NetFront™ ソフトウェアアーキテクチャといまどきのソフトウェア開発

プログラミング言語処理系論 ***2010.05.19***

山上　俊彦

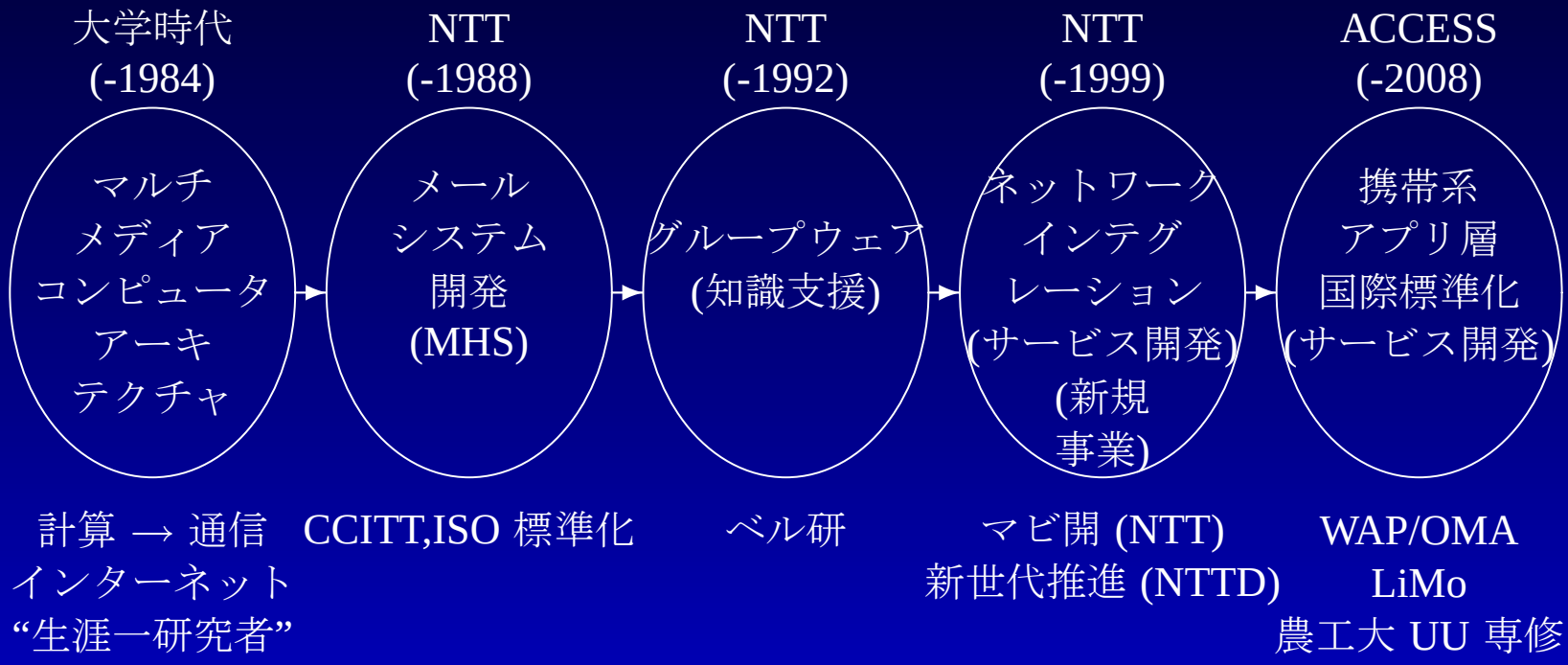
Toshihiko.Yamakami@access-company.com

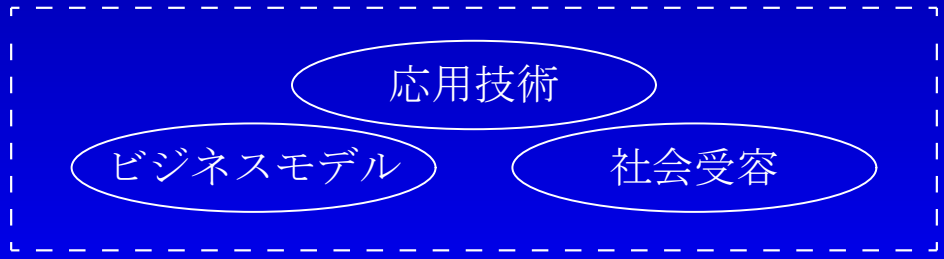
(株)ACCESS CTO Office

全体

- はじめに: ACCESS, 山上俊彦
- NetFront のアーキテクチャ
- ソフトウェア開発、そして、知のパラダイム変更

経歴





ACCESS overview

- Founded in 1984 (Tokyo)
- 1,484 employees (Group 2010/2)

NetFront™ Family

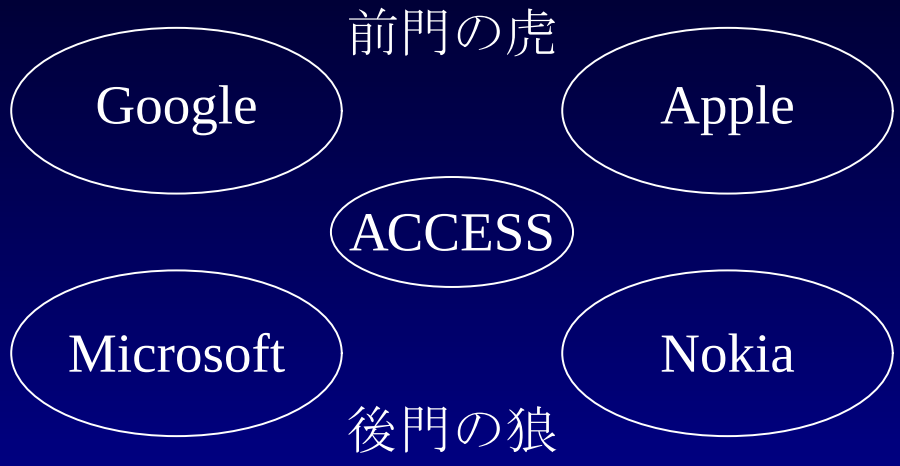
- ブラウザおよびネットワークソフトコンポーネント:
  - 8.07 億ライセンス (2009 年 7 月現在)
  - 年間 1.5–2.0 億ライセンス

- 東大理学部情報科学科誕生から 35 年経過

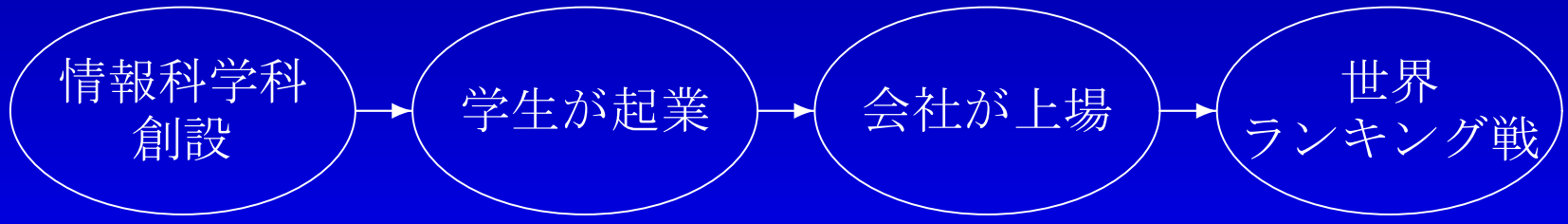
熱き思い

- 現役にこだわりたい。まだ熱い思いがある。(中山雅史)
- 燃えていないろうそくは次のろうそくを燃やせない (タゴール)
- 研究は Rock-n-roll。講義はライブ
- コンピュータがすごいということを伝えたい (e.g. 旭山動物園小菅園長)
- 贈る言葉
  - 若さはそれだけでひとつの才能 (本田宗一郎)
  - 金儲けより時間儲け: 人に聞く能力、探す時間を減らす
  - コミュニケーション能力の向上は他人と自分を幸福に
  - 仕事に感性や創造性を使えなければ人生は味気ない
  - 健康でいてさえくれればいい (もし余力があれば弛まず努力) ／他人の痛みがわかる人に
  - 命は大事、しかし命を賭けられる仕事があるのも幸せ
  - 自分を教育できるのは自分だけ

