。

文書化のフェーズ

分析の次の段階は、この分析の結果を「要件定義書」としての文書にまとめることである。この文書化の方法には、いくつかのものがある。

オーソドックスな書き方の例

この標準に基づいたオーソドックスな書き方の例として、秋本芳伸氏らの著書[AKI04]が示している方法がある。

またこの著書で秋本氏は、要件定義書に書かれる「要求」には 3 つのレベルがあると指摘している。具体的には、以下の通りである[AKI04]。

- 「業務要求」: システム開発の動機となる要求。この要求がかなえられないのであれば、システム開発の意味がないというレベルのもの。REBOK の「ビジネス要求」が、これに相当する。
- 「ユーザ要求」: ユーザがシステムで行う作業についての要求。実現しなければユーザの仕事に支障がある。REBOK の「システム要求」が、これに相当する。
- 「機能要求」: 業務要求やユーザ要求をシステムとして実現するために必要な機能に関する具体的な要求。REBOK の「ソフトウェア要求」の一部が、これに相当する。

この 3 種類の要求を全て記載する方法として、最初の「業務要求」は要件定義書の最初の部分などで、「このシステムの目的」というような標題の下で明確に記述する。その上で要件定義書の本体の部分で、2 つ目の「ユーザ要求」と 3 つ目の「機能要求」を仕様とペアにして、「ひな形」の中で記述する方法がある。これについても、後述する。

オーソドックスな方法では、実体関連図、データフロー図、状態遷移図⁷などの図を要件の説明に使用することがある。

オブジェクト指向流の書き方の例

オブジェクト指向流の UML (Unified Modeling Language) に基づいた書き方の例として、アリスター・コーバーン氏の著書[COC01]が提示している方法がある。

コーバーン氏は、UML⁸の中のユースケース図とユースケース記述を使用し、時には「シナリオ」を使って具体的な動きを記述することも取り込んで、オブジェクト指向のやり方での要件定義書の記載方法を提案している。

なおユースケース図以外に、クラス図、シーケンス図、状態マシン図などの他の UML で定義されている図も、要件の説明に使用される。

⁷ 実体関連図、データフロー図、状態遷移図については、第 22 章で説明する。

⁸ UML についても、第 22 章で説明する。

それ以外の書き方の例

ユニークな記載の仕方を提案したものに、清水吉男氏の方法がある⁹[SIM05]。清水氏の表記方法を、USDM (Universal Specification Describing Manner) 表記法と呼ぶ。

清水氏は、要件定義書ではまず「何を実現したいのか」を明確にするために「要求」をしつかりと記載し、それぞれの要求に「なぜそれが必要なのか」を明らかにする「理由」を付ける。その上でそれぞれの要求を具現化するための「仕様」を、該当する要求の傘の下に記述する方法を提案している。さらにこの要求を、上位の要求と下位の要求の二段階に階層化する。

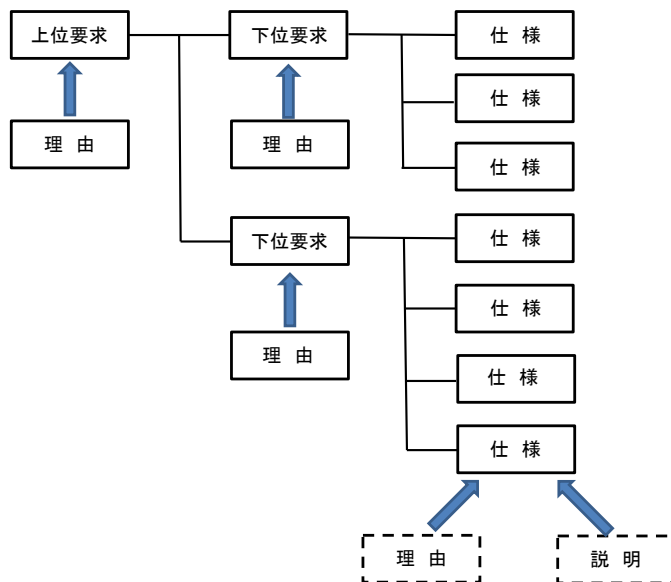
清水氏の方法によれば、要求とは「実現したいゴール」である。端的に「実現して欲しいこと」といっても良い。例えば「お風呂が沸いたら、すぐにお風呂に誘導したい」は要求である。この要求は、ヒアリングの場などで言葉として交わされることはあるが、文書上では一般にどこにも表現されていないものであるという。

