2006－EMB－1（6）
2006／6／27

POSIX と SQuaRE の対応付けの詳細化

ウ ダイキョウ(*)，小川　清，斉藤直希（**），桐山　清（*）

*中部大学,**名古屋市工業研究所,

Email:{ogawa.kiyoshi, saito.naoki}@nmiri.city.nagoya.jp, kiriyama@chubu.ac.jp

SQuaRE の評価プロセス定義に基づいて，POSIX の試験の視点の明確化を，POSIX と SQuaRE の対応付けの詳細化として行った。供給側と取得側の２つの視点に基づいて品質指標の副特性ごと仕様の例を示し，それぞれ試験仕様が定義できることを示した。

Detail mapping to POSIX and SQuaRE quality sub-characteristics and view points of developers and acquirers

Daikyo U, Kiyoshi OGAWA, Naoki SAITO, Kiyoshi KIRIYAMA

ABSTRACT: On SQuaRE evaluation process, POSIX test is clarified as a detailed mapping of POSIX and SQuaRE. It is proposed that from 2 points of supplier and obtainer, we can define each test specification and that from a mapping of POSIX and SQuaRE, openness should be added as sub-characteristics.

１　はじめに

組込みシステムで OS を採用するかどうかを判断するための OS の試験について検討する。特に、組込みシステムでソフトウェアの品質の向上を目的として OS を採用する場合を考える。OS を利用した場合に，製品全体で品質を高くするには，OS の品質も高い必要がある。

OS のような個別のソフトウェア製品の品質規格は，従来，あまり提供されて来なかった。POSIX[6]と SQuaRE[3]については，POSIX の Test ガイド[7]における考え方と，SQuaRE との関係を３×３の試験マトリックスとして提起したものがある[1][4]。本研究では，POSIX と，SQuaRE の評価の枠組みに従い，ソフトウェアの供給者，利用者の視点での OS に対する品質副特性を詳細化した。対象としては公開の Test Suite[5] がある POSIX を想定し，品質指標として国際規格である ISO/IEC 25000 シリーズの SQuaRE を枠組みとして利用する。第２章で POSIX 及び C 言語について考察し，第３章で SQuaRE における評価の枠組みと試験仕様の例を示し，第４章で Test Suite についての検討を示し，第５章で今後の課題を示す。

２　POSIX

組込みで利用している OS で仕様を公開しているものには ITRON，自動車向けの OSEK，Linux がある。また，国際規格としては，POSIX がある。POSIX は可搬 OS の国際標準である。Linux が国際標準になるにあたっても POSIX との整合性を考慮している[11]。

OS の試験を行うためには，前提としてハードウェアと開発言語の試験が済んでいることが重要である。メモリが壊れているのに，ソフトウェアの試験を始めて，壊れたところに関係するところで止まってからハードウェアを調べるのは煩雑である。最初に，ハードウェアの試験を実施するとよい。ハードウェアの試験には，ハードウェア測定機器を用いる場合と，ソフトウェアを利用する場合と，両方の組みあわせの場合がある。

次に OS を実装している言語の試験がある。C 言語で実装している場合は，C 言語の試験が必要である。POSIX では C 言語に適合していることを前提としている。C 言語の実装が，OS を記記述する上で十分でなければ，そこを補う必要がある。また，C 言語では実装定義の事項があるが，OS の実装と実装定義との間に矛盾がないかを確認するとよい場合がある。

３　SQuaRE における評価の枠組み

SQuaRE は，ソフトウェアの品質の国際標準でソフトウェア製品の基本的な枠組みを提供している。ここでは，改訂後に SQuaRE に統合予定のソフトウェア製品の評価に関する ISO/IEC 14598[8][9]を含む。ISO/IEC 14598 では，取得者，供給者，試験機関における手順を示している。また，ソフトウェアパッケージ品質要求事項及び試験の ISO/IEC 12119 は，SQuaRE シリーズとして改定[10]している。そこで，

