（2）持続的発展可能システム

欧州研究所では，サービスやプラットフォームの進化を柔軟かつ効率的に行うことを可能とし，同時に，ユーザ受容性も高める持続的発展可能システムを目指している．その実現に向け，再構成技術，サービス適応技術および適応ユーザインタフェース技術の研究を行っている．

河村 克也　John Hamard　Robert Hirschfeld
巳之口 淳　Bertrand Souville

1. まえがき

将来の移動通信網では，異種アクセス網が収容されるのに加え，遍在する大小多数のコンピュータ，センサノード，組込機器が，アドホックネットワークやセンサネットワークを介して階層的に統合されると考えられる．そしてユーザに対しては，高い可用性と品質を持つサービスへのシームレスで安全なアクセスの保証が求められる．また，さまざまなサービス提供者とのかかわりによりサービスの豊かさが促進され，さらにそれらサービスを十分に活用できるユーザインタフェースの導入が期待される．

将来の移動通信網では，さまざまなリソースが最適に利用され，システムやサービスが分権的に発展できることが望ましい．

例えば，異種アクセス網間の無線リソースや計算機リソースの配分は最適化されるべきであり，新規サービスと既存サービスとの間に共通の構成要素があれば，それは再利用されるのが効率的である．また，ユーザが豊かなサービスを容易に享受できるための検討が重要となる．システムやサービスの分権的な発展を促進するためには，個々の構成要素を異なるペースで向上させられる仕組みであることが望ましい．これは，異種のシステムやサービスの混在の度合いを深め，一方のシステムには予期せぬ変化として見える場合もあるが，その変化に適応し，十分に活用できることが望ましい．

上記移動通信網の実現には，以下の技術課題を解決する必要がある．

- 異種アクセス網環境での無線リソースや計算機リソースの効率的運用を，ソフトウェアの変更により柔軟に行うこと．
- サービスやプラットフォームの個々の構成要素を，そのいくつかは再利用しながら，異なるペースで向上させること．予期せぬ変化には，ソフトウェア実行時に対応できること．
- ユーザが負担を感じることなく，複数のサービスと同時に，あるいは各々に切り替えながら，相互作用できること．

本稿では，上記の課題に対する新しい解決法，すなわち，再構成技術，サービス適応技術および適応ユーザインタフェース技術について述べる．

2. 再構成技術（reconfigurability）

再構成可能な移動端末は，ソフトウェアの更新により，異種アクセス網環境下で最適な無線インタフェースを選択し適応することができ，異種アクセス網間での移動性を確保し，ネットワークの負荷状況やサービスの提供状況の変化に適応することで，ユーザの要望に柔軟に応えられる．

再構成可能な基地局は，最適な無線インタフェースと関連するリソースの配分を適応的に選択し，最適化させることで，異種アクセス網の効率的運用を可能とする．これは，移動通信事業者が将来の移動通信網を導入展開する際の異種アクセス網の同時運用にも効果的な技術である．

本研究では，再構成技術を構成する要素として，移動端末へのより効率的で，より安全なソフトウェア配布更新の仕組みを開発している．

ソフトウェア配布を行う際には，移動通信網の無線リソースの希少性を考慮する必要がある．ソフトウェア更新を行う際には，複数の移動端末への一括同時更新が必要となる場合がある．また，ソフトウェア配布更新に伴うセキュリティの確保は，各レイヤにまたがる重要な課題である．

これらの課題については，移動通信網とアドホックネットワークがシームレスに統合されている状況を仮定して検討している．ソフトウェア配布用のローカルサーバの考えを導入し[1]，アドホックネットワークの利用による効率的なソフトウェア配布の提案を行ってきた．複数の移動端末へのソフトウェアの一括同時更新を実現する方法として，更新失敗時の切戻しを含めた，タイムアウト型のアルゴリズム[2]を開発した．なお，本研究では再構成技術を検証するための試験環境も構築しており，このアルゴリズムも実装し動作を確認した．セキュリティに関しては，Webサービスを利用した安全なソフトウェア配布の仕組みを提案し，本試験環境に実装した．

