第 9 章　ソフトウェア・メトリクス

ソフトウェア・メトリクスの目的

ソフトウェア工学の領域での重鎮の一人であるトム・デマルコ（Tom Demarco）は、その著書の中で「測定できないものは制御できない」と言った[DEM82]。

我々はソフトウェア開発のプロジェクトを成功に導きたいし、ソフトウェアの品質を良くしたい。さらにソフトウェア開発の生産性を、一層向上させたい。つまり我々はソフトウェア開発の過程などを制御して、より良いソフトウェアと、そのソフトウェアを開発するより良い方法を手に入れたい。だからデマルコのこの言葉が正しいとすれば、我々はソフトウェアそのものと、ソフトウェア開発に関わるいろんな側面についての計測を行わなければならない。

端的に言えば、これがソフトウェアに関するいろんな項目を測定すること、つまりソフトウェア・メトリクスの目的である。

すぐ後で述べるが、ソフトウェアの測定について JIS X 0141：2009（元の国際規格は ISO／IEC 15939：2007）という規格がある。この規格の解説の部分に次のような記述がある[JIS09a]。

「ソフトウェアの測定は、ソフトウェアプロセスとソフトウェア製品の管理と改善を支援する。測定は、ソフトウェアライフサイクル活動を管理し、プロジェクト計画の実行可能性を評価し、プロジェクト活動がそれらの計画に準拠しているかどうかを監視するための最も重要な道具である。ソフトウェアの測定は、ソフトウェア製品の品質及び組織のソフトウェアプロセス能力を評価するための鍵となるものである。（中略）

継続的に改善を進めるためには組織内が変化していかなければならない。変化を評価するためには測定が必要である。・・・」

この文章は、この JIS 規格の元になっている ISO／IEC 15939：2007 の冒頭の部分との断りがある。つまりこの部分は、当初の ISO 規格から JIS 規格を作成する際に、割愛されたものらしい。それはともかくとして、この文章はデマルコが端的に言ったことをもっと丁寧に述べている。つまりソフトウェアに関わる事項を測定することの必要性と重要性が、ここで明確に述べられている。