これらの規格の主旨をもとに，POSIX Test Suite を組み込み Linux で利用するためクロス開発環境において試験するとともに，具体的に企業の供給側の試験要件，調達側の試験要件について，試験プログラムを拡張しながら検討を行っている。OS を実現する側にとっての品質特性と，OS を利用する側にとっての品質特性を，同じ特性ごとに対応をつけ，それぞれに対する試験仕様を検討している。

３．１　機能性

表１に，機能性の４つの副特性ごとに，OS を実現する側と，OS を利用する側での視点を示す。

表１　機能性の品質副特性

特性	副	OS を実現する側	OS を利用する側
機能性	合目的性	OS が目的を要件として定義していることが必要。その目的に適合しているかどうか。	OS を利用する際に，そのアプリの目的に対して，そのOS が目的に合っているかどうか。
	正確性	OS が要件として確認可能なデータ，時刻などを定義していることが必要。それらの値が正確性かどうか。	アプリが利用する OS の機能がどこまで正確性を保証しているか。
	相互運用性	２つのOS を，同時に一つの CPU 上で動かす場合と，ネットワーク上の２つのOS が相互に運用できるかの２つのどちらか又は両方。さらに，複数のアプリが同時に動作することができるようにする。	複数のアプリを OS が同時利用できるようにしているか。複数のアプリが相互にデータ共有または通信できるか。
	セキュリティ	OS にセキュリティ機能を実装する。また，セキュリティ機能のない I/O 機能を実装しない。	OS のセキュリティ機能を利用するか，アプリでセキュリティ機能を実装するか。

合目的性は，OS の目的が何かを明確にしているか，どういう OS を狙っているかによる。例えば，Portable であることが重要であれば，可搬性を参照する。Open であることが重要であれば，SQuaRE の副特性にはないが，公開性を定義して要件にするとよい。小さいことが重要であれば，資源効率性を参照する。速いことが重要であれば，時間効率性を参照する。

OS の正確性を確認するには，C 言語がデータの正確性をどこまで保証しているか，CPU が正確性をどこまで保証しているかをまず確認するとよい。通信機能がある場合には，通信結果の正確性を含めることができる。

それぞれの副特性に対する，OS を実現する側と OS を利用する側の試験仕様を表 2 に例示する。

表２　機能性の品質副特性の試験仕様

特性	副	OS を実現する側の試験仕様	OS を利用する側の試験仕様
機能性	合目的性	OS の目的を明確にし，その目的にあう試験を行う。API を提供するだけが OS の目的であれば，API の試験のみを行う。	アプリごとの目的を実現する上で必要な OS の機能，制約が目的にあっているかを試験する。
	正確性	CPU の正確性の試験結果，C 言語の正確性の試験結果に基づき，OS で正確性を保証する上で必要な試験を実施する。	アプリが必要とする正確性を，OS がどの程度提供しているかを確認する。
	相互運用性	２つの OS を同時に１つの CPU で動かす場合は，主たる OS がある場合には，その主たる OS に試験機能をつける。あるいは，別々に試験機能をつけて試験する。	複数アプリの仕様の上限値，下限値に基づいて，OS 上で同時に利用して，必要な結果が得られるかを確認する。複数のアプリが，仕様に基づいて，相互にデータの共有または通信ができるかを試験する。
	セキュリティ	OS のセキュリティ機能を試験する。OS にセキュリティ機能がない機能がないかを試験する。	アプリが利用する OS のセキュリティ機能を，そのアプリの仕様に基づいて試験する。

３．２　信頼性

表 3 に，信頼性の３つの副特性ごとに，OS を実現する側と，OS を利用する側での視点を示す。

表３　信頼性の品質副特性

特性	副	OS を実現する側	OS を利用する側
信頼性	成熟性	想定外の入力に対して，障害を発生しない。	想定外の入力に対して，アプリが正常動作をするかどうか。
	障害許容性	OS で定義している障害と，その障害のうち，どの障害は許容し，どの障害は OS の終了または異常処理	OS が定義している障害の許容性が，OS を利用する上で妥当でない場合には，アプリでその許容性を変更できるようにする