図1は上記のタイムアウト型ソフトウェア一括同時更新アルゴリズムを示している．ここでは，再構成時あるいは

異なる構成を持つノードは互いに通信できない場合があるという課題の解決方法を示している．その例としてアドホックネットワークのルーティングプロトコルの変更を取り上げている．複数の移動端末間で新ソフトウェアが配布済みであり，なおかつ，再構成制限時間Tが交渉済みであるという状態から，図1のフローチャートに示すアルゴリズムが各移動端末上で動作する．こういったいくつかのシナリオの下で複数移動端末間のソフトウェアの整合性が確保されることを確認した[2]．

次の段階では，現在までに開発したソフトウェア配布更新の仕組みを拡張し，次章で述べるサービス適応技術に適したものとする予定である．

欧州研究所はまた，再構成技術を活用する将来の移動通信網の実現に向け，その要求条件やアーキテクチャ全体の検討へ貢献することを目的として，欧州研究計画の共同研究プロジェクトにも参加している．

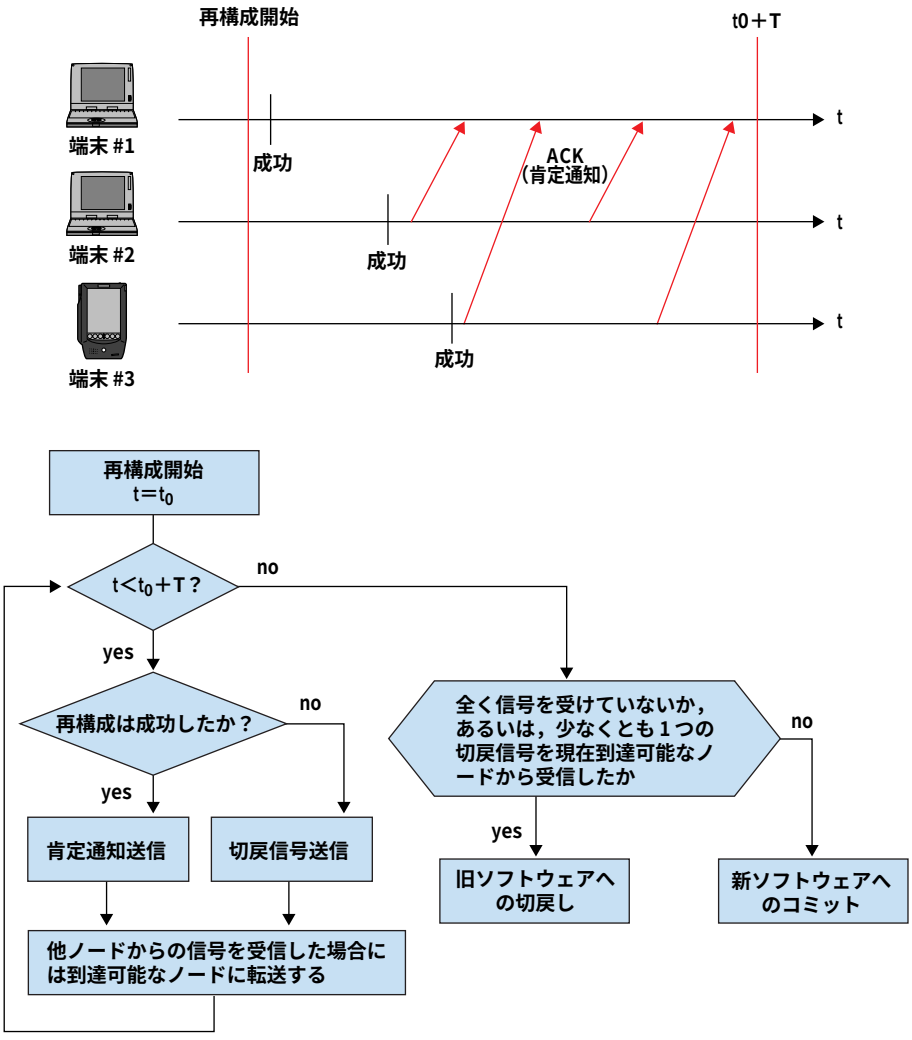

図1　タイムアウト型再構成制御シナリオとアルゴリズム（ルーティングプロトコルの変更例）

2002年4月以降，欧州研究所は，第5次欧州研究計画のソフトウェア無線と再構成技術に関するプロジェクトSCOUT（Smart user-Centric cOmmUnication environmenT）に参画し，主に以下の項目について寄与してきた．