世界への長い道



- 事を構える気がなくて、先方が攻め入ってくる



[参考] 世界の携帯電話ユーザ数

国名	ユーザ数 (百万)	時点	2007 末ユーザ数 (百万)
China	703	2009.07	530
India	506	2009.11	237
USA	276	2009.09	252
Russia	204	2009.09	172
Brazil	158	2009.05	122
Indonesia	145	2009.03	92
Japan	111	2009.12	100
Germany	107	2008.12	97
Pakistan	96	2009.07	76
Italy	89	2009.06	90

http://en.wikipedia.org/wiki/List_of_
mobile_network_operators_of_Europe et al

[参考] 世界携帯電話出荷数 2005-2008

ベンダ	2008		2007		2006		2005	
	台数 (百万台)	シェア (%)	台数 (百万台)	シェア (%)	台数 (百万台)	シェア (%)	台数 (百万台)	シェア (%)
Nokia	468.4	39.7	437.1	38.2	347.5	34.1	264.9	31.8
Samsung	196.7	16.7	161.1	14.1	118.0	11.6	102.8	12.3
LG Electronics	100.7	8.5	80.5	7.0	64.4	6.3	54.9	6.6
Motorola	100.1	8.5	159.0	13.9	217.4	21.3	146.0	17.5
Sony Ericsson	96.6	8.2	103.4	9.0	74.8	7.3	51.1	6.1
その他	218.5	18.5	202.9	17.7	197.8	19.4	213.1	25.6
合計	1180.9	100.0	1,144.1	100.0	1,019.9	100.0	832.8	100.0

Note: IDC Worldwide Quarterly Mobile Phone Tracker Excluding OEM sales

[参考] 3G及びパケット定額

月	携帯電話加入者(百万)	3Gユーザ(百万)	3G比率(%)	パケット定額(百万)	定額比率(%)
2004.9	84.31	24.94	29.6	2.38	2.8
2005.9	89.13	39.23	44.0	9.27	10.4
2006.9	93.81	58.15	62.0	17.90	19.1
2007.9	99.33	79.83	80.4	31.03	31.2
2008.9	104.83	94.03	89.7	39.73	37.9

電気通信事業者協会 および MCF

ケータイの技術進歩

	1999	2008
パケット速度	9.6Kbps	14Mbps (HSPDA 3.5G)
CPU	数 MHz	500MHz (Dual CPU)
メモリ	1MB	256MB
推奨 Web サイズ	2KB	100KB
Java サイズ	なし	1MB
外部記憶	なし	(microSD) 16GB
画面解像度	94x72 (白黒)	480x864 (カラー 18bit)
ユーザ	数百万人	8700 万人

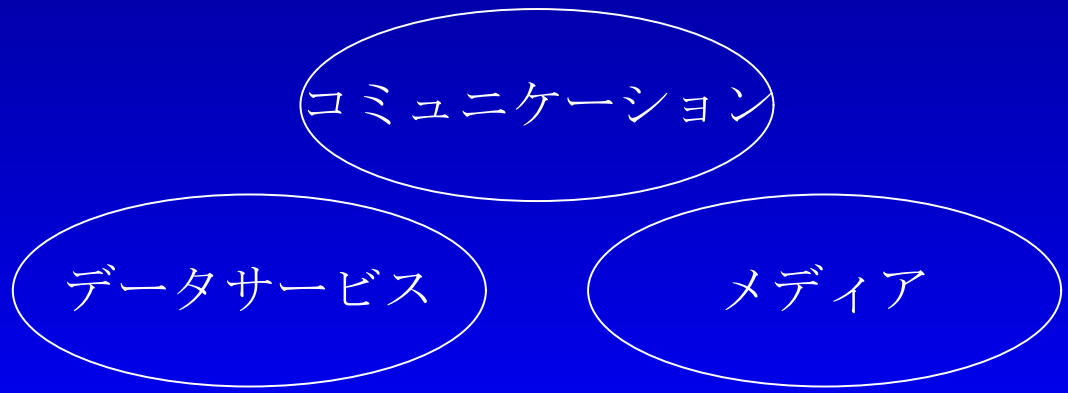
- 1000 倍になればエンジニアリングは激変

アーキテクチャ変化

- 重視する項目の変化
  - 性能 → コードサイズ → 拡張性 → 大規模開発
- 環境変化
  - 特定環境 → 汎用環境 →OS や Window システムの発達
- 移植
  - 移植コストの削減 → メンテコストの削減
  - 開発環境差異の意識 (移植方法, 開発言語)
  - 移植用組み込みソフトと専用組み込みソフトは全く違う

モバイルインターネットの歴史(1)

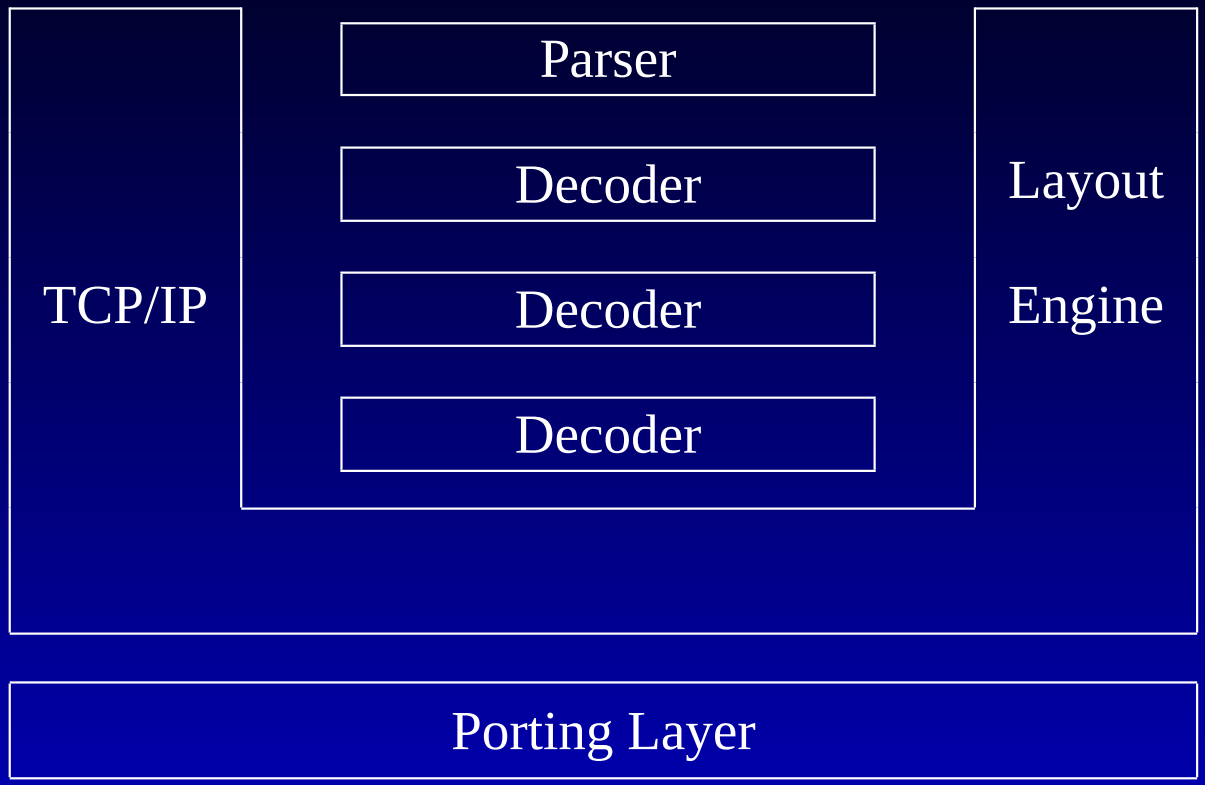
- 小さく作る
  - 初期は 300KB のプログラム、150KB の作業領域
- PC に収束する
  - Web が始まると PC インターネットのケータイ化は進む
- ユーザ経験を作る
  - ナビゲーション
  - SNS: ミクシィ、モバゲータウン
  - ショッピング、オークション
  - お財布ケータイ