		へ行くか。	か。
	回復性	障害からの異常処理のうち，何を回復するか。	障害からの回復の際に，OS が定義している回復性が妥当でない場合には，アプリでその回復性を変更できるかどうか。

３．３効率性

表 4 に，効率性の２つの副特性ごとに，OS を実現する側と，OS を利用する側での視点を示す。

表４　効率性の品質副特性

特性	副特性	OS を実現する側	OS を利用する側
効率性	時間効率性	OS が持つ機能の時間効率を提示する。	OS が持つ時間効率が，アプリで利用するのに十分か。
	資源効率性	OS が利用する資源の効率性と，OS があることによる資源の効率性	OS が持つ資源効率性が，アプリで利用するのに十分か。

時間効率性，資源効率性は，測定プログラムが時間と資源を消費するため，ソフトウェアだけの測定では十分な試験ができない場合がある。

３．４　保守性

表 5 に，保守性の４つの副特性ごとに，OS を実現する側と，OS を利用する側での視点を示す。

表５　保守性の品質副特性

特性	副	OS を実現する側	OS を利用する側
保守性	解析性	障害発生時に解析可能な機能を入れておくか。	障害発生時にアプリで解析可能な機能が利用できるか。
	変更性	OS が定義している事項を，変更する際に容易かどうか。	OS の変更がアプリの変更を必要がないようにできているか。
	安定性	OS が定義している事項で，改訂する場合に影響を受ける部分を規定しておく。	OS の変更でアプリが影響を受けないような仕組みがあるか。
	試験性	変更した部分に対して，試験可能な仕組みがある。	OS の変更に対して，アプリへの影響が測定できる。

３．５　可搬性

表 6 に，河搬性の３つの副特性ごとに，OS を実現する側と，OS を利用する側での視点を示す。

表６　可搬性の品質副特性

特性	副	OS を実現する側	OS を利用する側
可搬性	環境適合性	OS が，異なるハードウェアにうまく適合するか。	異なるハードウェアの OS 上で，アプリが同じように動作するか。
	設置性	同じ機能の C 言語が容易に設置できるか。C 言語に依存しない部分の OS の設置性がよいか。	異なるハードウェアの OS 上で，アプリに変更が必要となる割合。
	置換性	OS の版を変えても，ドライバがそのまま動くか，アプリがそのまま動くか。	OS の版を変えても，アプリがそのまま動くか。

ＰＯＳＩＸは，ＰＯＳＩＸ自体の設置性が高いことは主張していない。Ｃ言語が存在していることを前提としている。C 言語に依存しない部分の設置性がよいか。

３．６　使用性

表 7 に，使用性の４つの副特性ごとに，OS を実現する側と，OS を利用する側での視点を示す。

表７　使用生の品質副特性

特性	副	OS を実現する側	OS を利用する側
使用性	理解性	OS のソースコードの理解性。OS でその OS の API を利用している場合は，OS の API 機能の理解性を含む。OS のユーザインタフェースとしての GUI またはコマンドインタプリタの理解性を含む。	OS の API 機能の理解性と，開発にその OS を利用していれば，その OS のユーザインタフェースとしての GUI またはコマンドインタプリタの理解性を含む
	習得性	OS の移植のための習得性，OS の新しい版を作る上での習得性	OS の改造のための習得性と，OS の試験のための習得性と，OS 上のアプリを開発するための習得性と，OS の操作の習得性
	運用性	OS を運用するために必要な機能があるか。	アプリの運用が，OS の運用機能を利用できるか，アプリが独自に運用機能を作りこむか。
	魅力性	OS がアプリを作る上で魅力があるか，OS をそのまま使う上で魅力があるか。	アプリを実現する上で，OS が提供している機能が魅力的かどうか。

３．７　利用時の品質

利用時の品質は，OS を利用する側の品質のように考えられることがある。OS を実現する側にとっては，利用時の品質を確保するための仕組みを作りこむことを検討するとよい。

表８　利用時の品質

特性	OS を実現する側	OS を利用する側
利用時の品質	OS が利用時に，どのような品質で測定される可能性があるかを想定し，利用時の品質を確保できる機能を明確にする。	OS を利用する時に，どのような品質が重要になるかを明確にし，それを満たすのをアプリで実現できるか。

