・ネットワークソフトウェア・アーキテクチャ

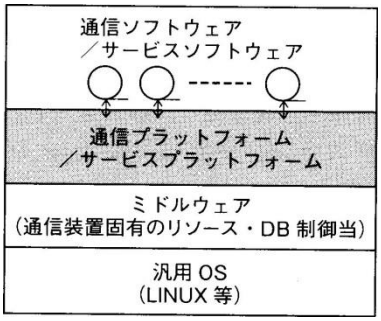

図 3．1 ネットワークソフトウェアーアーキテクチャ

このネットワークソフトウェア，プラットフォーム，ミドルウェア，OS をすべて総合してネットワークソフトウェア・アーキテクチャと呼ぶ

ミドルウェア 【 middleware 】 M/W / MW / ミドルソフト / middle software
OS 上で動作し、アプリケーションソフトに対して OS よりも高度で具体的な機能を提供するソフトウェア。OS とアプリケーションソフトの中間的な性格を持っている。

多くのアプリケーションソフトで共通して利用される機能は、個別に開発するのは非効率であるため、通常は OS の機能として提供され、アプリケーションソフトは OS の機能を利用するだけですむようになっている。

だが、このようにして OS に実装される機能はどんなアプリケーションソフトでも必ず必要とされるような極めて基本的なものに限られる。

このため、特定の分野でしか使われないが、その分野では必ず必要とされるような具体的で基本的な機能は、ミドルウェアの形で提供されることが多い。

また、**ミドルウェアには OS やハードウェアによる違いを吸収し、様々なプラットフォームで動作するアプリケーションソフトの開発を容易にするというメリットがある。**

代表的なミドルウェアにはデータベース管理システム(DBMS)や、トランザクション処理機能を提供する TP モニタ、分散オブジェクト環境を提供する ORB などがある。

・通信プラットフォーム，サービスプラットフォーム

3．1．2　プラットフォームと API

上記で述べたプラットフォームは，ネットワークシステム基盤（ネットワークインフラ）を構成する各**通信ノード**上に構築されるものと，主にサービスを提供するため各種サーバや端末，ホーム GW などに構築されるプラットフォームとは，持つべき機能や性質が異なる．前者を本書では，**通信プラットフォーム**と呼び，後者を**サービスプラットフォーム**と呼ぶ．

通信プラットフォームは，各種通信プロトコルに対応した機能を提供するライブラリや物理的なリソースを隠蔽し，論理的なリソース管理機能などの基本的な呼制御を行うために必要とされる機能要素がライブラリとして通信ソフトウェアから呼び出す形式で使用できるような構成になっている・これは，通信ソフトウェアから見ると，API（Application Programming Interface）が提供されているものとなる．

一方，サービスプラットフォームは，各種通信ノードで構成されるネットワークインフラを制御してサービスに必要な呼制御機能を実現する通信制御機能や，認証，課金，プレゼンスなどのサービスを実現するために共通に必要となる機能を API として，サービスソフトウェアが利用可能なインタフェースを提供する．

いずれのプラットフォームも通信ソフトウェアやサービスソフトウェアに共通機能としての API を提供するものであり，提供する機能が異なる点と API をユーザに公開するかどうかの点が異なるだけであり，本質的な構成は変わらない．

ノード 【 node 】

ネットワークを構成する一つ一つの要素のこと。通信ネットワークではコンピュータやハブ、ルータなど一台一台の通信機器がノードに当たる。ノードとノードを結ぶ線はリンクという。

API 【 Application Programming Interface 】
プログラミングインターフェース / programming interface
あるコンピュータプログラム(ソフトウェア)の機能や管理するデータなどを、**外部の他のプログラムから呼び出して利用するための手順やデータ形式などを定めた規約**のこと。

個々のソフトウェアの開発者が毎回すべての機能をゼロから開発するのは困難で無駄が多いため、多くのソフトウェアが共通して利用する機能は、OS やミドルウェアなどの形でまとめて提供されている。そのような汎用的な機能を呼び出して利用するための手続きを定めたものが API で、個々の開発者は API に従って機能を呼び出す短いプログラムを記述するだけで、**自分でプログラミングすることなくその機能を利用したソフトウェアを作成することができる**。

近年ではネットワークを通じて外部から呼び出すことができる API を定めたソフトウェアも増えており、遠隔地にあるコンピュータの提供する機能やデータを取り込んで利用するソフトウェアを開発することができる。