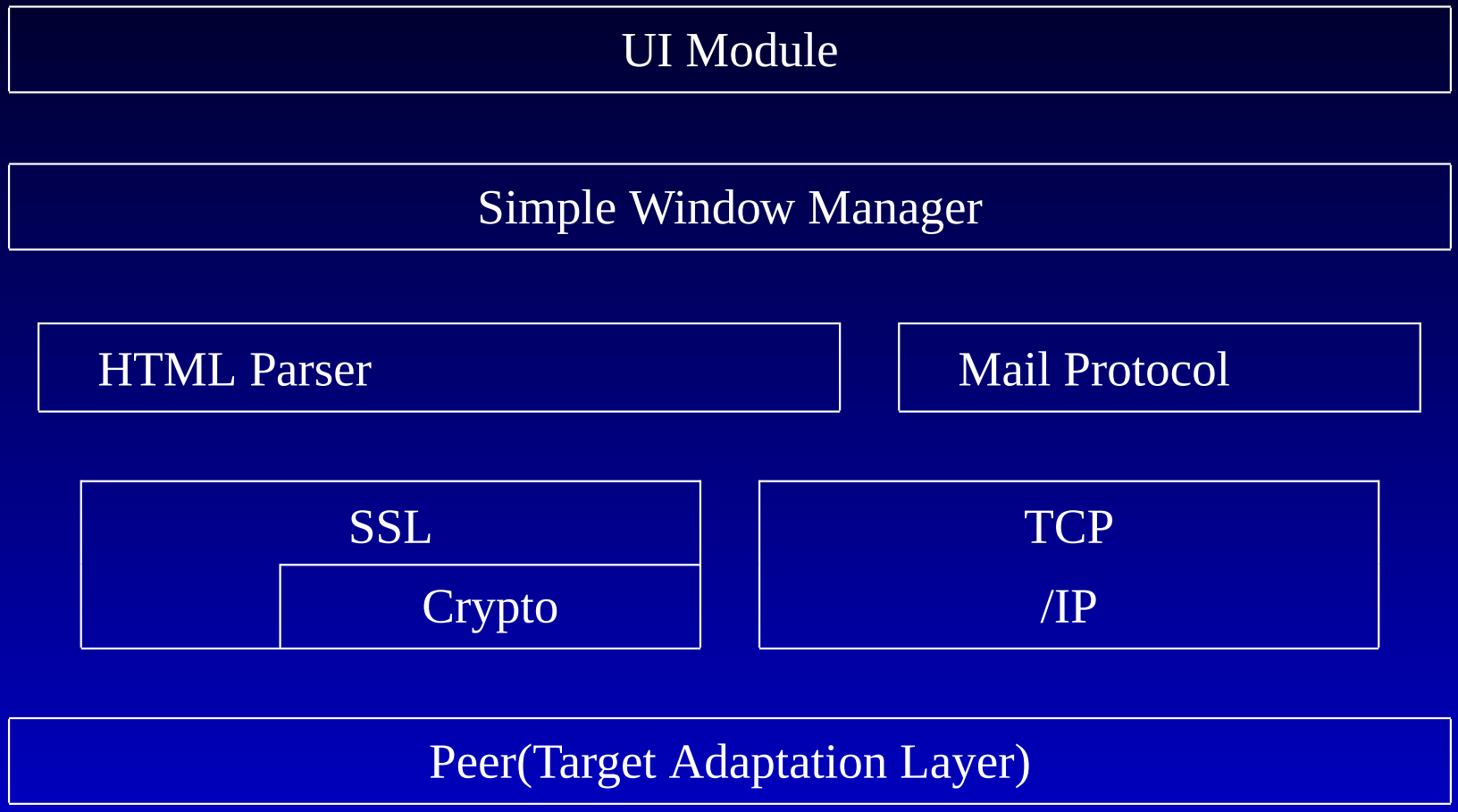
モバイルインターネットの歴史(2)

- Unwired Planet(UP) 社創立, HDML の開発 (1994)
- WAP Forum (Wireless Application Forum) 結成 (1997) Ericsson, Motorola, Nokia, UP
- i-mode 開発開始 (1997) サービス (1999)
- カラー、写メール (日本 2000)
- WAP Forum が XHTML Mobile Profile 採用 (2001)
- Java 搭載 (2001 503i)
- 3G 商用サービス開始 (2001 FOMA)
- パケット定額制 (2003 PHS, 2004 FOMA, . . .
- フルブラウザ登場 (2004 AH-K3001V, 2005 N901iS)