要求には、必ず「理由」を付けると決めた。「理由」は、その要求がなぜ必要なのかといった背景を示す。このお風呂の件に関わる要求の理由には、「燃料の消費を節約する」が該当する。

それでは「仕様」とは何だろうか。仕様とは「要求（実現して欲しいこと）を満たすための『具体的な振る舞い』の記述」である。したがって仕様は、必ずいずれかの要求に属することになる。

さらに仕様は、最終的には何らかの形でプログラムに変換されるものであり、「実現されていることを検証できるもの」である。つまり仕様は、実行可能でなければならない。別のいい方をすれば、「仕様」はプログラムを読むことで把握することができる。しかし「要求」と「理由」は、そのような方法で把握することができない。

上記の「お風呂の件」の要求に対する仕様の1つとして、「『沸く』状態の1分前に、『もうすぐお風呂が沸きます』と知らせる」が該当する。



図表 21-3 要求と仕様の関係 ([SIM05]より)

⁹ 清水氏が提案した形式は、IEEE の標準が提示している「ひな形」とは表面的には合致しない。しかし IEEE の標準にも 8 種類のひな形が用意されており、必ずそれらの中の 1 つに従わなければならないものというようなものではないと考える。

さらに SE が要件定義書を書いている過程で、「仕様」をどのようにソフトウェアとして実現したら良いかに気づいてそれを記述したくなると、その内容を「説明」として書けば良い。この「説明」を書き加えることで、要件定義書は一層深みを増すことになる。但し設計工程でこの「説明」として記述されたことを採用するかどうかは、設計者の裁量に任される。

これらの関係を、図表 21-3 で示す。

この清水氏が提案する方式では、要件定義書上に最低限必要とする「仕様」以外に、「要求」と「理由」を追加している。しかしこの「要求」と「理由」を追記することを通して、要件定義書を書く人の頭が整理されて要求に漏れや矛盾などがなくなるという効果があるという。さらに読む人も、何を実現すればよいのか（「仕様」）を把握するに当たって、それを通して本当は何をしたいのか（「要求」）、その背景は何か（「理由」）といったことを明確に把握できるようになる。たいへんすばらしい方式と評価できる。

なお清水氏は、この文書を Excel で記述することを推奨している。Excel を使用して箇条書きのように記載することで、要件定義書としての内容が一層充実するという。

なお日本情報システム・ユーザー協会（JUAS）はこの清水氏の方式を核にして、「要件定義書の書き方」についてまとめている[JUAS07a]。

検証のフェーズ

要件定義書は、図表 21-1 で示した場所に位置づけされる。従って、開発したソフトウェアがこの要件定義書に記述されたとおりに作成されているかを確認することが、このソフトウェア開発全体の検証作業である。

それでは、要件定義書そのもののチェックは、どうすれば良いのだろうか。これには、3つのポイントがある。

1 つ目は、誤字脱字がないこと、読みやすいことなどの基本的な文書としてのチェックに加えて、ステークホルダの要件が適切に記述されていること、記述されている非機能要求が妥当なものであること、などの視点からのチェックがある。要件定義書に書かれなかったことは、ソフトウェア上に実現しない。この観点からのチェックも必要である。これは、要件定義書そのものの検証である。

2 つ目は、「この要件定義書に基づいて開発されるソフトウェアが、この情報システムを取得しようとしている企業の要望を適切に満たしたものになるか」という、「妥当性確認」の見地からのものである。これがなされていなければ、最初に述べたソフトウェア全体の検証にこの要件定義書を使用することは適切ではない。

そして最後の 3 つ目は、「ここに書かれている要件が実際に開発を担当するソフトウェア技術者に的確に伝達されるか」のチェックである。ここで間違いが起きると間違ったソフトウェアが開発されることになり、そのまま進めると後で大きな手戻りが発生する。それを予防するために、この段階での、この視点からのチェックが重要である。

しかしそれは、どのようにしたら確認できるのだろうか。単純な確認なら、話し合いをすれば良いかもしれない。質問の機会を設ける、というような方法もある。しかしより完全に行うためには、要件定義書を読んだソフトウェア技術者が理解したことをソフトウェア技術者の立場と言葉で再度記述し、それを要件定義書の作成者が読んで確認する、というようなことしか方法がないのかもしれない。

