特集
システムとソフトウェアの品質

基専

5. ソフトウェア非機能要求の定義

―品質の良いソフトウェアを作るために―

野中　誠（東洋大学）　東　基衛（早稲田大学）

品質の良いものを顧客に提供するとは

システムやソフトウェアに限らず，品質を考える上で忘れてはならない大原則がある．それは「品質の良し悪しは顧客の評価で決まる」という原則[1]である．開発者の論理に基づいて製品のカタログスペック上の数値をいくら競ったところで，それが顧客の満足に結びつかなければ品質が良いとはいえない．たとえ顧客が専門性のない素人であってもこの原則は変わらない．なぜなら，品質の良し悪しは相対的認識に立脚しており，誰かの評価が得られない限り「品質が良い」とはいえないためである[1]．品質を考える起点は，製品やサービスに対して外的な基準となる顧客満足である．

また，顧客満足は心理的なものであり，製品やサービスの特徴にかかわる値（特性値）を高めればつねに満足してもらえるものではない．特性値を高めても心理的に満足しないが，特性値が低くなると不満に思うような「当たり前品質」や，ある特徴が備わっていなくても不満に思わないが，ひとたびその特徴が備わると心理的に満足するような「魅力的品質」などがある[2]．また，今日の製品やサービスでは「顧客」の範囲を消費者だけでなく環境や社会にまで広げることが求められることが多い[1]．このように，品質の良し悪しが顧客満足を起点にしていることを踏まえた上で，顧客満足と製品やサービスが持つ特徴との対応関係，さらには「顧客」の範囲の広がりを考えると，品質の良い製品やサービスを顧客に提供し続けることがいかに奥深く，挑戦的なことであるかが分かる．

非機能的特徴が顧客満足に強く影響する

さて，議論の対象をソフトウェアに限定し，ソフトウェアの非機能な特徴という顧客満足に大きく影響する要素に絞って話を進めよう．特に，非機能的な特徴の中でも，とりわけ要求定義が難しい品質面の特徴に焦点を当てて，ISO/IEC 25010[3]およびISO/IEC 25030[4]の内容におおむね基づいて解説する．なお，非機能的な特徴と品質面の特徴との関係については，本稿コラムにて述べる．

機能面の特徴と品質面の特徴

ソフトウェア開発の第一歩は，外的基準である顧客満足を達成するため，すなわち顧客の明示的ニーズと暗示的ニーズの両方を満たすために，所与の制約の範囲で，ソフトウェアに固有の特徴としてどのようなものを持たせるかを考えること（要求定義を行うこと）である．ソフトウェアに固有の特徴は，ソフトウェアの機能にかかわる特徴（機能面の特徴）と，ソフトウェアの品質にかかわる特徴（品質面の特徴）に大別される（**図-1**）．機能面の特徴とは，入力を出力に変換するソフトウェアの働きにかかわるものを指す．一方，品質面の特徴とは，機能面の特徴が利用者に適切に提供されている度合いにかかわるものであり，信頼性や使用性，保守性などの品質特性により分類される．

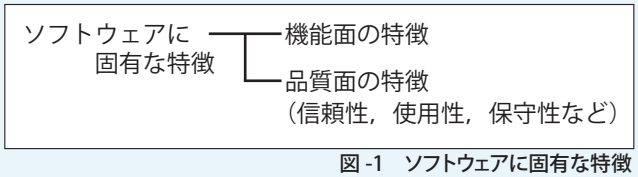

図-1　ソフトウェアに固有な特徴

品質面の特徴は，機能面の特徴に比べて，漏れなく識別することが難しい．機能面の特徴であれば「このような出力を得たい」ということが顧客や利用者の視点でイメージしやすいが，品質面の特徴は，実際に作られたものを使ってみないとイメージしにくいことに加えて，経験，スキル，好みなどが顧客，特に利用者によって異なることが問題を難しくしている．しかし，品質面の特徴の充足状況は顧客満足に大きく影響する．そのため，顧客満足を追求するためには，「魅力的品質」や「当たり前品質」としてどのような品質面の特徴をソフトウェアに持たせる必要があるのかを強く意識する必要がある．そして，重要度の高い品質面の特徴を識別し，これがソフトウェアに確実に作り込まれたことを評価する必要がある．