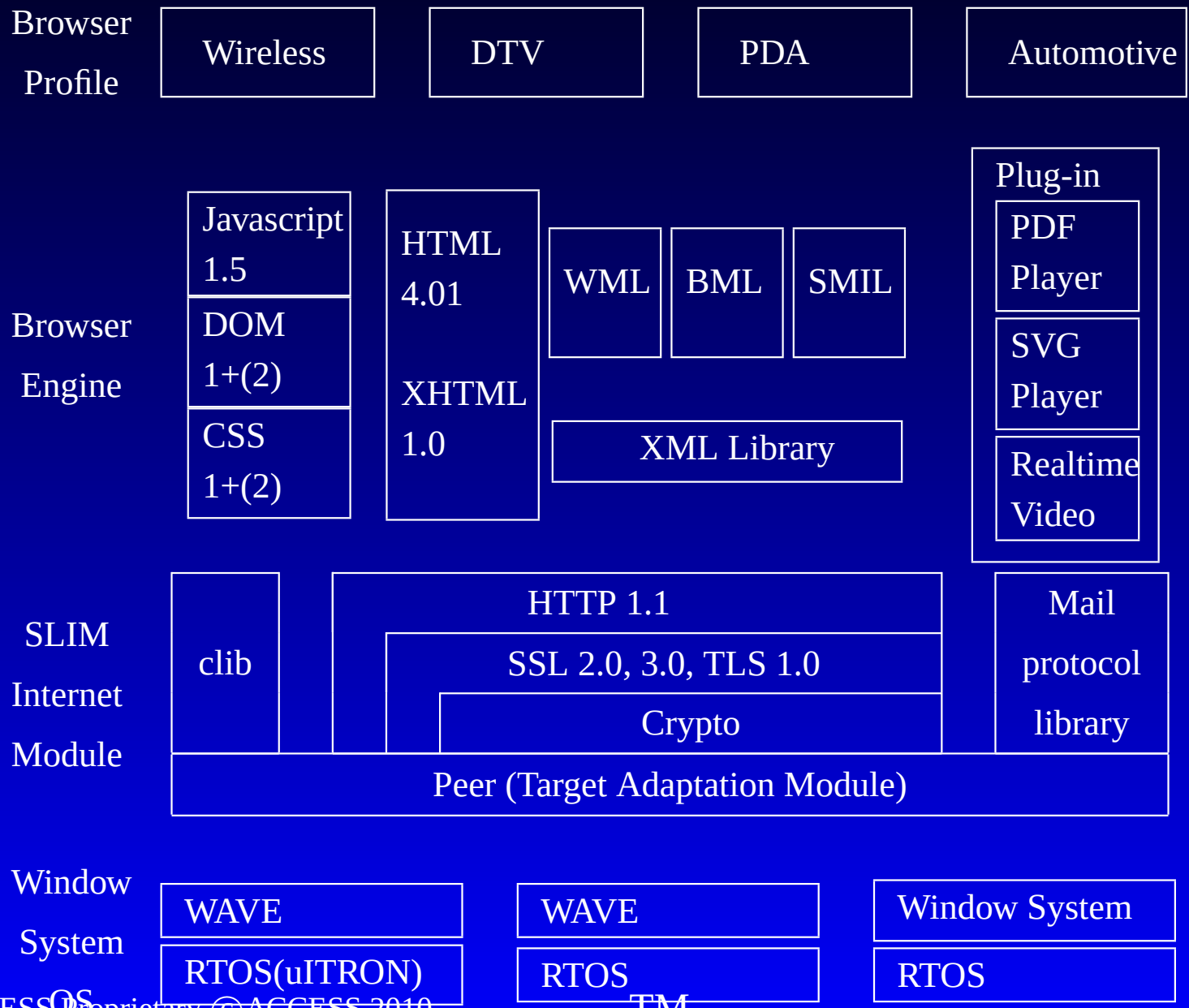
NetFront™ 1.0



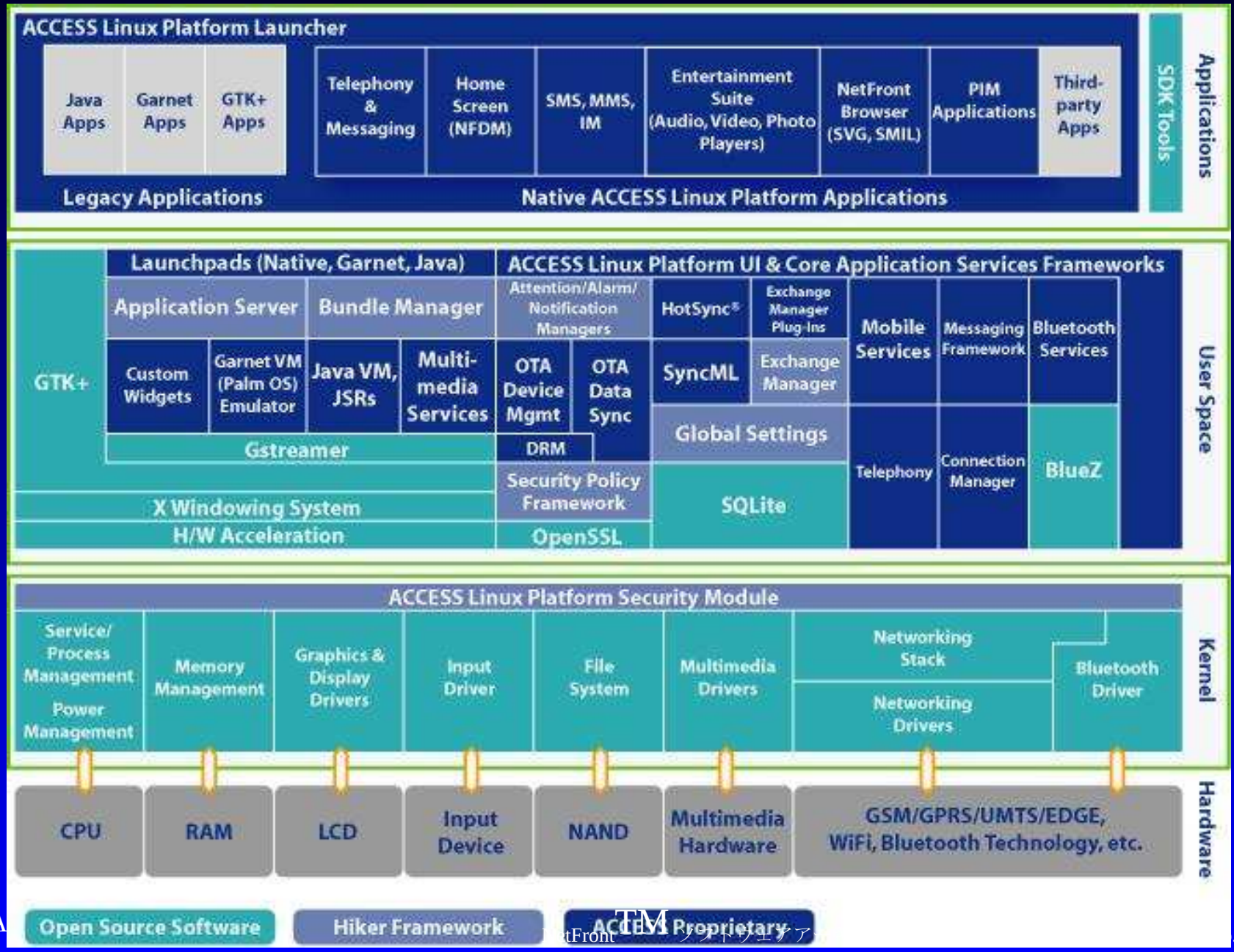
NetFront™ 2.0



NetFront™3.0 アーキテクチャ



Linuxプラットフォーム



ケータイネット実装技術の歴史

- サイズを小さくする魔法はない: 最初から小さく作る
  - 小さな工夫の集大成
  - 10 年間のノウハウの蓄積
    - 移植層
    - 前提条件 (マルチプロセス、優先転送、C)
    - 共通層 (ウィンドウ、暗号)
    - クライアント側拡張機能
    - ファイルシステムすら前提としない
    - Clib の書き換え
  - 品質 (リグレッション試験、自動試験)
- サービスプロデュース (キャリア、ベンダ、サービス業者): 市場知
  - 携帯電話には電力、干渉、容量、重量などの総合知識が必要 (Nokia)
  - ソフトにも同じような多様な制約に対する総合知が必要
- 大型システム対応 (携帯ソフトの肥大化)
- サービスエコシステム (AppStore, Android Market, . . .)

ケータイの黒船

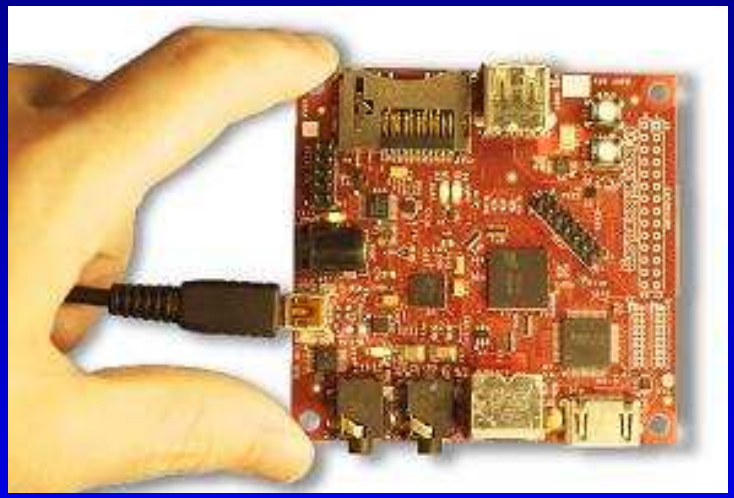
プレイヤ	Apple (iPhone)	Google (Android)
目的	iPod の携帯化	シームレスなモバイルアプリ
黒船	大規模サードパーティーアプリ	フリーなミドルウェア
新機軸	ソフトではなくエコシステムを作る	サービス側から機種依存性を打破
掟やぶり	ソフトはユーザが選択、購入	サービス事業者主導のプラットフォーム作り
アプリ製作	マーケットとしては No.1 洗練されたつくり	WebAPI の威力 (300 行でカメラ羅針盤アプリ)
ブランド	ライフスタイル提案	インターネットブランド