- 再構成可能な移動端末のユーザ操作性[3]
- ミドルウェアに焦点をあてた移動端末のシステムアーキテクチャ[4]
- 再構成可能な移動端末に対応するネットワークアーキテクチャ[5]
- アドホックネットワークにおける再構成技術[1][2]

2004年1月以降は，第6次欧州研究計画のエンド・ツー・エンド（E2E：End to End）での再構成技術に関するプロジェクトE2R（End to end Reconfigurability）に参画寄与している．

3. サービス適応技術

将来の移動通信網において予想されるサービスへの高い需要に適切に対処するには，複数の分散サービス基盤が稼動するコンピューティング環境を，包括的にサポートすることが不可欠である．課題の一部として，異種環境の統合，開発提供周期の短縮，サービス提供者が提供するサービス構成要素の統合，実行時のソフトウェア更新，サービスのパーソナル化，システム中断時間の最短化が考えられる．

本研究では上記課題の解決法として，サービス適応（DSA（Dynamic Service Adaptation））[6]技術を提案している．

DSAの目的は，複雑なサービスやアプリケーションの進化（USE（Unanticipated Software Evolution））[7]に対応することである．DSAは，サービスとプラットフォームを進化させ，個々の構成要素を異なるペースで向上させられるようにし，適応的なサービス統合，パーソナル化，コンテキスト認識，およびユビキタスコンピューティングを促進

する．将来の移動通信網においては，ネットワーク側と移動端末側の双方でDSAが重要となる．

適応技術に関しては，さまざまな技術が研究および実用化されている．それらのほとんどはコンテンツおよび通信に関する適応技術であり，サービス論理／振舞いに関する適応技術はほとんど注目されていなかった．そこで本研究では，サービス論理／振舞いの適応技術に焦点を当てている．また，この適応技術が実行される時期は，ソフトウェア開発時，コンパイル時，読込時，実行時が想定されるが，本研究では，アスペクト指向プログラミング*1（AOP：Aspect Oriented Programming）[8]，自己反映計算（computational reflection）[9]，実行時実装技術（very late binding）[10]を組み合わせ，実際に適応が要求される場合，実行時にサービスを中断せずにサービスやプラットフォームを適応させることに焦点を当てている．適応の3つの局面を図2に示す．効果的な事例として，システム中断時間の最短化を取り上げる．システムの初期導入後には問題が起こることが多いが，これまでは，それに対し実行時に対処することは出来なかった．DSAでは，実行時に要求に応じて修正措置を施すことができ，システムやサービスの中断を回避する，または，それらの中断時間を最小限に抑えることができる．

適応性の概念は，モジュール性[11]および変更可能点（variation point）[12]の概念と密接な関係がある．モジュール性は，システムの柔軟さと分かりやすさを向上させながら開発時間の短縮を可能にする．変更可能点は，システム設計において変化が起こると予測されるモジュール境界を明確に指定するが，それらの変化を指定する必要はない．変更可能点の導入により，システムの共通および可変部分の分離・合成が可能となり，システムの柔軟さが高まる．変更および変更可能点は，システム構築の基礎となるプログラミングプラットフォームが提供するモジュール性によって決まる．AOPでは，新たな細かいモジュール性の構成概念を用意することにより，複数の役割分担（これまでのオブジェクト指向技術では複数のモジュールに相当する）をまたがる横断的な関係を表すことができる．DSAでは，変化の単位を最も適切に示すことが可能なアスペクトモジュラリティ構成体を使用している．