◉ 品質面の特徴が顧客満足に影響する例

やや抽象的な話なので，具体的な事例を交えながら品質面の特徴と顧客満足との関係を考えていく．ここでは，品質面の特徴の不備が顧客の不満に結びつく例，ソフトウェアの品質面の特徴を充足することにより顧客不満を解消した例，顧客満足度の向上につながった例を説明する．

事例 1：品質面の特徴の不備が顧客の不満を招く例

ある産業用装置は，加工精度の細かさと稼働率の高さという 2 つが製品競争力の主たる要素である．また，納入先における顧客のものづくりの状況に適応させるために，装置内部に組み込まれたソフトウェアに顧客独自の要求に基づく機能を個別に追加したり，変更を行ったりしている．そして，ソフトウェア構成管理の複雑さを回避するためにソフトウェアを一本化し，顧客ごとに異なる多様な機能要求を 1 つのソフトウェアに集約して改良開発を行っている．筆者の知る範囲では，この事例に限らず，同様のスタイルでソフトウェアの改良開発を行っていることが多いようである．

いま，装置の機能および性能の向上や不具合修正などのために，ソフトウェアを改版したとする．これを顧客先で稼働中の装置に適用するときに，顧客の立場からするとどのような不満が生じるだろうか．顧客からすれば，自分たちが必要としない機能が追加されたソフトウェアのために，目の前で動いている装置をいったん止めて，ソフトウェア構成の検証や動作確認などを行わなければならない状況になるかもしれない．先に述べたとおり，この製品の主たる競争要因は稼働率の高さであり，顧客としてはできるだけ稼働を止めたくない．しかし，自分たちにとっては不要な機能を多く含んだソフトウェアの改版のために，装置の稼働を止めざるを得なくなってしまう．すなわち，顧客満足度が低下してしまう．装置の出荷時には高い稼働率を誇っていたのに，ソフトウェア改版時における装置稼働への影響という品質面の特徴が，重要な製品競争力の要素をスポイルしてしまっているのである．このようなソフトウェアは，残念ながら総合的に品質の良いソフトウェアであるとはいえない．この例では保守性という品質特性が顧客満足度に影響を与えている．

組込みシステム製品の開発では，機械系統，電気系統，ソフトウェアの順で開発が進み，この順に従ってシステムに作り込まれる特徴が決まることが多い．また，その決定権の強さもこの順序に従うため，ソフトウェアの品質面の特徴が十分に検討されなかったり，重視されなかったりすることが多いようである．しかし，このような「言い訳」をいつまでもしていてはダメで，顧客満足にかかわる重要な特徴を，機械系統，電気系統，ソフトウェアが一体となってどのように実現するかを考えなければならない．繰り返しになるが，品質の良し悪しは開発側の論理で決まるのではなく，顧客の評価で決まるのである．

事例 2：品質面の特徴の充足が顧客の不満を解消する例

あるカーナビメーカでは，市場に後発で参入したこともあって，独自性をより明確に打ち出すために操作体系を工夫し，利用者の直観に近い操作体系を備えた（と思って作った）製品を市場に投入した．しかし，利用者からすると，他社製品から当該製品へと乗り換えたときに違和感を覚える操作体系となっていたため，操作方法を学習するのに時間がかか

ったり，誤操作の原因となったりするなど，利用者の不満の原因となっていた．そこで，次期製品では，せっかくの工夫を施した操作体系を全面的に見直し，他社製品から乗り換えたときにスムーズに利用できる製品へと改良した．これにより，利用者の不満を解消することができ，このカーナビ製品に新たに作り込んだ「魅力的品質」にかかわる特徴を利用者に強く訴求できるようになった．

この例では，製品競争力の主たる軸ではないところに独自の工夫をしてしまったために，それがかえって利用者の不満を招いてしまった．しかし，これを改善し，「他社製品からの乗り換えのしやすさ」という品質面の特徴を早期にソフトウェアに作り込んだことが顧客不満の解消に貢献した．そして，この製品ブランドに対する利用者のロイヤルティ低下を最小限に抑えることができた．顧客満足は，ブランドロイヤルティに影響し，買い換え時のブランド再選択や口コミにつながる．それは当該製品にかかわる事業の継続性に影響する．このような一連のつながりを意識して，ソフトウェアに作り込むべき品質面の特徴を考えてほしい．