組み込みの伝統的作り方

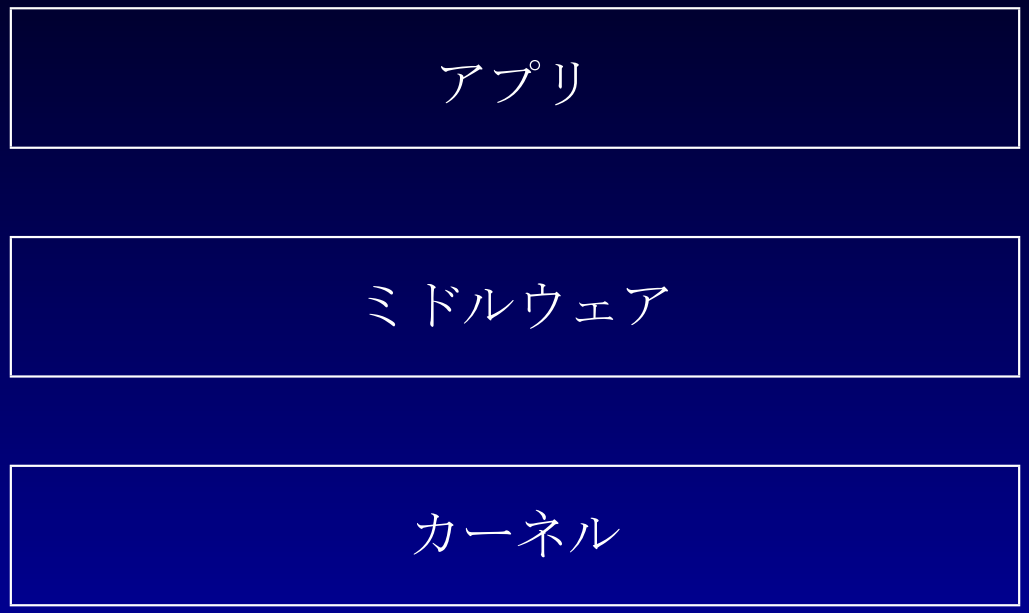
- 普通はクロス開発環境で作る
- 昔は OS なしで、ハードウェアの上でソフトウェアを起動
  - 電源を入れるとある番地から実行する
  - ハードウェアのデバイスドライバを書く
  - OS がないとプロセス管理もメモリ管理もファイルもない
- OS のかわりも自分で書く
  - ハードウェア割り込みとタイマー割り込み
  - 並列処理や割り込み前の状況の格納
  - ファイルがないと Cookie とかも大変

携帯 Linux の開発環境例

- TI(beagleboard.org) が販売している開発ボード (149US$)
- 約 8cm 四方という手のひらサイズの基板上
- CPU は ARM Cortex-A8 な CPU TI OMAP3530 を採用。メモリは 128 or 256MB
- 消費電力最大 2.5W (PC だと 30W 程度)
- 採用 CPU は C64x DSP と 2D/3D グラフィックアクセラレータの PowerVR を内蔵
- ディジタル映像出力（HDMI）や SD カード・スロット，USB2.0 ポート，RS-232C ポート，オーディオ入出力などの豊富な外部インタフェース
- ボードに USB キーボード、USB マウス、USB Ethernet をぶらさげて DVI-D なモニタを接続するだけで、X-window 完備のフルスペックな自作デスクトップ Linux を実現 (ただし ARM バイナリ)

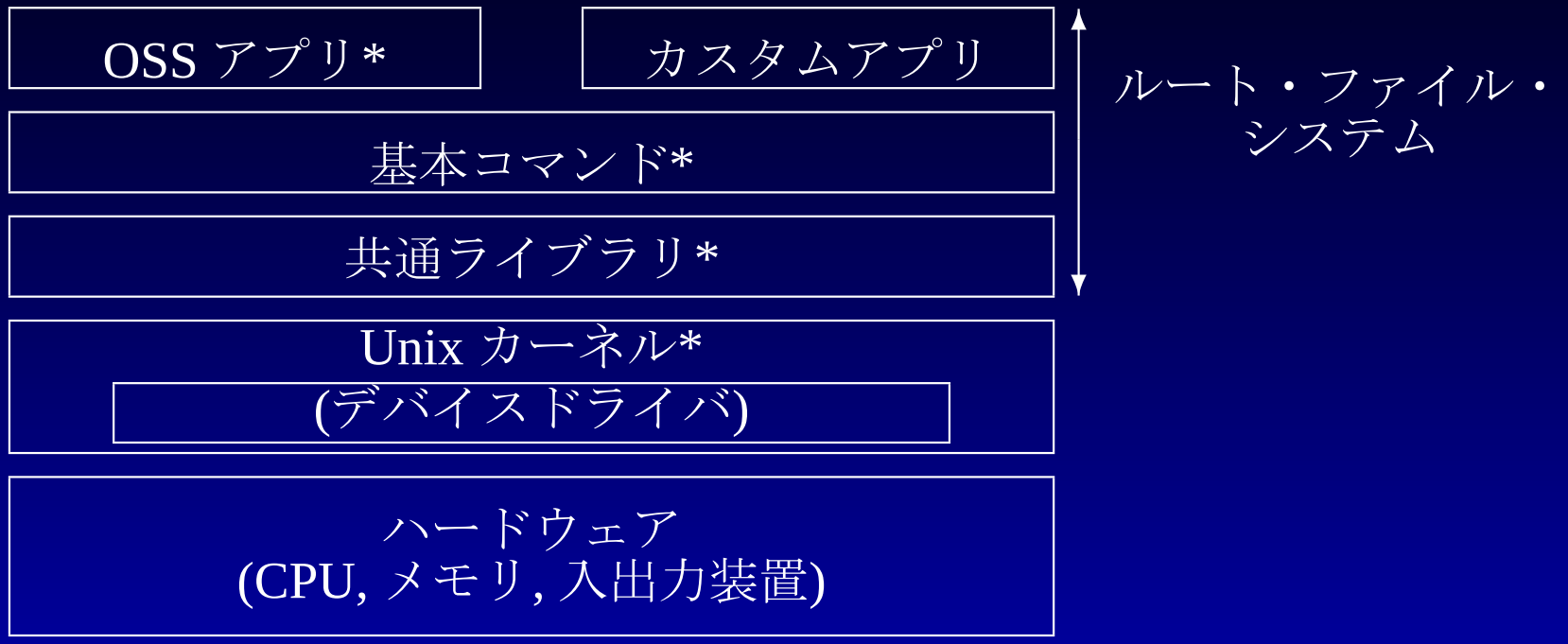


爆発するコードサイズ



- いずれの階層も 500–1000 万行サイズ
- 100 万行越えるとどんなソフト開発も困難

組み込みLinuxの例(1)



Note: * OSS を中心

組み込みLinuxの例(2)

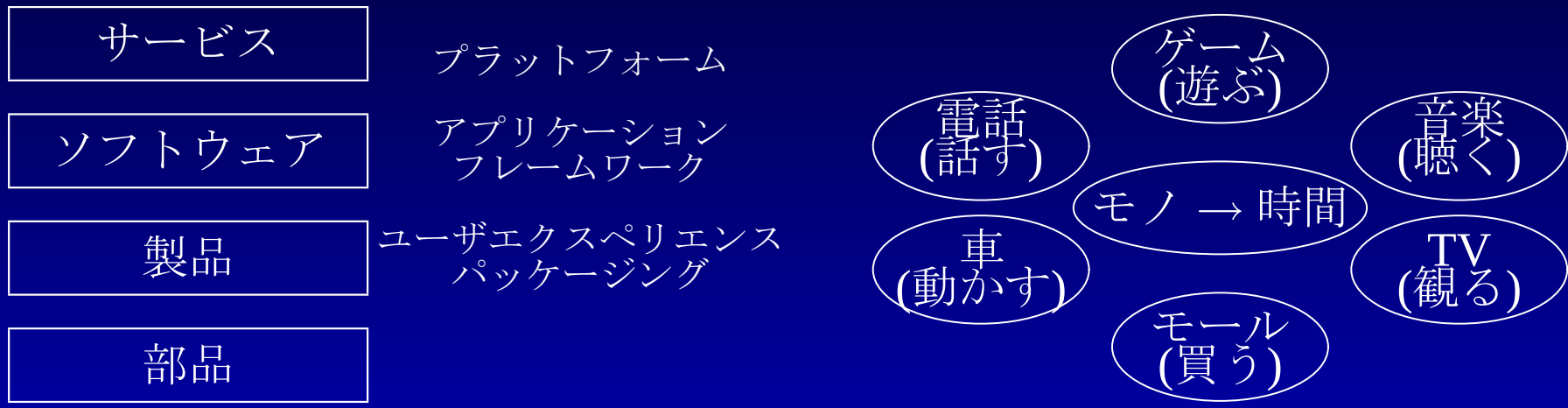
- スタートアップスクリプトでKernelをロード
- UNIXのrc.sysinit and rc.localをいじってサービスが動くようにする
- ブートしてファイルシステムイメージが見えるようにする
- ファイルシステムに組み込み用のプログラム(ROMライターなど)を書く
- ROMライターを利用してファイルシステムイメージのROMへの焼付けなどを行う