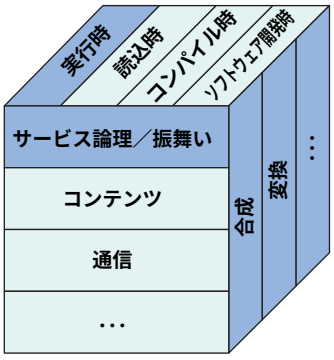

図2　適応の3つの局面

本研究では，自己反映計算と実行時実装技術をサポートするSmalltalk/Squeak[13]およびSmalltalk/Squeak向けAOPであるAspectS[14]を基として，試験環境も構築している．実行時でのバグ修正，広告，スタイルシートの変更，既存提供アプリケーションへの機能追加など，デモアプリケーションやサービスを実装した状態で提案の一部を評価した[6]．図3に，DSAを活用する適応プラットフォームのデモの様子を示す．

次の段階としては，異なる実行環境への拡張を行うとともに，前章で述べたソフトウェアの配布更新の側面に取り組む予定である．また，DSAに関しては，ソフトウェア工学原理やソフトウェア進化の仕組みも含めて取り組んでいる．AOP開発（AOSD（Aspect Oriented Software Development）[15]）やUSEにおける研究活動と連携して研究を進めている．

4. 適応ユーザインタフェース技術

将来の移動通信網において，ユーザは，さまざまな形態の通信に加え，情報アクセス，遠隔監視，機器制御など多岐にわたるサービスを，さまざまなデバイス（本章では，移動端末あるいは環境に遍在する端末）を用いて享受する．また，それらのサービスにかかわる機会も増大する．つまり，将来，ユーザとサービスの相互作用はより多様で多量なものとなる．

上記サービス環境下において必要となるユーザインタフェースに関しては，以下の課題がある．

- ユーザは，多様なデバイスを用い多様なサービスを享受する際やそれらを設定する際，さまざまなモダリティや手順を処理する必要がある．したがって，ユーザインタフェースは目的に対して十分に単純化およびパーソナル化され，ユーザのコンテキスト情報*2に合わせて効果的に調整されていること．また，ユーザの注意（attention）や感情の状態もコンテキスト情報として考慮されること．

＊1　アスペクト指向プログラミング（AOP：Aspect Oriented Programming）：複雑なシステムで本質的な横断的関心という課題に対し，明示的に横断構造（アスペクトと呼ぶ）を捕捉するために新しいモジュラリティ構成体（アスペクトモジュラリティ構成体）を導入することで対処しているプログラミング手法．このような構造は，ソフトウェアシステムの設計時および実装時に見られる．

＊2　コンテキスト情報（context information）：エンティティの状況を特徴付けるために使用可能なあらゆる情報．エンティティはユーザとアプリケーションの相互作用に関連すると見なされる人，場所，またはオブジェクトで，ユーザとアプリケーション自体も含む．