・SDP（Service Delivery Platform）

3.3.1 **SDP**： Service Delivery Platform

サーバ系のサービスプラットフォームとして，**今最も着目されているもの**として，SDP（Service Delivery Platform）がある．これは，通信の標準化団体である ITU－T，3GPP，OMA などで検討されており，従来のテレコムサービスだけでなく，Web 系のサービスと連携させたサービスを容易に実現する仕組みを提供するものである．このイメージ図を図 3．2 に示す．

SDP では，通信ノード群で構成される，各種基盤となるネットワークを制御するための網機能をコンポーネント化し，これをサービス制御ソフトウェアに提供するものである．提供機能としては，図 3・2 にあるように，認証機能，課金機能，プレゼンス機能などがある．

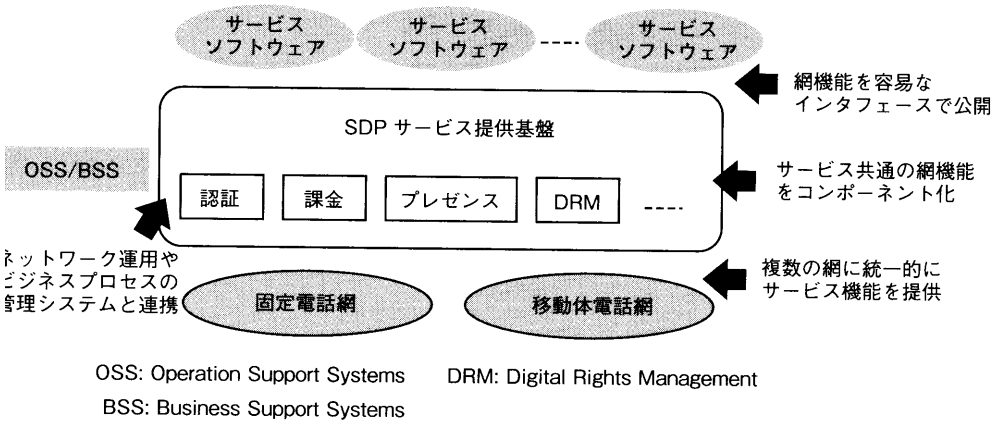

図 3.2 SDP (Service Delivery Platform) のイメージ

発呼
はっこ【発呼】
電話・LAN などの通信回線に接続すること。 「 －者」

・Android プラットフォーム

3.4.2 Android プラットフォーム

Android プラットフォームには、図 3.13 に示すように、Linux OS 上に Java の仮想マシンである DalvikVM とそのライブラリ群が配備されている．さらにこの上に SDP のサービス提供基盤に相当する「アプリケーションフレームワーク」が置かれている．

アプリケーションフレームワークには，起動や終了といったアプリケーションのライフサイクルを管理する「アクティビティ管理」や「電話管理」，「位置情報管理」といったものが，Java API で提供される．

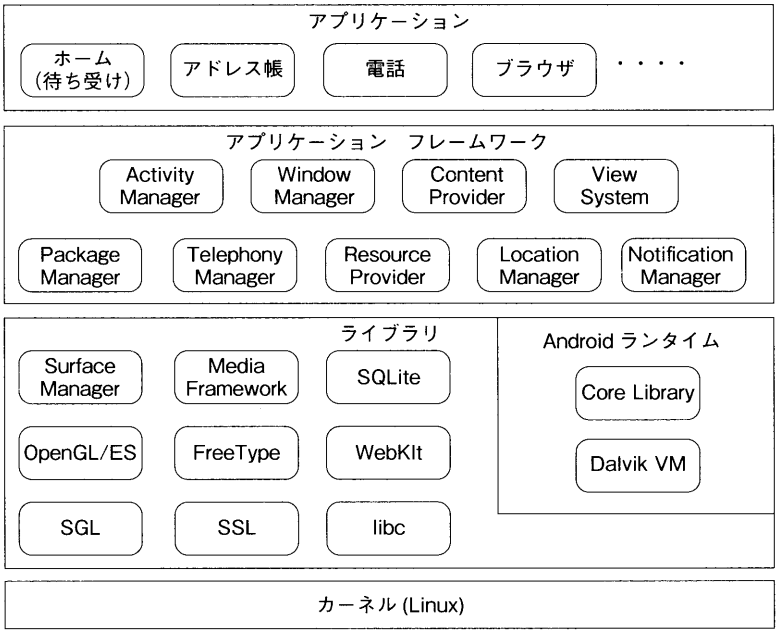

図 3.13 Android プラットフォーム

フィーチャーフォンと Android スマホ

さらに，一品物であったフィーチャーフォンと異なり，Android の場合，標準品の OS と，標準的なチップセットを組み合わせることにより，簡単に製造できるようになった．海外メーカなどでは，**ほとんど，テストをしないで新製品を出荷する**例もある．