情報家電ネットワーク化の先

- インターネット融合はほぼ完成
  - OMA Open Mobile Alliance のブラウザグループは 2008 年解散
- 映像
  - YouTube 対応、BeeTV など携帯用専用コンテンツ開拓
  - ワンセグ
- 実世界融合
  - 金融: お財布ケータイ, DCMX, . . .
  - 位置: GPS
  - 実世界認識: QR コード, トルカ, 世界カメラ, . . .
- 過剰付加価値社会
- デバイスコンバージェンス iPod touch, iPhone, iPad, . . .

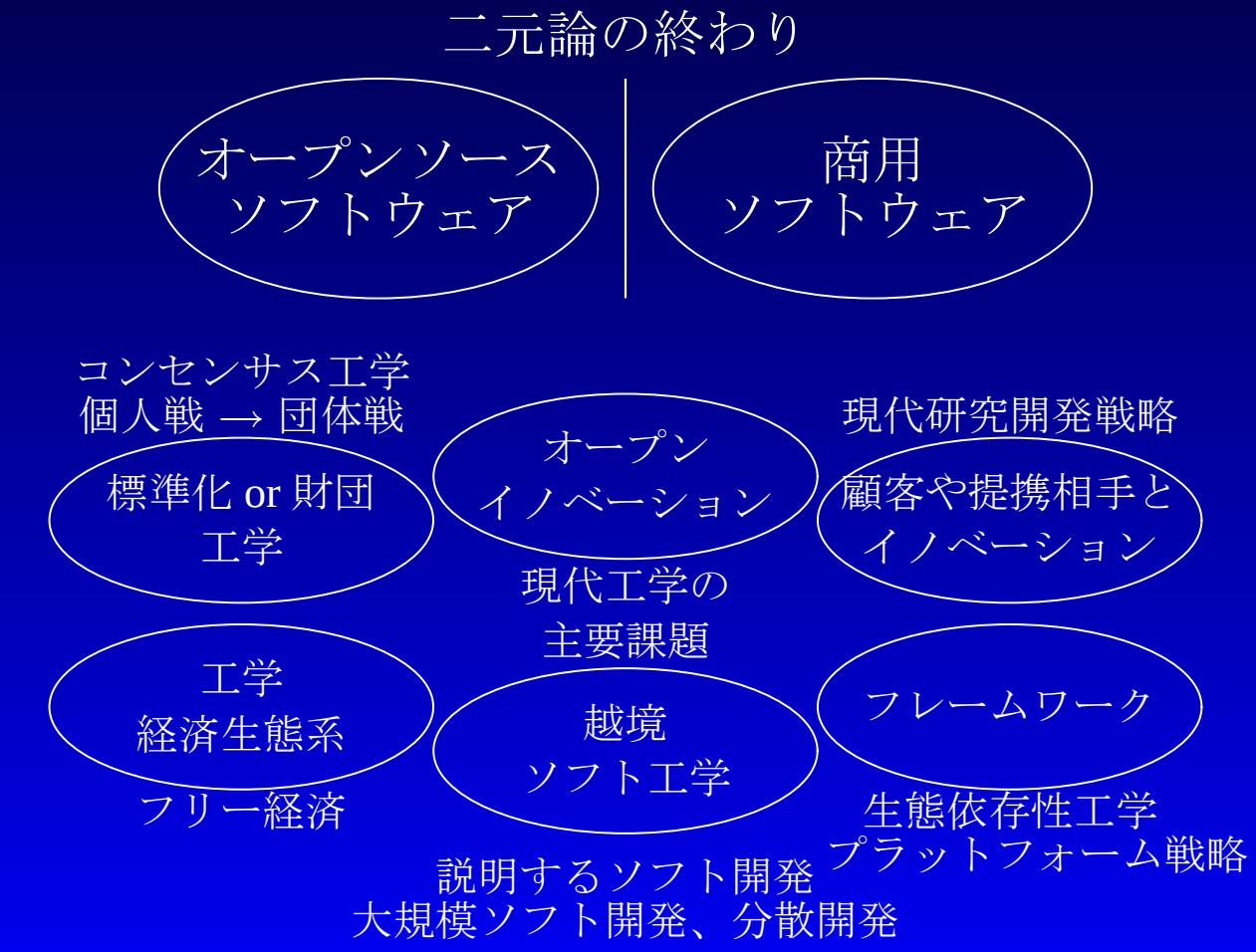
サービスイノベーション

- ディジタルコンバージェンス → サービスイノベーション
- 買いたいけど時間がない消費者 → 時間のマインドシェア

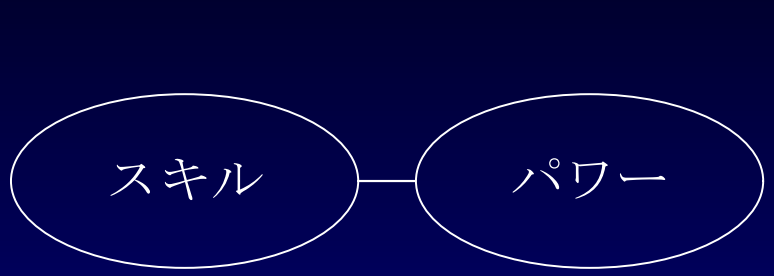


今日の話はOSS

- OSS (Open Source Software): ソフトウェアの著作者の権利を守りながらソースコードを公開することを可能にするライセンス（ソフトウェアの使用許諾条件）を指し示す概念 (Wikipedia)



OSSに見るプログラミングスキル転換



10 年前の
プログラミング

今の
プログラミング

- ソフトは資産ではなくて負債 (1 行でも書かないほうがいい)
- 80 %以上はフレームワークが仕事をしている
- 何をどう流用し、組み合わせるかを知っていることが決定的