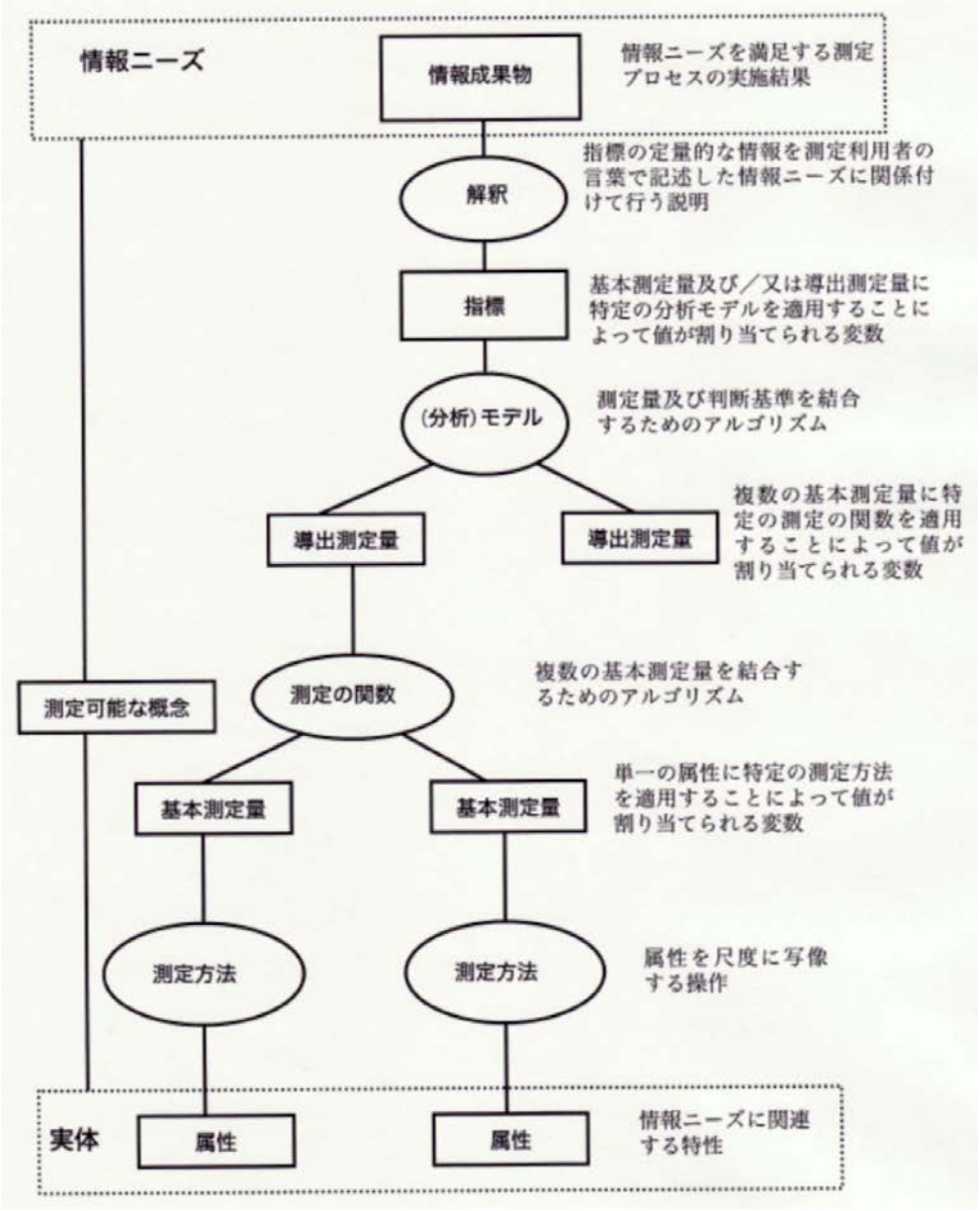
ソフトウェア・メトリクスに関わる国際規格

前述のソフトウェアの測定に関わる規格（JIS X 0141：2009、元の国際規格は ISO／IEC 15939：2007）には、「ソフトウェア測定プロセス」という標題が付いている。ソフトウェア・ライフサイクル・プロセスを規定した JIS X 0160：2012（元の国際規格は ISO／IEC 12207：2008）には、「測定」というプロセスが定義されている。JIS X 0141：2009 には何も書かれていないけれど、この規格は JIS X 0160：2007 の測定に関わる部分だけを抽出し、もっと詳細に記述したものと考えることができる[1]。

ソフトウェア・メトリクスの枠組み

[1] このような位置づけにある他の規格に、ソフトウェアの保守についての規格（JIS X 0161：2008（元の規格は ISO／IEC 14764：2006））がある。

その JIS X 0141：2009 に、ソフトウェア・メトリクスの全体の枠組みを記した図が掲載されている。それを、図表 9-1 として示す[JIS09a]。



図表 9-1　ソフトウェア・メトリクスの枠組み（[JIS09a]より）

直接何らかの方法によって測定できるものを、図表 9-1 では「基本測定量」と呼んでいる。

例えば、「テスト期間中に発見されたある特定のソフトウェアの累積の欠陥件数」と、「そのソフトウェアの規模」は、ともに基本測定量である。この最初の基本測定量を 2 つ目の基本測定量で割ると、「欠陥密度」という「導出測定量」が得られる。この欠陥密度の数字 1 つだけでは、まだ何らかの判断をするとか行動を起こすということはできない。しかしこの導出測定量

（欠陥密度）を、目標としていた数値と比較するとか、他のプロジェクトの数値と比較する[2]とかの形で加工／操作して、「このプロジェクトの欠陥密度」という「指標」を作成すると、何らかの判断や行動を起こす基になるデータが得られる。