暗黙知の問題もある。要件定義書には、それを書く人が必要と考えることを記述する。しかし極端な話をすると、要件定義書を書く人は「自分が何を知っているのか」を知らない。ここで気がつかないことは、要件定義書に記載されることはない。この部分に、大切な事項が含まれていることがあり得る。

いずれにしろ要件定義書を書く側と読む側が、このような問題が起きる可能性が常にあることと、起きると大きな問題になることをよく認識して、この問題を発生させないように留意することが重要である。

非機能要求として何を記述するか

IEEE の標準が要求している非機能要求には、次のものが含まれている [IEEE98f]。

- 外部とのインタフェース（人間、ハードウェア、他のソフトウェアとの）
- パフォーマンス（スピード、アベイラビリティ、レスポンスタイム、復旧時間、など）
- アトリビュート（移植性、正確さ（コレクトネス）、保守の容易性、安全性、など）
- 設計上の制約

JUAS が平成 7 年度に実施した非機能要求についての研究の成果が、報告書の形で発行されている。それによると非機能要求には、次の 10 種類、230 個の要求がある [JUAS08a]。

- 機能性
- 信頼性
- 使用性
- 効率性
- 保守性
- 移植性
- 障害許容性
- 効果性
- 運用性
- 技術要件

この中の最初の 6 つは、ソフトウェアの品質についての外部品質／内部品質をそのまま取り込んでおり [JIS03a]、この 6 つについては ISO と IEC が発行した技術報告書 ([ISO03a]、[ISO03b]) に非機能要求のサンプルがある¹⁰。

実際に要件定義書で非機能要求を記述するに当たっては、JUAS が挙げた 230 個の非機能要求の中から自社に必要なものなどを 20～30 個程度選んで、目標とするそれぞれの値を定めて記述するのが良い。

なお JUAS が非機能要件の研究を行った頃は、情報システムへの不正なアクセスなどがあまり報告されていなかったもので、この分野は前記報告書では取り扱われていない。現時点では、権限のないアクセスや不正なアクセスにどう対応するのかということは、非機能要求の非常に重要な領域と考える。

なお IPA が現在管理している「非機能要求グレード」は、コンパクトでうまくまとめられていると評価できる [IPA10b]。

¹⁰ ここで取り上げた ISO/IEC 9126 シリーズの規格と技術報告書は、既に廃止されてしまった。

要求工学全体をカバーする書籍

IEEE の標準とこれまで紹介した書籍は、いずれも要件定義書を作成するという立場で用意されたものだった。しかし要求工学までさかのぼって、いかにしてステークホルダの要求を把握するかという段階から議論を始めている書籍がある。その一冊がイアン・サマヴィル (Ian Sommerville) 他のもので[SOM97]であり、もう一冊がカール・E. ウィーガース (Karl E. Wiegars) のもので[WIE03]である。

IEEE は現在の 1998 年版の標準の前に、1993 年版の標準を持っていた。サマヴィル他の本は、その 1993 年版の標準に基づいて書かれている。その意味では、少し古い。しかしステークホルダから要求を聞き出してきてそれを整理するところについては、1993 年版と 1998 年版で差はない。

一方のウィーガースのものは、1998 年版に基づいて作成されている。

単に要件定義書の書き方という表面的なものでなく、要求工学までさかのぼってしっかりと勉強してみたいということであるなら、これらの 2 冊の本のいずれかをひも解いてみることはたいへん有効である。

要件定義書は誰が書くのが良いか

要件定義書は、情報システムの開発を委託するユーザ側の組織が作成するべきである。もっと具体的に、それでは要件定義書は誰が書くのが妥当だろうか。これには、以下の 3 つの考え方があ

- 「SE、つまりソフトウェア技術者が書くべき」とするもの
- 「ユーザと SE が共同で書くべき」とするもの
- 「『要求エンジニア』と呼ばれる、特別の訓練を受けた人が書くべき」とするもの

要件定義書には、ユーザに関する業務処理に関わる側面と、その遂行を支援する、あるいは実施する機能をネットワークとコンピュータを使ってどう実現するかという情報処理に関わる側面がある。したがってこれを 1 人の人が記述するとすれば、その人はこの両面に堪能でなければならない。換言すれば、ユーザが書くとするればその人は情報処理の側面について特別の訓練を受けておかなければならず、SE が書くとするればその人は業務処理の側面を熟知していなければならない。要件定義書を書くことに関してこのような特別の訓練を受けた人を、ここでは「要求エンジニア」と呼んでいる。しかし情報処理推進機構 (IPA) が公表した IT スキル標準 (v3) [IPA08a]には、このようなエンジニアは定義されていない¹¹。

