ソフトウェア工学の共通問題は身近なものからボトムアップに作ろう

権　藤　克　彦[†1]

このポジションペーパーではソフトウェア工学研究の共通問題に対する私見を述べる．

1. はじめに

著者はこれまでのウインターワークショップ「ソフトウェア工学の共通問題」セッションにおいて，良い共通問題の構築は本質的に難しく，実現が難しいことを述べた[1)2)]．しかし難しいと言っているだけでは何も実現できないので，ここでは逆に著者にとって身近な例題からボトムアップに例題を作ることを考える．偉大な画家もいきなり大作を描くのではなく，数多くの習作を描くことが当たり前であり，ソフトウェア工学の共通問題でも同様と考えるからである．

主な主張は次の通り．

- 共通問題は「当たり前」のことが対象でも良い．
- 類似する数多くの問題を共通化するアプローチを取る．
- 比較尺度として十分に使えないことを恐れない．ただし外枠と欠点は明示する．
- いわゆる「やらせ」を使うことも検討する．

2. 共通問題は「当たり前」のことが対象でも良い．

例えば、共通問題は「時間を処理するAPI」でも良いと考える．この問題は一見簡単そうだが、時間を表現するデータ構造はPOSIXの場合，`time_t`，`struct timeval`，`struct timespec`など複数あり[3)]，必ずしも簡単では無い．これは，処理内容に応じて適切なデータ構造が異なるからである．さらに，ローカルタイム，多言語化，うるう秒の処理，2000年問題など，スマートな解法が自明ではない問題をこの問題は含んでいる．エラー処理や移植性の確保など，一般的な問題も考慮するとさらに複雑になる．つまり，良くありそうな「当たり前」の共通問題でも，十分な複雑さを含んでいる．

また，良くありそうな「当たり前」の共通問題には，次の利点もある．

- 製品レベルの実装がすでにあるので，それとの比較が可能である．
- これまでの経緯により，何が難しいかがある程度明らかになっているので,「この問題を解決できたか？」と解答例を評価しやすい．

3. 類似する数多くの問題を共通化するアプローチを取る．

著者が行った教育用コンパイラや教育用OSの研究[4)5)]で驚いたことは，類似しているが細部が異なる実装が非常に数多く存在することである．

これらを比較するための共通問題は容易に作れるだろうか? 例えば，POSIXがAPIの定義で構成されているので，教育用OSの仕様として小さなAPIを与え，それ以外に「プロセスはタイマー割り込みでプリエンプションすること」「各プロセスのメモリ空間は仮想メモリで独立させること」などを仕様として与えれば，比較的容易に共通問題を作成できそうに一見みえる．しかし，実際はやはり難しい．それは

- 教育対象に仮定するスキル
- 何に重点をおくか（何を本質ととらえるか）
- 抽象レベルをどこにおくか（単純な仮想マシンで良いのか，i386などの実機を対象とするのかなど）
- どの機能を仕様に含めるべきか（どの程度リッチなファイルシステムを実装させるべきか）

などにより，自明では無い差異が生じるからである．

この問題に対しては，プロダクトライン開発的なアプローチを適用し，何がコアアセットなのか，どんなバリエーションがありえるのかを地道に整理すれば，

†1 東京工業大学
Tokyo Institute of Technology

（ある程度はであるが）共通問題化は可能であると考える．

つまり，ブレインストーミングのように，類似を恐れずに多くの共通例題をまずは作り，それを分析して取捨選択・共通化するアプローチを取ると良いのではないか．

4. 比較尺度として十分に使えないことを恐れない．ただし外枠と欠点は明示する．

かつて筆者が担当していたソフトウェア工学関係の実験課題は「スケジュール帳」であった．これは上流から下流を含み，具体的に「どのようなスケジュール帳を実装するか」は学生には与えていなかった．これは要件定義や仕様書の作成自体が課題の対象だったからである．

この例から分かるとおり，上流工程の共通問題では，その要件定義や仕様にあいまいな部分が残らざるを得ない部分があり，その結果，解答例を比較する際に十分な比較尺度として使えない可能性がある．

ここでの提案は「それはあまり気にしなくていいのではないか？」ということである．ただし，

- 許されるスケジュールやリソース（つまり外枠）をある程度，きちんと決めること．
- 開発後にポストモーテムを行って，何が比較できなかったか（共通問題の欠点）を分析すること．

は（可能な範囲で）やるべきと考える．この事例が集まれば，共通化も可能となるだろう．

5. いわゆる「やらせ」を使うことも検討する．

以前のポジションペーパー[1]で，デスマーチ自体を研究対象にすることが難しいことを述べた．共通問題にデスマーチ自体を含めることもいろいろな困難がありそうだが，比較的安価にこれを達成するにはどうすればいいだろうか．

ここでは（ある意味で無責任に）意図的にトラブルを混入することを提案する．書籍「熊とワルツを」[6]によれば，ソフトウェア開発のコアリスクは次の５つである．

- 内在的なスケジュールの欠陥
- 要求の増大（要求の変更）
- 人員の離脱
- 仕様の崩壊
- 生産性の低迷

これに対して，例えば，複数のステークホルダーが「仕様には○○を含めるべきだ」と意図的に敵対する議論を行えば，「要求の増大（要求の変更）」や「仕様の崩壊」を演出できる．

もちろん，このような実験には倫理上の問題が生じるため，（広義の）疫学研究に関する倫理指針を遵守する必要があるため，決して容易ではない．しかし，本当のデスマーチよりもコストは安くつく．防災訓練はリアルではないが，リアルではないからと言って，防災訓練に意味がないわけではない．疑似デスマーチも同様と考える．

6. おわりに

本稿ではソフトウェア工学の共通問題は身近なものからボトムアップに作るべきであることを，４つの観点から述べた．

参考文献

1) 権藤克彦：ソフトウェア工学研究の評価に対する私見，情報処理学会ウインターワークショップ2011・イン・修善寺, pp.139–140 (2011)
2) 権藤克彦：共通問題は良い成功事例と失敗事例を集めて作るべき，情報処理学会ウインターワークショップ2012・イン・琵琶湖, pp.131–132 (2012)
3) Josh Carter：プログラマのためのサバイバルマニュアル，ISBN：978-4-87311-571-9 (2012)
4) 権藤克彦, 大場勝: 中レベル抽象・薄い中間層・追跡性の実践によるコンパクトな教育用オペレーティングシステム udos の設計と実装, 電子情報通信学会論文誌，J90-D[5], pp.1194-1208 (2007)
5) ネイティブアセンブリコードを出力する教育用コンパイラ (XCC) と，水平スライスが可能な可視化ツール (MieruCompiler)，権藤克彦，福安 直樹，荒堀 喜貴，電子情報通信学会論文誌，vol.J95-D, No.5, pp.1225-1241 (2012)
6) トム・デマルコ，ティモシー・リスター:熊とワルツを－リスクを愉しむプロジェクト管理，ISBN-13: 978-4822281861 (2003)