つまりソフトウェア・メトリクスでは、この「指標」が重要な位置を占める。

それでは、指標として何が重要なのだろうか。そのために、何を測定すれば良いのだろうか。それを、次に考えてみたい。

何を測定するのか

ソフトウェア・メトリクスで、測定の対象になるものを「実体」と呼んでいる。この測定の対象になりうるものには、プロセス、製品、プロジェクトと資源の 4 つがある[JIS09a]。

この 4 つにはそれぞれ多くの属性があり、それらは全て測定可能である。しかしこれらを闇雲に測定し、組み合わせてみたところで、費用と手間がかかるばかりでたいした成果は得られそうにはない。それでは何を測定し、それらをどう組み合わせて、どういう「指標」を得ればよいのだろうか。

米国メリーランド大学のヴィクター・バシリ（Victor Vasili）教授は、ソフトウェア・メトリクスで測定する対象を決める手順について、次のような方法を提案している[BAS92]。

まず、組織として実現したい「目標（Goal）」を明確にする。次に、その目標を実現するためにこれからも努力を続けるとして、将来のある時点でその目標実現の状況を把握するために、つまりその目標がすでに達成されたのか、あるいは達成に向かって順調に進んでいる過程にあるのか、逆に足踏みをしたり、場合によれば後退していたりしていないのか、といったことを明確に認識するための「質問（Question）」を考える。最後に、この質問に答えるためのデータを前記の「指標」として明らかにし、その指標を得るために何を「測定値（Metrics）」として得ればよいかを考える。このアプローチを、バシリ教授「GQM パラダイム」と名付けた。

ここでバシリ教授は、ソフトウェア・メトリクスの測定項目を決めるためのアプローチは、ボトム・アップではなく、トップ・ダウンでなければならないと述べている。

測定の手順

ISO／IEC 12207：2008 に準拠して作成された「共通フレーム 2013」の管理プロセスの一部である「測定」のアクティビティには、ソフトウェアの測定のためのタスク（手順）として、以下の 3 つの作業が挙げられている[3][IPA13a]。

1. 測定の計画
2. 測定の実施
3. 測定値の評価

ここでは、すでに測定のための組織作りは終わっているという前提に立っている。組織／体制が確立された後は、個別に測定に関わる計画を立て、その計画に基づいて準備も含めて測定を実施し、その測定結果を分析し、評価し、報告／伝達し、さらに得られた結果を蓄積することになる。当然これらの活動は全て、適切に管理されていなければならない。

---

[2] 他のプロジェクトのデータと比較することなどを、ソフトウェア・ベンチマーキングという。ソフトウェア・ベンチマーキングについては、第 11 章で述べる。

[3] 共通フレーム 2013 などをベースにしたソフトウェア開発の作業については、その概略を第 12 章で述べる。

ここで、データ／情報の蓄積が重要であることを強調しておきたい。プロジェクトの活動は重要な測定対象の 1 つであるが、過去のプロジェクトの活動の結果は将来のプロジェクトの見積もりで、たいへん重要な役割を果たす[4]。

ガーネギー・メロン大学ソフトウェア工学研究所が策定した「開発のための CMMI（Capability Maturity Model Integration-DEV、能力成熟度モデル統合）」のバージョン 1.3 では、後述するようにレベル 2（「管理されている」レベル）のプロセス領域の 1 つに「測定と分析」がある。ここではこの作業のゴールと、そのゴールを達成するための作業（プラクティス）として、次のものが挙げられている[CMM10a]。

SG1　「測定と分析」活動を整合させる
- SP1.1　測定目標を確立する
- SP1.2　尺度を明記する
- SP1.3　データ収集手順と格納手順を明記する
- SP1.4　分析手順を明記する

SG2　測定結果を提供する
- SP2.1　測定データを獲得する
- SP2.2　測定データを分析する
- SP2.3　データと結果を格納する
- SP2.4　結果を伝達する