事例 3：品質面の特徴の充足が顧客やビジネスの価値を生み出す例

ソフトウェアに必要な品質面の特徴を作り込むことで顧客や利用者に満足をもたらし，それがビジネス価値を生み出す例ももちろんある．身近にある例では，iPhone や iPad に導入されている iOS の「ヌルヌル感」と評されるスムーズな画面表示などがある．このような特徴は利用者の感性に訴求し，製品を使うことに楽しみを与え，所有する喜びを刺激し，ひいては利用者のブランドロイヤルティに結びつく．このような特徴は，マーケティング分野などではヒドニック（hedonic）な特徴（後述する ISO/IEC 25010 で定める「利用時の品質モデル」の満足性に対応）と呼ばれ，機能的な特徴とともに顧客満足を構成する重要な要素として位置づけられている．

業務系システムの例として，セブン - イレブンの GOT（グラフィックオーダ端末）の例が挙げられる．同社では，「仮説検証」と呼ばれる商品発注方式を重視することで，販売における機会損失を減らす努力を長年にわたって積み重ねていることはよく知られている．しかし，2005 年以前の GOT では，発注者が行う「情報収集→仮説→発注→検証」といった業務の流れと，GOT 上で閲覧できる情報や画面遷移，使用するアプリケーションの流れとの間にギャップがあった．そのギャップが発注者の違いによる発注精度のばらつきを生み，現場のオペレーションの進化を阻害する要因になっていた．この状況を打破するため，2006 年に新たに導入した第 6 次総合情報システムの GOT では，発注者の業務の流れと GOT における流れが整合するように改良した．画面数も従来の 41 画面から 230 画面へと増加し，業務の流れに応じた横断的なシステム操作ができるようにした [5]．

これらの例で示したように，ソフトウェアの非機能的な特徴，とりわけ品質面の特徴は顧客満足に大きく影響する．ソフトウェア開発に携わる技術者で，顧客や利用者と直接の接点を持つ人は必ずしも多くない．そのためか，ソフトウェアの品質面の特徴が顧客満足やロイヤルティ，さらにはビジネス価値に結びつくという認識が薄れてしまいがちである．技術者として，自分の提供したものが顧客や利用者にとってどのような価値をもたらしたのかを振り返る習慣が必要である．

品質要求を定義するには

これまでの説明で，品質面の特徴の重要性を繰り返し述べてきた．ここでは，品質面の特徴を個別の要求事項（品質要求事項）として識別し，要求水準を定め，達成状況を評価するプロセスについて説明する．

◉ 制約条件やステークホルダを把握する

品質要求事項を識別する前に，まず，コストや納期などの制約条件を正しく把握する必要がある．そして，識別すべき重要なステークホルダ（利害関係者）を漏れなく把握し，それぞれのステークホルダが重視している要求事項を把握しておく必要がある．

品質特性	概要（品質副特性）
機能適合性	暗黙的・明示的ニーズを満たす機能が提供されているか（機能完全性，機能正確性，機能適切性）
性能効率性	時間や CPU など資源の使用量が効率的か（時間効率性，資源効率性，容量満足性）
互換性	他製品と共存したり情報交換したりできるか（共存性，相互運用性）
使用性	利用者のタスクが，効率的に，効果的に，満足して行われるか（適切度認識性，習得性，運用操作性，ユーザエラー防止性，ユーザインタフェース快美性，アクセシビリティ）
信頼性	所与の条件，所与の期間において要求機能が遂行できているか（成熟性，可用性，障害許容性（耐故障性），回復性）
セキュリティ	不当なアクセスに対して情報やデータが保護されるか（機密性，インテグリティ，否認防止性，責任追跡性，真正性）
保守性	保守による製品の修正を効率的で効果的に行えるか（モジュール性，再利用性，解析性，修正性，試験性）
移植性	異なる環境へと効率的に移動させることができるか（適応性，設置性，置換性）