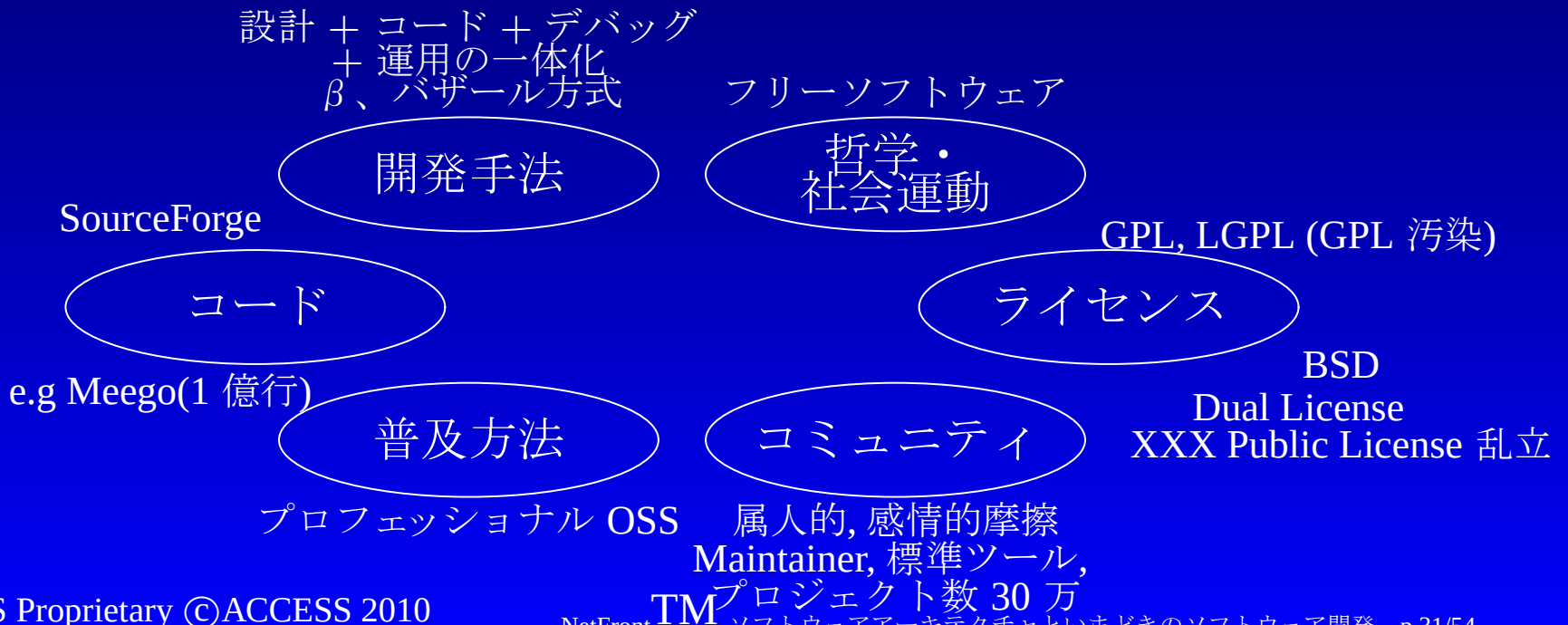
21 世紀のパラダイム
・外部資源の流用 (所有 → 利用) は、知識、ビジネス、ソフト開発、すべてに共通の潮流変化

OSSに見る複雑さ

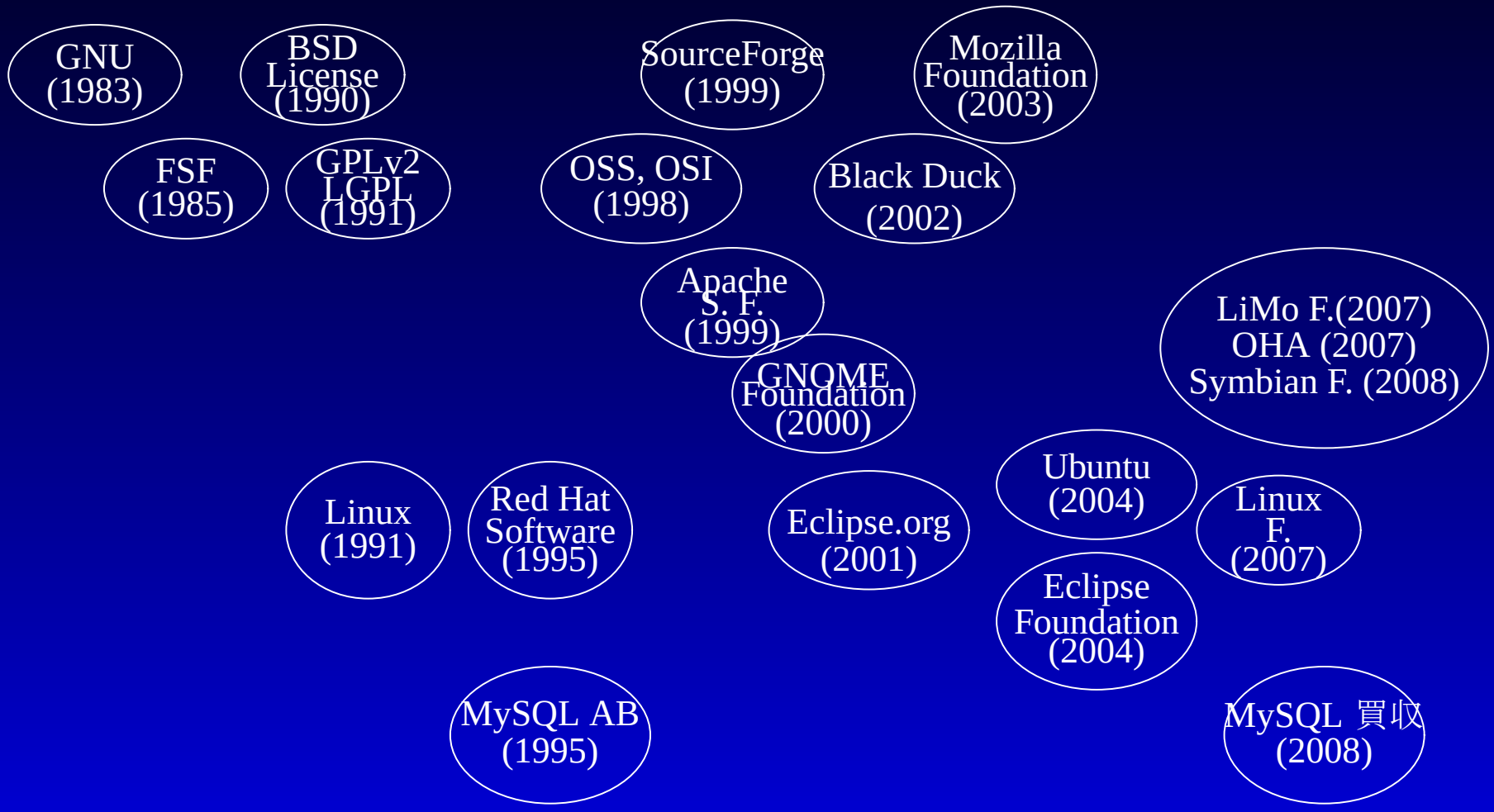
- 開発
  - 分散開発することは複雑さを呼ぶ 動的、継続変化、β
  - 商用よりも複数ライセンスの OSS のほうが管理が複雑
  - 動くこと以外にライセンス混入にも気を使う
  - 不確定さなどのリスク管理が複雑、隠れたコスト
- 対外対応
  - グローバルにオープンにすることは複雑さを呼ぶ 外部コミュニケーション、対応
  - Foundation や Alliance が複雑さを呼ぶ: 社内連携も複雑化
- ライセンス、知財
  - 必ず知財や法務や広報に確認などオーバーヘッドが増える
- 現代企業の OSS への理解、社内体制・教育への取り組みは発展途上

OSS とは？

- オープンソースはソフトウェア以外にもある
  - 知財の一部を無償公開し、周辺領域で事業運営する
  - ソフトの場合は OSS (オープンソースソフトウェア)
- OSS とは何かは非常に難しい (私だけ?)
  - 複雑な概念だという理解が不足 (OSS= 無料という程度の理解)
  - IT インフラ進化により加速
  - OSS はプロジェクトによって千差万別