共通フレーム 2013 と CMMI-DEV v1.3 の間には、作業について齟齬はない。

なお、ここに「尺度」という言葉がある。この言葉についての JIS X 0141：2009 での定義は、以下の通りである[JIS09a]。

> 「尺度(scale)　連続的若しくは離散的な値の順序集合または分類の集合で、それに属性を対応付けるもの」

いささか分かりにくい定義だが､広辞苑第六版にはこの言葉は以下のように記載されている。

① 物の寸法を正確に測定するのに用いる具。木・金属などに目盛を刻んで製し、メートル尺・曲尺（かねじゃく）・鯨尺などがある。ものさし。「―をあてる」

② 広く、計量の標準。物事を評価するときの規準。「社会進歩の―」

ここでの「尺度」は、この広辞苑の 2 つ目の定義であると考えればよい。

測定に当たっての留意事項

ソフトウェアの測定を行うに当たって、注意しなければならないことが 3 つある。

その 1 つ目は、正確なデータを集めなければならないということである。測定には、測定誤差は避けられない。当然それを極力回避するべく測定の手順などを考えなければならないが、その測定誤差は仕方がないとして、不正確なデータが集まらないように注意することが重要である。例えばソフトウェア技術者の活動についてのデータをそのソフトウェア技術者から直接収集し、さらにその結果を人事考課に使用したりすると、正確なデータが集まらないと考えなければならない。ソフトウェアの測定の結果はソフトウェア・プロセスの改善や将来のプロジ

[4] プロジェクトの見積もりについては、第 52 章のソフトウェアのコストモデルについての話の中で述べる。

ェクト計画の立案などだけで使用し、人事考課などに使用することはどのようにして集めたデータであっても避けるべきである。特に、個人から集めたデータを個人の評価に使用するといったことは絶対に避けなければならない。

2 つ目は、可能な限りデータは自動的に集めることに心がけるべきである。人間は、間違いを犯す。自動化によって、その人間が犯す間違いを避けることができる。そのために適切なツールが必要になるが、それを入手するための手間と費用を惜しまない方が良い。

3 つ目は、測定のための費用も考慮して、その結果を有効に使用することである。前述の通り測定には、手間も費用もかかる。だから測定の結果はうまく活用できて、その手間と費用に見合ったものでなければならない。単なる記録のために測定を行う、というようなことは避けなければならない。つまり測定の結果は、我々のソフトウェアそのものとソフトウェアを開発する過程を改善するために有効に使われなければならない。かけることになる手間や費用と比較して結果があまり有効でないと思われる場合には、その測定は見合わせた方が良い。

ソフトウェア・メトリクスの位置づけ

前述の通りカーネギー・メロン大学のソフトウェア工学研究所が策定した「開発のための CMMI」のバージョン 1.3 には、レベル 2（「管理されている」レベル）のプロセス領域の 1 つに「測定と分析」があることはすでに述べた[5]。このことは、ソフトウェアの測定を行うこととその測定結果を分析することが、ソフトウェア開発におけるごく基本的な活動であることを示している[CMM10a]。

しかし、この CMMI バージョン 1.3 のレベル 4（「定量的に管理されている」レベル）のプロセス領域の 1 つに「定量的プロジェクト管理」があり、そこでは計測の結果を基にしてソフトウェアの品質とプロジェクトの実績が統計的に管理され、監視され、適切な是正処理が講じられる。このことから、ソフトウェア測定の奥行きは非常に深い、ということができる。

この CMMI の前身である SW-CMM（Software Capability Maturity Model、ソフトウェアに関わる能力成熟度モデル）の 1993 年に策定されたバージョン 1.1 には、このような形でソフトウェアの測定が定義されていなかった。また、ソフトウェアの測定は JIS X 0160：2007 の管理プロセスの中に規定されていると述べたが、その元になる JIS X 0160：1996 の管理プロセスの中には、これは規定されていない。つまりこの部分は、2002 年に発行されたこの基になる ISO の規格の修正票で追加された。

