3J−8　異なるOS間での効率的なソフトウェア移植技術

阿川 雄資　若野 勝己　藤原 英二

NTT情報通信網研究所

1　はじめに

近年のソフトウェアの高機能化・大規模化に伴い、生産性の高いソフトウェア開発が望まれている。そのための一手法として、既存のソフトウェアを異なった動作環境のOS下に移植し、再利用することにより生産性を上げる方法が有効であり、そのための効率的な移植技法の確立が重要となっている。

本稿では、移植の際問題となるOS間のインタフェース差／機能差を吸収するソフトウェアを開発することにより、移植対象ソフトウェアの本体を改造しない効率的な移植技法について提案する。

2　ソフトウェア生産性

生産性の高いソフトウェア開発のための工学的な手法として、CASEなどのソフトウェアツールや、第4世代言語（4GL）、またソフトウェア自身の高い生産性、すなわち再利用性までを考慮したOOA/OOD等のオブジェクト指向技術を用いる方法などが提案されている。

これらが、主に新規ソフトウェア開発における生産性の向上手法であるのに対し、現在、あるプラットホーム下で動作しているソフトウェアを、UNIXやCTRON (Communication and Central TRON)等の異なった動作環境のOS下に移植し、同様の機能を異なる環境下で実現することにより、生産性を上げる方法なども考えられる。この移植による生産性の向上は、既存のAP資源の有効利用だけでなく、今後のソフトウェア大規模化に伴い、高品質なソフトウェアの維持のための有効的な手法となる。

3　ミドルソフトウェア

ソフトウェアの大規模化に伴い、AP開発において様々な知識が要求され、開発者にとって大きな負担となっている。これを軽減するために、データベースや通信、GUIといったAPに共通に必要とされる機能を持つインタフェースをOS上であらかじめ提供し、AP開発者は、これらを利用することによってAP開発の生産性を向上させる方法がある。本稿では、これらのAPとOSの間に位置するソフトウェアをミドルソフトウェアと呼ぶことにする。

ミドルソフトウェアは、「特定の機能を持ち、OSの上位に位置し、直接ユーザに対するインタフェースを持たず、APや他のミドルソフトウェアに対してその機能を提供するソフトウェア」と定義することができる。ミドルソフトウェアには、CTRONの拡張OSやUNIXのXlibのようにタスクやプロセスで動作するものや、OSF/Motif GUIのようにライブラリとして提供されるものなどがある。また、複数のミドルソフトウェアが、同一環境に同時に存在することも考えられる。

今後、このミドルソフトウェアがさらに高機能化し、AP開発における重要な役割を果たすと考えられる。それに伴い、さまざまなプラットホームでミドルソフトウェアのインタフェースが必要とされ、そのためのミドルソフトウェアの効率的な移植が必要とされる。

4　ソフトウェア移植技術

APやミドルソフトウェアを移植するための方法としては、ソフトウェア本体を改造する、すなわち移植対象ソフトウェアの使用しているOSの機能をターゲットOSの機能にマッピングしたり、ターゲットOSに適したプログラム構造／制御構造に改良する方法が考えられる。しかしそのためには、移植する毎に多くの工数、開発規模が必要となり、またターゲットOSに対する高度の知識が必要とされる。OSのインタフェースを統一することにより、ソフトウェア本体の移植性を高める方法[1]も考えられるが、既存のソフトウェアの移植という観点から考えると、現実的に困難である。

本稿では、効率的な移植手法として、移植の際問題となる、APとそのAPが必要とするミドルソフトウェア、OSの各相互間のインタフェースの整合性をとるスライムソフトウェアをターゲット環境毎に開発し、これ

An Effective Software Porting Technique among Heterogeneous Operating Systems.
Yuji AGAWA,Masaki WAKANO and Eiji FUJIHARA
NTT Network Information Systems Laboratories