清水氏は、要件定義書はユーザと SE が共同で書くべきとしている[SIM05]。つまり「要求」の部分ユーザが書き、「仕様」の部分 SE が書くのがよいという。さらに複数の SE が担当分野を分けて、平行して仕様を記述することも可能という。

前章に記述した「要求仕様書」をユーザが作成した場合は、それを参考にして SE が要件定義書を書く、という方法もある。

要件定義書は保守で使用するのか

清水氏によると、要件定義書は開発を担当する SE がその業務領域について詳しい場合、簡便なものでよいという。逆にその業務領域について詳しくない場合は、詳細に記述する必要がある。つまり要件定義書に記述すべき内容は必ずしも一定しておらず、開発担当者のその領

¹¹ 「IT スキル標準 (ITSS)」については、第 47 章で記述する。

域の理解の深さに応じて変わるものだという。

そうだとすると、要件定義書はソフトウェアの保守の作業ではたいへんに使いにくい。また清水氏を引き合いに出すが、彼は、保守で使用するために仕様を明記した資料を「機能仕様書」と名付け、開発期間中、または開発が終了した直後に要件定義書を基に作成する必要があると述べている[SIM05]。

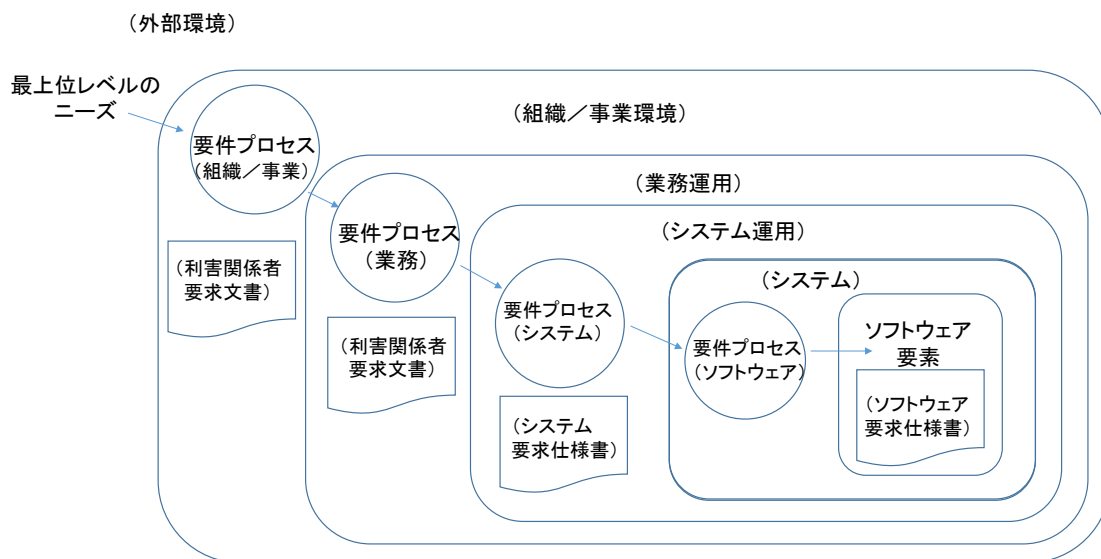
上位の要求の記述方法

REBOK では、要件定義書に記述される「ソフトウェア要求」の上位に、さらに 2 種類の要求があると書いた。上から、「ビジネス要求」と「システム要求」である。これらの要求は、どう表せば良いのだろうか。これには、3 つの考え方がある。

1 つ目は、USDM 表記法で要件定義書を記述している場合、USDM 表記法での上位の要求に「システム要求」を、下位の要求に「ソフトウェア要求」を記述する、「ビジネス要求」はこれとは違う体系で記述する、という方法である。例えば、この章末に付 1 として添付した要件定義書の例では、「1.2. 範囲」の部分で「ビジネス要求」を記述することができる。