これらのことは、ソフトウェアの測定についての重要性の認識が 1995 年以降に急速に高まってきたことを意味している。

何を測定するのか・その 2

ソフトウェア・メトリクスで測定するべきものを「GQM パラダイム」の考え方で決めるのが良いと、すでに述べた。

ケーパース・ジョンズ（Capers Jones）はその著書の中で、もっと明確に次の 9 項目を測定の対象として挙げている「JON08」。

① 運用環境
② 開発中のプロジェクト
③ ソフトウェア資産とバックログ

---

[5] 「開発のための CMMI」については、第 40 章で述べる。

④　顧客満足度
⑤　完了プロジェクト[6]
⑥　ソフト要因
⑦　ソフトウェアの欠陥
⑧　企業内従業員統計
⑨　企業内の意識調査

この中の「⑥ ソフト要因」には、補足が必要かもしれない。ソフト要因とは、次のようなものを指す言葉として、ここでは使われている。

- ソフトウェアの開発と保守の手法
- ツール
- スキル
- 組織
- 環境

キィワード

ソフトウェア・メトリクス、基本測定量、導出測定量、指標、GQM パラダイム、尺度

略語

CMMI：Capability Maturity Model Integration
SW-CMM：Software Capability Maturity Model

規格

JIS X 0141 : 2009、ISO／IEC 15939 : 2007、ISO／IEC 12207 : 2008、JIS X 0160 : 2012、共通フレーム 2013、CMMI-DEV v1.3

人名

トム・デマルコ（Tom Demarco）、ヴィクター・バシリ（Victor Vasili）、ケーパース・ジョンズ（Capers Jones）

参考文献とリンク先

[BAS92] V. Basili, “Software Modeling and Measurement: The Goal/Question/Metric Paradigm,” University of Maryland, CS-TR-2956, UMIACS-TR-92-96, September 1992.
なお、この論文は次の URL からダウンロードできる（確認日：2017 年（平成 29 年）1 月 6 日）。
http://www.cs.umd.edu/~basili/publications/technical/T75.pdf

[CMM10a] CMMI 成果物チーム、「開発のためのCMMI® 1.3 版 CMMI-DEV, V1.23 CMU/SEI-2010-TR-033 ESC-TR-2010-033 より良い成果物のためのプロセス改善」、カーネギー・メロン大学ソフトウェア工学研究所、2010年
この資料は、次の URL からダウンロードできる（確認日：2017 年（平成 29 年）1 月 25

[6] 開発の生産性や開発費用などについては、②開発中のプロジェクト／⑤完了プロジェクトの中で取り上げられる。

日)。
http://cmmiinstitute.com/resource/japanese-language-translation-of-cmmi-for-development-v1-3/

[DEM82] Tom DeMarco 著、渡辺純一訳、「品質と生産性を重視した ソフトウェア開発プロジェクト技法 見積り・設計・テストの効果的な構造化」、近代科学社、1987 年.
この本の原書は、以下のものである。
Tom DeMarco, “Controlling Software Projects Management, Measurement & Estimation,” Yourdon Inc., 1982.

[IPA13a] 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および取引のために～ ソフトウェアライフサイクルプロセス 共通フレーム 2013」、(株) オーム社、平成 25 年.

[JIS09a] 日本工業標準調査会審議、「ソフトウェア測定プロセス JIS X 0141 : 2009 (ISO／IEC 15939 : 2007)」、日本規格協会、平成 21 年.

[JON08] Capers Jones 著、富野壽、小坂恭一監訳、「ソフトウェア開発の定量化手法 生産性と品質の向上を目指して 第 3 版」、構造計画研究所、2010 年.
この本の原書は、以下の通りである。
Capers Jones, “Applied Software Measurement: Global Analysis of Productivity and Quality Third Edition,” McGraw-Hill, 2008.

（2008 年（平成 20 年）9 月 14 日　初版作成）
（2014 年（平成 26 年）3 月 10 日　一部修正）