表 -1　ISO/IEC 25010 システムとソフトウェアの品質モデル

特に，ステークホルダの識別では漏れが生じやすく，利用者に目を向けすぎたばかりに経営という「観点」が漏れてしまったり，この記事の冒頭に述べたように環境や社会という「顧客」が漏れてしまったりする．そうならないために，所与の制約条件と，ニーズを満たすべきステークホルダを正しく把握しておく必要がある．

品質モデルを用いて要求事項を識別する

品質要求事項を挙げるには，ISO/IEC 25010[3] に示されたシステムおよびソフトウェアの品質モデルが有用なモデルとして利用できる．**表 -1** に，ISO/IEC 25010 品質モデルに含まれる 8 つの品質特性とその概要，品質副特性を示す．

表 -1 に示したとおり，品質特性にはそれぞれ 2 ～ 6 個，全部で 31 個の品質副特性が定められている．なお，機能完全性という機能適合性の品質副特性は，機能全体がすべてのニーズを満たしている度合いにかかわる特性である．一方で，機能正確性は，正確な結果が必要な精度で得られている度合いを示している．また，機能適切性は，利用者のタスクを円滑に遂行させることができる度合いを示している．

表 -1 に挙げた品質特性や品質副特性に基づいて，具体的な品質要求事項を定義する．その際，品質モデルに示された「～性」という抽象度の高い記述にとどめるのではなく，対象のシステムに関連した具体的な表現で記述する．たとえば，事例 1 で挙げた産業用装置の例では，品質要求事項として「ソフトウェア機能変更に伴うソフトウェア更新の際に，その機能を disable にしている装置の動作に影響を及ぼさないこと」が挙げられる．これは，保守性の副特性である修正性にかかわる品質要求事項である．また，事例 2 で挙げた「他社製品から自社製品に乗り換えてきた利用者にとって習得しやすい操作体系」は，使用性の副特性である習得性にかかわる品質要求事項である．さらに，事例 3 で挙げた「仮説検証という業務の流れと整合性のある GOT の流れ」は，機能性の副特性である機能適切性にかかわる品質要求事項である．このように，品質要求事項を具体的に記述する．

品質特性および品質副特性は，システムを構成するコンポーネントによって結びつきの強さが異なる．たとえば，ユーザインタフェースのコンポーネントであれば使用性，データベース関連であればセキュリティ，通信関係であればセキュリティと相互運用性などのように，重点を置くべき品質特性が異なる．コンポーネントの性質に基づいて，どの品質特性にかかわる品質要求事項を重点的に識別すべきかを考えるとよい．

また，品質要求事項を具体化していくと，新たな機能要求の必要性に気がつくことがある．たとえば，使用性にかかわる品質要求事項を実現するためにグラフィカルな入力機能やショートカット入力機能を設けたり，信頼性にかかわる品質要求事項を実現するためにロギング，バックアップ，データ回復処理などの機能を設けたりすることがある．機能要求事項にかかわる性質として品質要求事項が識別されるだけでなく，品質要求事項を実現するために機能要求事項が識別されるという双方向の関係性を理解しておくとよい．

ところで，表 -1 に示した品質副特性について漏れなく検討すると，ソフトウェアの規模にもよるが，

膨大な量の品質要求事項が識別されることがある．品質要求事項を実現したときの顧客満足への影響度，実現しなかったときの安全性や経済的損失などにかかわるリスクの影響度の大きさ，実現するために必要なコスト，企業としてどのような製品を世に送り出したいかという製品戦略などを勘案して，取り扱う品質要求事項を重点指向で定めることも実務的には求められる．

ボトムアップ的に要求事項を識別する

前節では顧客ニーズに基づいて品質要求事項をトップダウン的に識別する方法を示したが，やはり，具体的なイメージのしにくさに起因する品質要求事項の識別漏れを防ぐことは難しい．これを防ぐには，アジャイル開発で用いられているアプローチのように，段階的にリリースされたソフトウェアを実際に用いながら顧客や利用者のフィードバックを早期に得て，品質要求事項を明確化していくという方法も有効である．また，W モデルと呼ばれるソフトウェア開発プロセスモデルで示されているように，受け入れテストやシステムテストで行われるようなテスト設計を要求定義プロセスで行うことにより，テスト観点に基づく品質要求定義を行うことも有効な方法である．このようにボトムアップ的に品質要求事項を識別する場合でも，表 -1 に示した品質特性は有用な枠組みとなる．