2 つ目は、3 つとも USDM 表記法で記述するという方法である。この場合 USDM 表記法の要求は上位と下位の 2 レベルではなく、3 レベルになる。

そして 3 つ目の方法は、今 IEEE が持っている規格をそのまま適用して、3 種類の要件定義書を作成するという考え方である。「ソフトウェア要求」を記述する要件定義書は、IEEE Std 830-1998[IEEE98f]で規格が用意されている。同じように「システム要求」については IEEE Std 1233-1998 [IEEE98h]として、「ビジネス要求」には IEEE Std 1362-1998 [IEEE98i]としてそれぞれ別の規格が用意されており、それらに基づいて、別の文書として要件定義書を作成するというものである。



図表 21-4 3 つの文書の位置づけ ([JIS14]より)

なお 2014 年に JIS 規格として発行された JIS X 0166 : 2014 (元の ISO 等の規格は ISO/IEC/IEEE 29148 : 2011) はこの IEEE の考え方を引き継いで、「利害関係者要求文書」、

「システム要求仕様書」、「ソフトウェア要求仕様書」という名称で要件定義書の内容を定めている[JIS14]。この3つの文書の位置づけを図表 21-4 に、その目次例を付 2 として章末に添付する。

仕様変更

要件定義書は構成管理の下に位置付けて、その変更は厳格に管理されなければならない¹²。

要件定義書に記載された内容の変更を、「仕様変更」と呼ぶ。設計作業以降に入る仕様変更は作業の手戻りを発生させて作業効率を低下させ、さらにその対応が良くないと欠陥の原因にもなって品質の低下を招くことになる。仕様変更は、極力避けることが望ましい¹³。

キーワード

要件定義書、SRS、ステークホルダ、利害関係者、品質、要求獲得のプロセス、要求工学、要求工学知識体系、REBOK、ビジネス要求、システム要求、ソフトウェア要求、機能、機能要求、非機能要求、業務要求、ユーザ要求、UML、ユースケース図、ユースケース記述、シナリオ、USDM 表記法、要求、理由、仕様、説明、要求エンジニア、IT スキル標準、機能仕様書、要求仕様書、検証、妥当性確認、暗黙知、利害関係者要求文書、システム要求仕様書、ソフトウェア要求仕様書、仕様変更

略語

SRS : Software Requirement Specification

RE : Requirement Engineering

REBOK : Requirement Engineering Body of Knowledge

ITIL : Information Technology Infrastructure Library

UML : Unified Modeling Language

USDM : Universal Specification Describing Manner

人名

フィリップ・B. クロスビー (Philip B. Crosby)、イアン・サマヴィル (Ian Sommerville)、カール・E. ウィーガース (Karl E. Wiegers)、シャリ・ローレンス・フリーガ (Shari Lawrence Pfleeger)

規格

ISO 9000 : 2015、REBOK、JIS X 0166 : 2014、ISO/IEC/IEEE 29148 : 2011

参考文献とリンク先

[AKI04] 秋本芳伸、岡田泰子著、「若手 SE のための要求仕様のまとめ方」、(株) ディー・アート、2004 年。

[COC01] アリスター・コバーン著、ウルシステムズ (株) 監訳、「ユースケース実践ガイド-効

¹² 構成管理については、第 8 章で述べた。

¹³ ケイパース・ジョーンズによると、要件定義の作業が終わった後のソフトウェア開発の期間中に、月平均で 2%の要求の追加があるという[JON07]

果的なユースケースの書き方」、(株) 翔泳社、2001 年。

この本の原書は、以下のものである。

Alistair Cockburn, “Writing Effective Use Case,” Addison Wesley Longman, 2001.

[CRO79] フィリップ・B. クロスビー著、小林宏治監訳、「クオリティ・マネジメント：よい品質をタダで手に入れる法」、日本能率協会、1980 年。

この本の原書は、以下のものである。

Philip B. Crosby, “Quality is Free,” MacGraw-Hill, 1979.

[IEEE98f] IEEE-SA Standards Board, “IEEE Recommended Practice for Software Requirements Specifications IEEE Std 830-1998”, The Institute of Electrical and Electronics Engineers, Inc., 1998.

[IEEE98h] IEEE-SA Standards Board, “IEEE Guide for Developing System Requirements Specifications IEEE Std 1233, 1998 Edition (R2002), “ The Institute of Electrical and Electronics Engineers, Inc., 1998.

[IEEE98i] IEEE-SA Standards Board, “IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document IEEE Std 1362 – 1998, “The Institute of Electrical and Electronics Engineers, Inc., 1998.