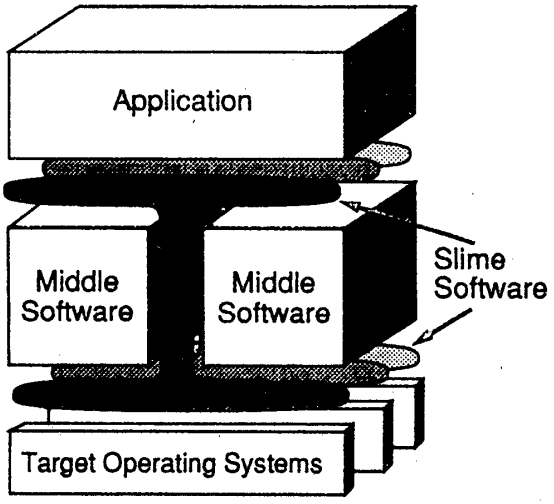

図 1: スライムソフトウェアによる移植

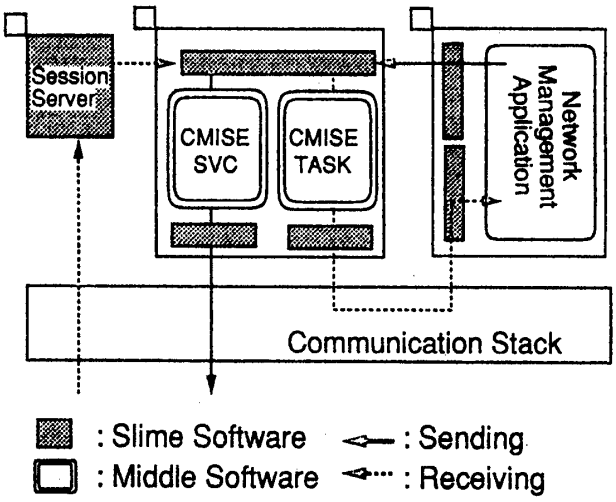

図 2: UNIX 上の CMISE の動作環境

をライブラリとして提供することにより、移植対象ソフトウェア本体の改造を行なうことなく、リコンパイルのみで移植を可能とする手法を提案する。図 1 において、各ターゲット OS とミドルソフトウェア間、ミドルソフトウェアと AP 間にスライムソフトウェアを準備し、ミドルソフトウェアや AP はスライムソフトウェアの提供するインタフェースを用いることにより、本体の改造なしで異なる環境下で動作する。スライムソフトウェアの持つ機能としては、インタフェース間の整合の他に、上位ソフトウェアに提供するインタフェースを制限する機能が考えられる。これにより、上位の AP やミドルソフトウェアに対して、それらの様々な要求に応じた柔軟で自由度の高いインタフェースの提供が可能となり、効率的なソフトウェア構築が可能となる。

5 通信ソフトウェアの移植

実際に提案手法を用いて、CTRON 上の通信ソフトウェアである OSI ネットワーク管理プロトコル CMISE (Common Management Information Service Element)[2] を UNIX に移植した。CMISE は、CTRON 拡張 OS として位置付けられ、前述のミドルソフトウェアと考えることができる。移植に際し、予め CMISE の使用する CTRON システムコールを洗い出し、最小限の機能を UNIX 上で提供することで、スライムソフトウェアによるオーバヘッドを減らすこととした。UNIX 上での CMISE の動作イメージを図 2 に示す。

6 評価・検討

移植対象としているソフトウェアを、本体の改造なしで異なる OS 環境下に移植することで、ソフトウェアの流通性が大幅に向上し、また開発工数の削減も期待できる。その反面、スライムソフトウェアによるオーバヘッドが性能面で問題となる。今回 UNIX 上に移植された CMISE の性能については現在計測中であるが、主なオーバヘッドとなる、2 つのミドルソフトウェア間の制御の振り分け部における処理時間は、最大で全体の処理時間の 1／5 以下であり、実用上には特に問題ないと考える。また、移植規模は CMISE 全体の 8 %程度であり、CMISE 本体の流通性や、スライムソフトウェアの流用性を考慮すると、十分実用的な範囲内であると考える。

今後、移植を行なう上での基本的に異なる OS の機能、例えばリアルタイム性などについて、検討する必要がある。

7 おわりに

今回提案した移植手法は、ソフトウェアの構成形態として捉えれば、OS のインタフェース規定上に作成されたソフトウェアに比べ、ソフトウェアの拡張性、移植性に優れており、今後のソフトウェア構成技術としても有効な手法と考えられる。

【参考文献】

[1] M. Wakano and N. Sugiyama, "A Study on the Portability of CTRON FTAM-CCL and CMISE-CCL interfaces," *Proc. of the Ninth TRON Project Symposium, IEEE Computer Society Press*, 1992.

[2] ISO/IEC ISP 11183, "Information Technology-International Standardized Profiles AOM1n OSI Management-Management Communications," 1992.