この他に，OS だけでなく開発環境もオープンになったことにより，専門家でない人，**中学生や高校生も含め，ネットワークソフトウェア市場に参入できた**ことも，大きなブレークスルーであるといえる．

このような流れを一言でいうと，ネットワークソフトウェアは，オープンソースソフトウェア（OSS：Open Source Software）の組合せで構成されるようになってきたといえる．

表 4.1 フィーチャーフォンとスマートフォンの違い

	伝統的な携帯電話端末（フィーチャーフォン）	スマートフォン
OS	リアルタイム性を意識した専用 OS	Linux などの汎用 OS
UI	シンプルな UI	リッチな UI
言語	C，BREW（Qualcomm 社が提供する組み込み用の言語，au が利用），Java	Java，JavaScript
生産性	低い	高い

OSS / Open Source Software

ソフトウェアの設計図にあたるソースコードを、インターネットなどを通じて無償で公開し、誰でもそのソフトウェアの改良、再配布が行えるようにすること。また、そのようなソフトウェア。

ソースコードがあれば、そのソフトウェアの類似品を作成したり、そのソフトウェアで利用されている技術を転用することが容易に可能なため、企業などでは自社の開発したソフトウェアのソースコードは極秘とし、他社に供与するときにはライセンス料を取ることが多い。

1998 年、The Open Source Initiative(OSI)という団体によって「The Open Source Definition」(OSD)という定義が発表されている。「自由な再頒布の許可」「派生ソフトウェアの頒布の許可」「個人や集団の差別の禁止」「適用分野の制限の禁止」など 10 項目からなり、これに準拠しているソフトウェアライセンスには「OSI 認定マーク」が付与される。ただし、人々が日常使う「オープンソース」という言葉が必ずしも OSD の内容を指しているとは限らない。

・世界で最も使われている MySQL

4．6．1 MySQL

MySQL【13，14】は，Oracle 社が開発する RDBMS（リレーショナルデータベースを管理，運用するためのシステム）の実装の 1 つである．オープンソースで開発されており，GNU GPL と商用ライセンスのデュアルライセンスとなっている．MySQL は，当初はスウェーデンの MySQL AB 社のものであったが，2008 年に MySQL AB 社が Sun Microsystems 社に買収されたことによって Sun Microsystems の所有となった．2010 年，Sun Microsystems 社は Oracle に買収されたため Oracle 社の所有である（2012 年現在）．MySQL は高速性と堅牢性

を追及したマルチユーザ・マルチスレッドの SQL データベースであり，**世界で最も使われている**．その主要な特徴は以下の通りである．
・ANSI SQL92 準拠
・様々なプラットホームへの対応
・トランザクション機能のサポート

・グローバル化されたネットワークソフトウェア

オープンソースソフトウェアは，グローバルに管理されている．このことは，ネットワークソフトウェアの世界そのもののグローバル化であり，それに対応ができるスキルが，技術者に求められる時代になってきたといえる．

・コンテキストアウェアネス

7.4.1 コンテキストアウェアネスとプレゼンス

コンテキストアウェアネス（Context Awareness）とは、対象が置かれている状況（コンテキスト）を認識し，それに応じた情報提供やサービスの推薦を行うものである．実例としては，書籍ネット通販の **Amazon** が，全ユーザの購入履歴の情報を基に，書籍を購入したユーザに次のお勧め書籍を推薦しているものが，**この技術の最初の適用例**といえる．この場合のコンテキストは，ユーザの購入履歴であるが，これ以外にも多くのコンテキストが検討されており，それをその取得方法と併せて以下に示す．

（1）コンテキストの種類
（a）ユーザの静的なコンテキスト
・ユーザのプロフィール
年齢，性別，住所，職業，趣味　など
（取得方法）ユーザアカウント情報より取得
（b）ユーザの動的なコンテキスト
・購入履歴
（取得方法）ネット通販購入履歴
・スケジュール，To Do リストなど
（取得方法）スケジュールアプリケーション
・動作状況（歩行中，乗車中，運転中，着席中，PC 作業中，通信中など）
（取得方法）装着センサなどからの情報
・位置情報
（取得方法）GPS，RFID，加速度センサなどからの情報
・通信方法（電話，メール送受信，Web アクセス，動画視聴，など）
（取得方法）通信端末上のアプリケーションなどの動作情報
・健康状況（心拍，脈波，脳波，血圧　など）
（取得方法）装着，設置各種センサからの情報
（c）ユーザの外的なコンテキスト
・環境情報（温度，湿度，天気，照度　など）
（取得方法）各種設置センサからの情報
・通信環境（端末能力（動作性能，画面表示性能など），アクセス回線速度）
（取得方法）通信端末，ネットワーク装置からの情報