[IPA08a] 独立行政法人情報処理推進機構 IT スキル標準センター、「IT スキル標準 v3」、平成 20 年 3 月 31 日。

なおこの資料は、以下の URL からダウンロード可能である。

http://www.ipa.go.jp/jinzai/itss/download_V3.html

[IPA10b] 独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター、「非機能要求グレード利用ガイド [利用者編]」、独立行政法人情報処理推進機構、2010年。

この資料は、以下の URL からダウンロードできる。

<http://www.ipa.go.jp/sec/softwareengineering/reports/20100416.html>

[IPA13a] 情報処理推進機構ソフトウェア・エンジニアリング・センター編、「共通フレーム 2013 経営者、業務部門が参画するシステム開発及び取引のために ソフトウェアライフサイクルプロセス 共通フレーム 2013」、オーム社、平成 25 年。

[ISO03a] ISO/IEC, “Software Engineering – Product Quality – Part 2 : External metrics ISO/IEC TR 9126-2 : 2003,” ISO/IEC, 2003.

[ISO03b] ISO/IEC, “Software Engineering – Product Quality – Part 3 : Internal metrics ISO/IEC TR 9126-3 : 2003,” ISO/IEC, 2003.

[JIS03a] 日本工業標準調査会審議、「JIS ソフトウェア製品の品質－第 1 部：品質モデル JIS X0129-1 : 2003 (ISO/IEC 9126-1 : 2001)」、日本規格協会、平成 15 年。

[JIS06a] 日本工業標準調査会審議、「JIS 品質マネジメントシステム－基本及び用語 JIS Q 9000 : 2006 (ISO 9000 : 2005)」、日本規格協会、平成 18 年。

[JIS14] 日本工業標準調査会審議、「システム及びソフトウェア技術－ライフサイクルプロセス－要求エンジニアリング JIS X 0166 : 2014 (ISO/IEC/IEEE 29148 : 2011)」、日本規格協会、平成 26 年。

[JISA11] 情報サービス産業協会 REBOK 企画 WG、「要求工学知識体系 REBOK 第 1 版」、近代科学社、2011 年 6 月 30 日。

[JON07] Capers Jones 著、「ソフトウェア見積もりのすべて－現実に即した規模・品質・工

数・工期の予測―第 2 版」、構造計画研究所、2009 年。

この本の原書は、以下のものである。

Capers Jones, “Estimating Software Costs Bringing Realism to Estimating Second Edition,” The McGraw Hill, 2007.

[JUAS07] 「要求仕様定義ガイドライン～UVC 研究プロジェクト報告書～」、(社) 日本情報システム・ユーザー協会、平成 19 年 3 月。

[JUAS08a] 日本情報システム・ユーザー協会、「検収フェーズのモデル取引整備報告書 UVC 研究プロジェクト II 報告書 非機能要求仕様定義ガイドライン」、日本情報システム・ユーザー協会、平成 20 年 6 月。

[OGC08a] Office of Government Commerce、「ITIL v3 サービスデザイン」、The Stationary Office、2008 年 5 月。

[PFL09] Shari Lawrence Pfleeger, Joanna M. Atlee, “Software Engineering Theory and Practice fourth Edition,” Prentice Hall, 2009.

[SAN94] J. サンダース、E. カラン著、原田暉他訳、「ソフトウェア品質向上のすすめ―新しいソフトウェア開発の標準」、(株) トッパン、1996 年。

この本の原書は、以下のものである。

Joc Sanders, Eugene Curran, “Software Quality A Framework for Success in Software Development and Support,” Addison-Wesley Publishing, 1994.

[SIM05] 清水吉男著、「[入門+実践]要求を仕様化する技術表現する技術～仕様が書けていますか」、(株) 技術評論社、平成 17 年。

なおこの本は、2010 年 (平成 22 年) に第 2 版が発行された。第 2 版は、以下のものである。

清水吉男著、「[入門+実践]要求を仕様化する技術表現する技術～仕様が書けていますか 改訂第 2 版」、(株) 技術評論社、2010 年 6 月 1 日。

[SOM97] Ian Sommerville 他著、富野壽監訳、「要求定義工学プラクティスガイド」、(株) 構造計画研究所、2000 年。

この本の原書は、次のものである。

Ian Sommerville, Pete Sawyer “Requirements Engineering A Good Practice Guide,” John Wiley & Sons, 1997.