要求水準を定める

識別された品質要求事項について，どの水準を達成すれば良しと判断するのかを明確にするために，要求水準を定める．その際，できるだけ定量的な表現とすることが望ましい．たとえば，事例 1 の例にかかわる品質要求事項「ソフトウェア機能更新の際に，その機能を disable にしている装置の動作に影響を及ぼさないこと」について，該当する機能のいかなる更新においても一切の影響がでないことを要求するのか，該当する機能更新のうち 20% までは許容範囲とするのかなどである．また，その更新に伴う検証と動作確認のための装置の稼働を停止する時間は平均で何分以内とするのか，最長でも何時間以内とするのかなどである．汎用ソフトウェアの場合には，それが導入された環境によって資源制約や処理性能が異なるため，特定の利用状況（ハードウェア性能，OS のバージョンなど）を設定した上で要求水準を定めるようにする．**表 -2** に，品質要求水準の例を示す．

品質特性（品質副特性）	品質要求事項と要求水準
性能効率性（時間効率性）	新規注文の受付処理を 2 ミリ秒以内で処理する
信頼性（可用性）	ファイブナイン（99.999%）の可用性を実現する
信頼性（成熟性）	障害復旧を 2 時間以内とする
使用性（UI 快美性）	解析結果を色分けして表示する（実現／非実現の 2 水準）

表 -2　品質要求事項の要求水準の例（文献 6）を参考）

留意事項として，開発者の視点ではなく，利用者や顧客の視点で要求水準を定める必要がある．利用者や顧客が実際にソフトウェアを使ったときにどの水準であれば満足なのか，または許容範囲なのかという視点で要求水準を考える．

また，ほかの留意事項として，無理な要求水準の定量化は避けるべきである．要求事項には，その実現状況を計数値や計量値として定量的に測定できるものもあれば，「実現／非実現」などの分類として定性的にしか測定できないものがある．定性的にしか水準を定められない要求事項は，当然ながら，定性的な水準を定めればよい．

要求事項の達成度を評価する

要求事項の達成度評価は，品質要求定義の話ではなく，できあがったソフトウェアの評価の話だが，要求定義と並んで重要な活動なので言及しておく．品質要求事項ごとに定めた要求水準に基づいて，ソフトウェアの品質を顧客や利用者に届ける前に評価し，品質要求事項の達成状況を把握する．あるいは，ソフトウェアの実運用の過程で収集された利用ログデータなどを用いて，品質要求事項の達成状況を評価する．近年よく見られるような，ベンダのクラウド型サービスを顧客や利用者が使用する環境であれば，ベンダはソフトウェアの利用ログデータを容易

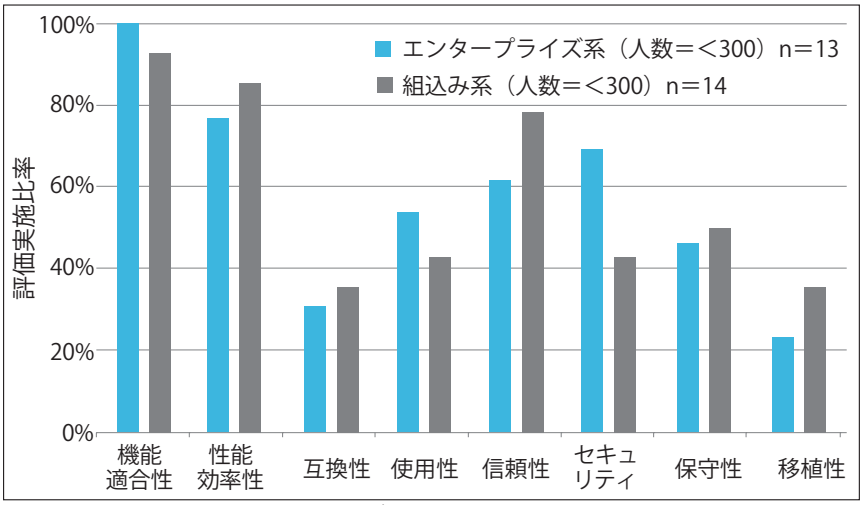

図 -2　開発部門で評価している品質特性 [7)]