４　POSIX　Test Suite

POSIX には，API の公開の Test Suite がある。Test ガイドが 1999 年版であるため，POSIX の 1990 年版に基づいた Test　Suite を利用している。クロス開発環境で用いているコンパイラには，ISO/IEC 9899:1999 年版への対応が明示されておらず，ISO/IEC 9899:1990 への対応を示しているものもある。POSIX:1990 は C:1990 対応である。C 言語が動作し，POSIX の API が定義されていれば，試験をすることが可能である。例えば，Windows では，cygwin 上の GCC でコンパイルし，実験することができる。クロス開発環境では，セルフ環境での作業とクロス環境での作業を明確区分すれば利用可能である。例えば，ディスクのない環境で，NFS をマウントして開発環境側のファイルシステムを利用して，クロス開発側の試験を行うことが可能である。規格に適合していることが重要ではない場合は，デバッグ過程として利用することができる。クロス開発の過程で機能が失われているか，対象システムの品質に影響を与えないかを確認できる。

次に，オープンソース OS の目的として可搬性と公開性を検討する。POSIX Test Suite では，公開性，可搬性の試験項目は見当たらない。ソースコードが公開されていることは公開性の一つの指標であり，複数の CPU に移植がされていることは可搬性の一つの指標である。OS の Test Suite を公開し，それが可搬であれば，OS の公開性の推進と可搬性の確認に役立つと考えることができる。このように、試験プログラムとしてではなく文書として示す必要がある品質指標も存在する。また，公開性は，理解性，運用性に影響を与え，結果として魅力性があるため，使用性が高いと考えることができる。また，解析性，変更性，試験性が高いため，保守性が高いと考えることができる。そのため公開性を，保守性の一部として検討できる。

５ 今後の課題

POSIX と SquaRE の枠組みに基づく仕様の詳細化とその試験仕様について検討した。具体的な試験は，Linux, Windows 上の cygwin, Macintosh OS X の開発側の OS と，SH4 ボードのようなクロス開発環境の対象ボードでの Linux の双方で試験している。今後の課題としては，ハードウェアの試験でソフトウェアを利用するものと、C 言語の試験の双方で公開されているソフトウェアに基づいて，OS の試験に役立てることを検討していきたい。

参考文献

[1] 小川清，斉藤直希，渡部謹二，真鍋孝顕，馬場雄二，藤本智明，松良　彰，山名明弘，小浜　徹，加藤秀也，組込み Linux の試験と検証，情報処理学会全国大会，2005.3

[2] 山名明弘,小川清，斉藤直希，真鍋孝顕，馬場雄二，藤本智明，松良　彰,小浜　徹，加藤秀也，組込み OS の試験と検証，電気関係学会東支部連合大会,2004.9

[3] ISO/IEC 25000:2005 Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide tOSQuaRE

[4] Kiriyama Kiyoshi, Ogawa Kiyoshi, Saito Naoki, Baba Yuuji, Matsura Akira, Yamana Akihiro, Testing and Verification for Embedded Linux, 3WSQC, German, 2005.9

[5] PCTS:151-2, POSIX Test Suite, NIST, 1999, http://www.itl.nist.gov/div897/ctg/posix_form.htm

[6] ISO/IEC 9945-1:2003, Information technology - Portable Operating System Interface (POSIX) - Part 1: Base Definitions

[7] ISO/IEC 13210:1999, Information technology - Requirements and Guidelines for Test Methods Specifications and Test Method Implementations for Measuring Conformance to POSIX Standards

[8] ISO/IEC 14598-3:2000 Software engineering Product evaluationPart3: Process for developers

[9] ISO/IEC 14598-4:1999 Software engineering -Product evaluation Part4: Process for acquirers

[10] ISO/IEC 25051:2006 Software engineering Software product Quality Requirements and Evaluation (SQuaRE) Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing

[11] ISO/IEC DTR 24715 Technical Report on the Conflicts between the ISO/IEC 9945 (POSIX®) Standard and the Linux Standard Base Specification