[WIE03] Karl E. Wieggers 著、渡部洋子監訳、「ソフトウェア要求 顧客が望むシステムとは」、日経 BP ソフトプレス、2003 年。

この本の原書は、次のものである。

Karl E. Wieggers, “Software Requirements, Second Edition,” Microsoft Press, 2003.

(2007 年 (平成 19 年) 5 月 17 日 初版作成)
 (2008 年 (平成 20 年) 8 月 14 日 一部修正)
 (2009 年 (平成 21 年) 6 月 26 日 一部修正)
 (2010 年 (平成 22 年) 9 月 9 日 一部修正)
 (2011 年 (平成 23 年) 10 月 18 日 全面改定)
 (2014 年 (平成 26 年) 11 月 30 日 一部修正)
 (2016 年 (平成 28 年) 4 月 19 日 一部修正)

(2017 年 (平成 29 年) 1 月 17 日 一部修正)

付 1. 要件定義書のプロトタイプ([IEEE98]より)

標 題	内 容
1. はじめに	
1.1. 目的	この要件定義書の目的、想定される読者
1.2. 範囲	開発するソフトウェアの種類。説明。このソフトウェア開発で達成しようとする目的、ゴール、得られる利益。「業務要求」もここに記載する。
1.3. 専門用語、頭文字語、略語の定義	
1.4. 参照	参照する資料・文献
1.5. 概要	この要件定義書の構成
2. 全体の記述	
2.1. 製品についての考え方	システムを構成する他の部分との関係、インタフェース
2.2. 製品の機能	3 で述べる機能の概要
2.3. ユーザの特性	想定されるユーザの経験や技術レベルなどで特記すべき事項
2.4. 制約	このソフトウェアの開発に伴う制約事項
2.5. 前提	このソフトウェア開発の前提条件
2.6. 先送りされる要求事項	
3. 要求と仕様	要件定義書の本体部分。ここでは内容の記述を省略するが、8種類の書き方が提示されている。
付録	
索引	

付 2 要件定義書の目次例（「JIS14」より）

1. 利害関係者要求仕様書
 - (1). 事業の目的
 - (2). 事業の適用範囲
 - (3). 事業の概要
 - (4). 利害関係者
 - (5). 事業環境
 - (6). ゴール及び目標
 - (7). 事業モデル
 - (8). 情報環境
 - (9). 業務プロセス
 - (10). 業務運用方針及びルール
 - (11). 業務運用制約
 - (12). 業務運用モード
 - (13). 業務運用品質
 - (14). 業務の構造
 - (15). 利用者要求事項
 - (16). システムレベルの運用概念
 - (17). 運用シナリオ
 - (18). プロジェクト制約

2. システム要求仕様書
 - (1). システムの目的
 - (2). システムの適用範囲
 - (3). システムの概要
 - ①. システムの状況
 - ②. システムの機能
 - ③. 利用者特性
 - (4). 機能要求事項
 - (5). 使用性要求事項
 - (6). 性能要求事項
 - (7). システムインタフェース
 - (8). システムの運用
 - ①. 人間システム統合要求事項
 - ②. 保守性
 - ③. 信頼性
 - (9). システムモード及び状態
 - (10). 物理的特性
 - ①. 物理的要求事項
 - ②. 適用性要求事項

- (11). 環境条件
- (12). システムセキュリティ
- (13). 情報管理
- (14). 方針及び規制
- (15). システムライフサイクル持続性
- (16). パッケージ化・取扱・出荷・配送
- (17). 検証
- (18). 前提条件及び依存性

3. ソフトウェア要求仕様書

- (1). 目的
- (2). 適用範囲
- (3). 製品の概要
 - ①. システムインタフェース
 - ②. 利用者インタフェース
 - ③. ハードウェアインタフェース
 - ④. ソフトウェアインタフェース
 - ⑤. 通信インタフェース
 - ⑥. メモリ制約
 - ⑦. 通信
 - ⑧. サイト適用要求事項
- (4). 製品の機能
- (5). 利用者特性
- (6). 制限
- (7). 前提条件及び依存性
- (8). 要求事項の配分
- (9). 詳細要求事項
- (10). 外部インタフェース
- (11). 機能
- (12). 使用性要求事項
- (13). 性能要求事項
- (14). 論理データベース要求事項
- (15). 設計制約
- (16). 標準への適合
- (17). ソフトウェアシステム属性
- (18). 検証
- (19). 支援情報