に取得できるため，実運用の状況に基づく品質評価が行いやすくなる．もちろんその場合には，顧客や利用者に対して事前に説明責任を果たすことが前提となる．

品質要求定義・評価プロセスを改善する

品質要求定義と評価を 1 つのプロジェクトで実施して終わりにするのではなく，そこで得られた知見に基づいて継続的に改善していくことが重要である．継続的改善に関するよく知られたモデルに PDCA（Plan, Do, Check, Act）がある．Plan には 2 つの意味があり，目的・目標を明確化することと，実施手順を計画することが含まれる [1)]．品質要求事項の識別と要求水準の明確化は，P の 1 つ目の「目標の明確化」に相当する．開発プロセスを通じて品質要求事項を達成できたかどうかを確認（Check）するとともに，その品質要求事項が顧客満足にとって妥当だったかを確認（Check）する．顧客や利用者のニーズは時間が経つにつれて移ろいやすく，ある時点では「魅力的品質」だった品質要求事項がいつのまにか「当たり前品質」になったりする．品質要求にかかわる経験を再利用しつつ，その定期的な見直しが求められる．

さて，これまでの説明を読んで「品質要求定義と評価は大変で，とても現場ではやりきれない」という印象を持たれていないだろうか．そうだとしたら，それは大いなる誤解であると声を大にして言いたい．本稿の内容は，ソフトウェア開発の現場でこれまでに積み重ねられてきた経験を，再利用可能な知識として整理したものに過ぎない．ISO/IEC 25010 の品質モデルも同様である．また，ソフトウェアをまったくのゼロから開発し，品質要求事項と要求水準をまったくのゼロから定めることは稀である．ソフトウェア開発の現場には多くの経験が蓄積されている．その経験を品質要求事項と要求水準として整理し，再利用可能な現場の知識として活用していくことに意味があり，また，本稿はその一助になることを意図している．品質要求事項という，顧客満足にとって重要であるにもかかわらず，ややもするとあいまいなままに扱われてしまう要素を，まずは明確に定めるところから始めるとよい．

品質要求の定義・評価状況

ソフトウェア開発の現場で，品質要求はどのくらい定義され，評価されているのだろうか．その状況をうかがい知るのに役立つ 2 つの資料を紹介する．いずれも，重点化される品質特性に偏りがあることを示唆している．

文献 6）では，金融・保険分野 3 事例，公共分野 5 事例，および Web・コンテンツ分野 5 事例における特徴的な品質要求定義の例を挙げている．それらは，表 -1 の品質特性のうち，機能適合性，性能効率性，使用性，信頼性，セキュリティの 5 特性に偏っており，互換性，保守性，移植性は含まれていない．重点化される品質特性に偏りがあるという状況の一端を表しているといえよう．

また，文献 7）では，300 人以下のソフトウェア開発部門における，エンタープライズ系ソフトウェアと組込みソフトウェアの品質特性の評価状況などを示している（**図 -2**）．やはり，上述のような偏り

コラム

非機能要求と品質要求の関係

非機能要求と品質要求との関係や違いについて，しばしば議論になることがある．ISO/IEC 25030 では，次のような認識で捉えている．「ソフトウェアプロダクトの要求には，機能要求，品質要求，管理上の要求の 3 つがある．これらのうち，品質要求と管理上の要求が，ソフトウェアプロダクトにかかわる非機能要求である」．つまり，非機能要求は品質要求を包含しており，さらに管理上の要求を包含している．ここで，管理上の要求とは，ソフトウェアプロダクトの価格，納期，ソフトウェアの供給者などが含まれる．これらの要求事項は，プロダクトそのものではなく開発プロセスや開発組織に対する要求と捉える向きもあるが，ISO/IEC 25030 ではそのように捉えていない．それは，顧客の視点でソフトウェアを捉えた場合，価格，納期（利用可能な時期），供給者などの管理上の要求は，ソフトウェアにかかわる特徴として顧客の満足度に直接的に影響する要因であるためである．