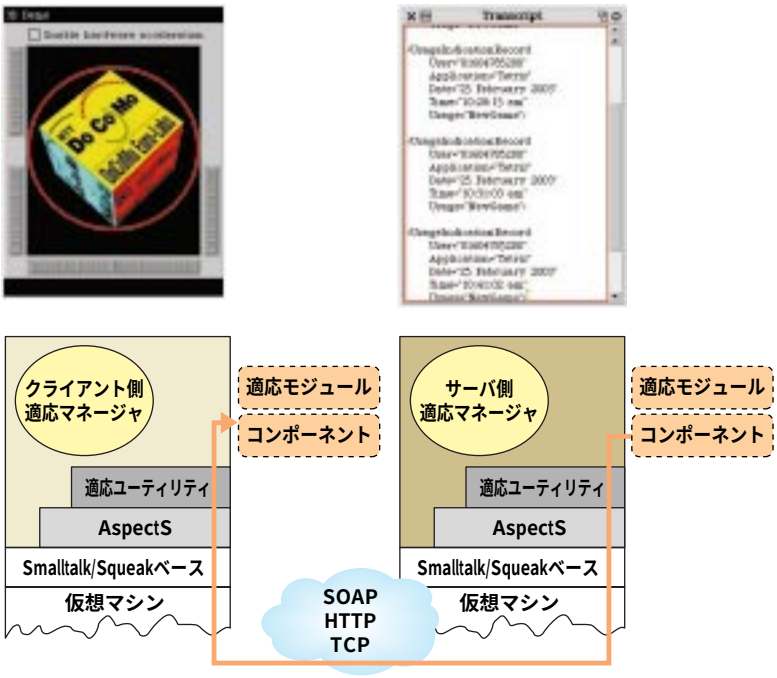

図3　適応プラットフォーム

- ユーザの多様な移動性のサポート．特に，ユーザプロファイルが各デバイスで共通に使用できること．また，環境に遍在するデバイスの利用に関しては，ユーザがその可用性や機能を認識できること．
- ユーザが多数のサービスと同時に相互作用することは難しい．したがって，ユーザは目的に対して必要な回数だけの明示的な相互作用を行えば十分であること．また，それら相互作用間の移行は滑らかであること．

本研究では，上記課題を踏まえ，同時並行する相互作用を制御するユーザインタフェース技術に焦点を当て，現在は，シナリオおよび要求条件の詳細化と主要な構成技術である通知（notification）技術の検討を進めている．

検討したシナリオ[16][17]について，以下に述べる．ユーザとその環境（社会的，物理的あるいは計算機リソースという意味で）の情報，すなわち，コンテキスト情報が，センサやデバイスなどから収集される．注意や感情に関する情報も収集される．このコンテキスト情報は，ユーザプロファイルに格納されてシステムで共有されることもあれば，いずれかのデバイスに送信されることもある．このコンテキスト情報は，デバイス間での転送が可能であり，ユーザの行動モデルを基に分析される．まず，ユーザが現在かかわっている複数の相互作用に関してユーザの現在の注意配分が推定される．次に，将来のあり得るあるいは望ましいと想定される複数の相互作用に関してユーザの注意配分が推定される．その後，それらの推定を基に，サービスやアプリケーションを含めたユーザインタフェースが効果的に調整され，何らかの通知がユーザに行われ，ユーザは現在の注意配分から将来の注意配分への滑らかな移行を行うことが可能となる．ユーザの注意を引き付けているサービスは適切な方法で拡張され，ユーザの注意を引き付けていないサービスは一時的に中断されることもある．

通知には，明示的に注意することなく認識される事物である，周辺（periphery）に関する感覚の活用も検討されている[17]．周辺感覚において，人間は気配を察しているだけであって，明示的に相互作用するには事物を中心に引き戻す必要があるものの，周辺は人間に過度の負担を強いることなく，多くの情報を提供し，全体状況の把握を可能にする．周辺感覚の活用は，ユビキタスコンピューティング環境や拡張現実感環境の実現に向けても検討されている[18][19]．ただし，通知によって緊急にユーザの現在の作業を中断させる場合もあるため，他の手法（例えば警報など）との併用が必要である．また，通知を行うためには，ユーザの注意の対象を追いかけるなど，ネットワークの機能も必要となる．さらに上記シナリオでは，通知には注意配分の移行を行うための事前交渉の意味もあり，確認手順が必要である[16]．

2004年1月以降は，第6次欧州研究計画のSIMPLICITY（Secure, Internet-able, Mobile Platforms LeadIng CItizens Towards simplicitY）プロジェクトにも参画寄与している．将来の移動通信網にふさわしいユーザインタフェースに関するシナリオや要求条件，アーキテクチャ全体の検討へ貢献することを目的としている．ここでは，ユーザインタフェースに関するもう一方の課題であるユーザの多様な移動性をサポートする単純でパーソナル化されたユーザインタフェースの検討も行う．

5. あとがき

本稿では，欧州研究所での持続的発展可能システムの研究成果および研究状況として，再構成技術，サービス適応技術，適応ユーザインタフェース技術への取組みを概説した．

文　献

[1] L. Yao and C. Prehofer: “Local Software Update for Terminal Reconfiguration using Ad Hoc Networks,” SDR ’03 Technical Conference, Nov. 2003.