上記のコンテキストのうち，特にユーザの現在の状態を示す，「動作状況」のコンテキストは，「プレゼンス」とも呼ばれる．プレゼンスは，Instant Messenger（IM）に利用されており，**PC の動作状態より，ユーザの在席中，離席中などを自動的に判定**している．他のユーザ状態については，ユーザが手動で入力することになるが，現在，加速度センサやジャイロセンサ，通信端末からの情報などより，自動的に判定する方法が研究・開発されている．

Amazon「その他のおすすめ:　最近の履歴に含まれている商品を買った人は、こんな商品も買っています」

・V字モデル

開発工程における品質管理

本項では，V字モデルに従って，各開発工程で実施すべき項目と，ソフトウェアの品質管理の観点から実施する施策について述べる．V字モデルは**ウォーターフォールモデル**の開発工程を図式化したもので，各開発工程の生産物の詳細度を表現している．図 8．1にV字モデルの概念を示す．図の左側は仕様を詳細に設計していく過程であり，図の右側はテストを詳細レベルからソフトウェア全体に，さらにシステムレベルへと結合していく過程である．図の左側の工程で決定した内容と同じレベルの右側の工程で確認する内容は一致する．たとえば，要件定義で決定した内容は出荷テストで確認されるべき内容である．

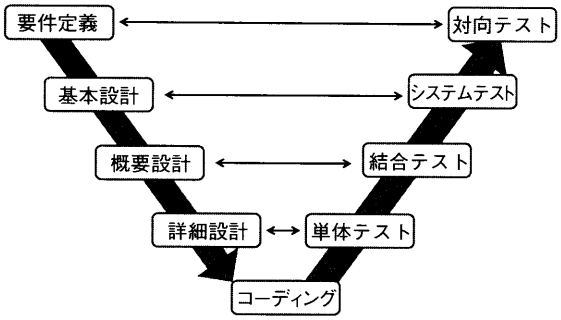

図 8.1　V 字モデル

・メトリクス

ソフトウェアをある視点で評価し数値化した指標をメトリクスと呼ぶ．表 8．1 に各工程の主な生産物とプロセス品質メトリクス，プロダクト品質メトリクスを示す．それぞれのメトリクスの具体的な意味については 8．3 節で述べる．

表 8.1　各工程の生産物と品質メトリクス

工程	主な生産物	主なプロセス品質メトリクス	主なプロダクト品質メトリクス
基本設計	ソフトウェア全体の構成図	レビュー工数，検出欠陥数	CK メトリクス (オブジェクト設計の場合) 結合度，凝集度 (非オブジェクト設計の場合)，DSM
概要設計	クラス構成図	レビュー工数，検出欠陥数	
詳細設計	フローチャート	レビュー工数，検出欠陥数	
コーディング	ソースコード	レビュー工数，検出欠陥数	規模，静的解析ツールの指摘件数，コードクローンメトリクス
単体テスト	単体テスト項目リスト，手順書，結果報告書	テスト項目数，レビュー工数，検出欠陥数	テストカバレッジ
結合テスト	結合テスト項目リスト，手順書，結果報告書	テスト項目数，レビュー工数，検出欠陥数	
システムテスト	システムテスト項目リスト，手順書，結果報告書	テスト項目数，レビュー工数，検出欠陥数	

ホワイトボックステスト　【 white box test 】

システム内部の構造を理解した上でそれら一つ一つが意図した通りに動作しているかを確認する、プログラムのテスト方法。

「命令網羅」「判定条件網羅」「条件網羅」「複数条件網羅」「経路組み合わせ網羅」などの方式があるが、基本的にはプログラム内の全ての命令、全てのルーチンが最低一回は実行され、検証されるようになっている。

つまり、プログラムの全ての部分が、プログラム記述者の意図通りに動作していることを確認するテストである。システムの機能よりも内部構造の整合性を重視したテストとも言える。

網羅的なテストであるため珍しい事態に対しても動作確認できるのが利点だが、あくまでプログラム記述者の意図との整合性を確認するだけなので、記述者自身に誤解があった場合は対処できないという欠点も持つ。

これに対して、プログラムの内部構造とは関係なく、外部から見て仕様書通りの機能を持っているか確認するテストを「ブラックボックステスト」という。