また，その他のよくある議論として，表 -1 に示した品質特性のうち機能適合性は，機能要求にかかわるものであって非機能要求とはいえないのではないかという見解もある．これは，機能適合性の捉え方に誤りがあるために生じる誤解である．機能適合性は，ソフトウェアが保有する機能の集合が顧客ニーズの集合をうまく捉えている程度を示しており，機能要求そのものを述べたものではない．事例 3 でセブン - イレブンの GOT の例を示したが，個別の機能が揃っていることと，それらが利用者のニーズに即した形で配置されているかは別である．

が読み取れる．エンタープライズ系ではセキュリティの評価実施比率が高く，組込み系では使用性の評価実施比率が高い．いずれの例からも，多面的かつ重点指向で品質要求を定義・評価している様子が読み取れる．これらの状況を考慮すると，品質要求定義にあたっては，品質モデルの品質特性および品質副特性を網羅的に扱うのではなく，ソフトウェアを構成するコンポーネントごとに品質副特性ごとに重要度を三段階程度に分け，品質測定法や方法の網羅性を考慮することが推奨される．

まとめ

ソフトウェア非機能要求の中でも品質要求事項は，漏れなく明確に定義することが容易ではない．しかし，これを定義し，要求水準を定めない限り，「品質の良いソフトウェア」として作り手がどのようなものを考えているのかを表明したり，評価したり，管理項目として開発プロセスを通じて評価したりすることはできない．一部の有能な技術者によって品質の良いソフトウェアが偶発的に生み出されることはあるが，そこに再現性を期待することは難しい．本稿が，現場の経験として蓄積された知識をいったん整理し，作り手が考える「顧客にとって品質の良いソフトウェア」とはどのようなものかを考えるきっかけとなれば幸いである．

参考文献

1) 飯塚悦功：現代品質管理総論，朝倉書店（2009）．
2) 狩野紀昭，瀬楽信彦，高橋文夫，辻　新一：魅力的品質と当たり前品質，品質管理学会，Vol.14, No.2, pp.147-156（1984）．
3) ISO/IEC 25010 : 2011, Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models (2011),（JIS X 25010 : 2013，システム及びソフトウェア製品の品質要求及び評価（SQuaRE）―システム及びソフトウェア品質モデル（2011）．
4) ISO/IEC 25030 : 2007, Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - Quality Requirements (2007), JIS X 25030 : 2012, システム及びソフトウェア製品の品質要求及び評価（SQuaRE）―品質要求事項（2007）．
5) 日経コンピュータ：セブンイレブンの研究：考えつくすための情報システム，2006 年 5 月 29 日号，日経 BP 社（2006）．
6) 経済産業省ソフトウェアメトリクス高度化プロジェクト，プロダクト品質メトリクス WG：システム／ソフトウェア製品の品質要求定義と品質評価のためのメトリクスに関する調査報告書，経済産業省（2011）．
7) 野中　誠，脇谷直子：ソフトウェア品質管理・品質保証実態調査 2012，ソフトウェア品質シンポジウム 2012（SQiP2012）講演資料，http://www.juse.or.jp/ sqip-sympo/2012/program/pdf/E3.pdf（2013 年 8 月 30 日アクセス）．

（2013 年 10 月 30 日受付）

● **野中　誠（正会員）** nonaka-m@toyo.jp

東洋大学経営学部経営学科准教授．2000 年早稲田大学大学院理工学研究科経営システム工学専攻単位取得退学．同大理工助手を経て現職．日科技連ソフトウェア品質委員会（SQiP）運営委員長．2013 年度日経品質管理文献賞受賞．

● **東　基衛（正会員）** az-mo@mtd.biglobe.ne.jp

現在早稲田大学理工学術院名誉教授，1963 年早稲田大学第一理工学部卒業，元日本電気（株）ソフトウェア生産技術研究所管理技術開発部長，1987 年より早稲田大学理工学部経営システム工学科教授．