[2] C. Prehofer and B. Souville: “Synchronized reconfiguration of a group of mobile nodes in ad-hoc networks,” ICT ’2003, Vol. 1, pp. 400-405, Feb. 2003.

[3] J. Hamard, G. Conaty and R. Navarro-Prieto: “A User-Centric Approach for the Development of Usable Reconfigurable Terminals,” IST Summit 2003, Vol. 2, pp. 842-846, Jun. 2003.

[4] N. Georganopoulos, T. Farnham, T. Schoeler, R. Burgess, P. Warr, Z. Golubicic, J. Sessler, F. Platbrood, B. Souville and S. Buljore: “Terminal-Centric View of Software Reconfigurable System Architecture and Enabling Components and Technologies,” IEEE Communications Magazine, Jan. 2004.

[5] C. Prehofer, L. Yao, K. Kawamura and B. Souville: “Middleware and Networking Support for Re-configurable Terminals,” SDR ’02 Technical Conference, Vol. 2, Nov. 2002.

[6] R. Hirschfeld, K. Kawamura and H. Berndt: “Dynamic Service Adaptation for Runtime System Extensions.” In: WONS ’04 Proceedings, 2004, to appear.

[7] Unanticipated Software Evolution homepage (http://www.joint.org/use/)

[8] G. Kiczales, J. Lamping, A. Mendhekar, Ch. Maeda, C. V. Lopes, J.-M. Loingtier and J. Irwin: “Aspect-Oriented Programming.” In: ECOOP ’97 Proceedings, 1997, pp. 220-242.

[9] P. Maes: “Concepts and Experiments in Computational Reflection.” In: OOPSLA ’87 Proceedings, pp. 147-155, 1987.

[10] A. Kay: “Is “Software Engineering” an Oxymoron?” Viewpoints Research Institute, 2002.

[11] D. L. Parnas: “On the Criteria To Be Used in Decomposing Systems into Modules.” In: Communications of the ACM, Vol. 15, No. 12, pp. 1053-1058, Dec. 1972.

[12] K. Czarnecki: “Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models.” Dissertation, TU Ilmenau, 1998.

[13] D. Ingalls, T. Kaehler, J. Maloney and S. Wallace, A. Kay: “Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself.” In: OOPSLA ’97 Proceedings, pp. 318-326, 1997.

[14] R. Hirschfeld: “AspectS-Aspect-Oriented Programming with Squeak.” In: M. Aksit, M. Mezini, R. Unland, editors, Objects, Components, Architectures, Services, and Applications for a Networked World, LNCS 2591, pp. 216-232, Springer, 2003.

[15] Aspect-Oriented Software Development homepage (http://www.aosd.net/)

[16] J. S. Shell, T. Selker and R. Vertegaal: “Interacting with Groups of Computers.” In: Communications of the ACM, Vol. 46, No. 3, pp. 40-46, Mar. 2003.

[17] D. S. McCrickard and C. M. Chewar: “User Goals and Attention Costs.” In: Communications of the ACM, Vol. 46, No. 3, pp. 67-72, Mar. 2003.

[18] M. Weiser and J. S. Brown: “The coming age of calm technology. revised version of, Designing calm technology.” Power Grid Journal, Vol. 1.01, Jul. 1996.

[19] G. D. Abowd, E. D. Mynatt and T. Rodden: “The Human Experience.” In: IEEE Pervasive Computing, Vol. 1, No. 1, pp. 48-57, Jan.-Mar. 2002.

用 語 一 覧

ACK：ACKnowledgement
AOP：Aspect Oriented Programming（アスペクト指向プログラミング）
AOSD：Aspect Oriented Software Development
DSA：Dynamic Service Adaptation
E2E：End to End（エンド・ツー・エンド）
E2R：End to end Reconfigurability
HTTP：Hyper Text Transfer Protocol
SCOUT：Smart user-Centric cOmmUnication environmenT
SIMPLICITY：Secure, Internet-able, Mobile Platforms LeadIng CItizens Towards simplicitY
SOAP：Simple Object Access Protocol
TCP：Transmission Control Protocol
USE：Unanticipated Software Evolution