[参考] OSSの歴史



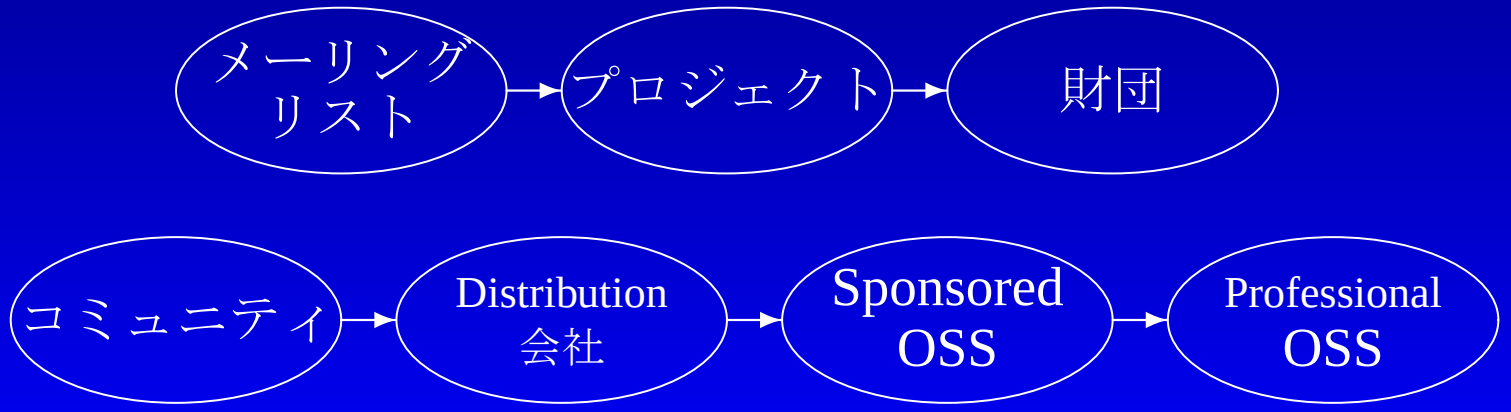
注 1: Red Hat は copyleft 注 2: Linux は今でもバザール方式
注 3: GNU/Linux はアプリも含めたパッケージ 注 4:BSD 系ライセンスは派生ライセンスが自由
注 5: Bugzilla(1999)

OSSのメリット

- ソフトウェアの開発期間とコストを圧縮
  - ライセンス料の削減
  - ソフトウェアの共通基盤化
  - ソフトウェアを自由に改良可能
  - 他社との情報共有の促進
- ソフトウェア利用企業にとっては明快

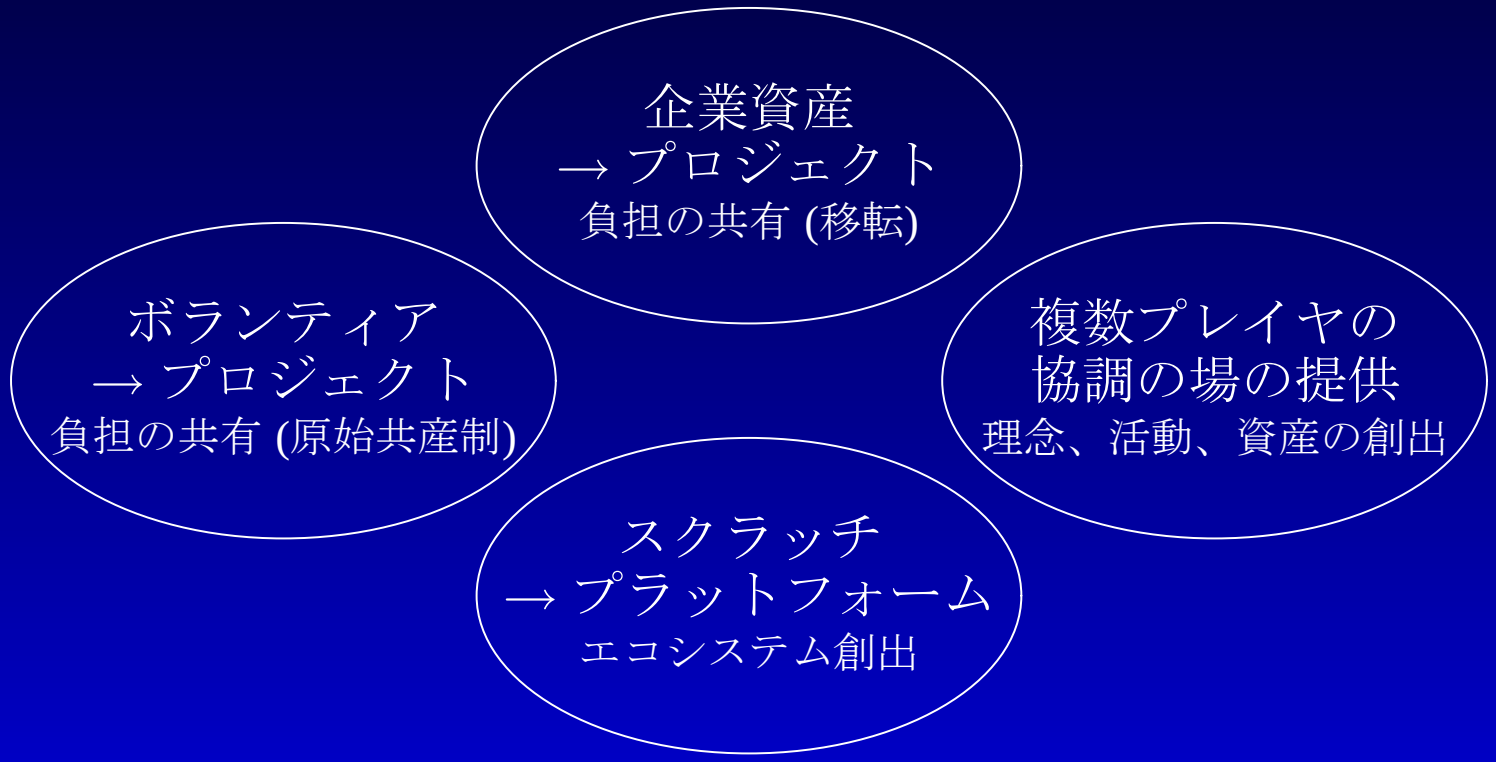
OSSの進化

- 優れた商用品質製品の提供 (Linux, MySQL, . . .)
- 企業開発環境に匹敵する開発環境の整備
- コミュニティの成熟 (OSS の商用利用の許容)
- Foundation 化
  - GNOME, Apache Software F., Eclipse, . . .
- OSS ビジネスモデルの進化
  - 買収: JBoss → Redhat (2006.6 350M ドル), MySQL → Sun Microsystems (2008.1 1B ドル) . . .



[参考] OSSの展開

- 大規模化: ローマは一日にしてならず
- きれいなイノベーション → でこぼこのイノベーション



OSS企業の成長

- ソフトを自分で書く
- デュアルライセンスで提供
  - 無償のオープンソース・ライセンスでは、ユーザー・コミュニティに品質の高いコードを提供し、その代わりに拡張案およびバグ報告とその改修法を入手
  - 有償ライセンスはオープンソースを使用するプロプライエタリ・コードを販売したい顧客のためのライセンス
    - 無期限のライセンス
    - 1年ごとに料金を支払えば1年に1回以上メジャー・アップデートを行うことができ、通常の技術サポートと保証が得られる
- 収入の65 %はライセンス、35 %はサービス
  - サービスは研修、サポート、コンサルティング、等
- 有力企業に買収される

OSS とはコード:携帯電話ソフトウェア関連 OSS 価値例

Project	行数 (K) (2010.3)	人年 (2010.3)	推定価値 (百万米ドル)
GTK+	569	156	8.57
BlueZ	109	27	1.51
Gstreamer	1,014	281	15.45
D-BUS	91	23	1.24
Glib	258	67	3.71
Gconf	38	8	0.49
Matchbox	86	21	1.17
Sqlite	113	28	1.57
Webkit	1,505	428	23.56

Note: www.ohlor.net 推定価値は COCOMO 2.4 モデルによる

[参考] OSSの価値

Project	行数 (K) (2010.3)	人年 (2010.3)	推定価値 (百万米ドル)
Google Chrome	2,563	744	40.94
Maemo	3,109	910	50.07
Symbian	39,754	13,087	719.77
Android	5,557	1,653	90.92
Linux 2.6	8,413	2,639	145.16
Debian	47,219	15,964	878.00

Note: www.ohlor.net 推定価値は COCOMO 2.4 モデルによる

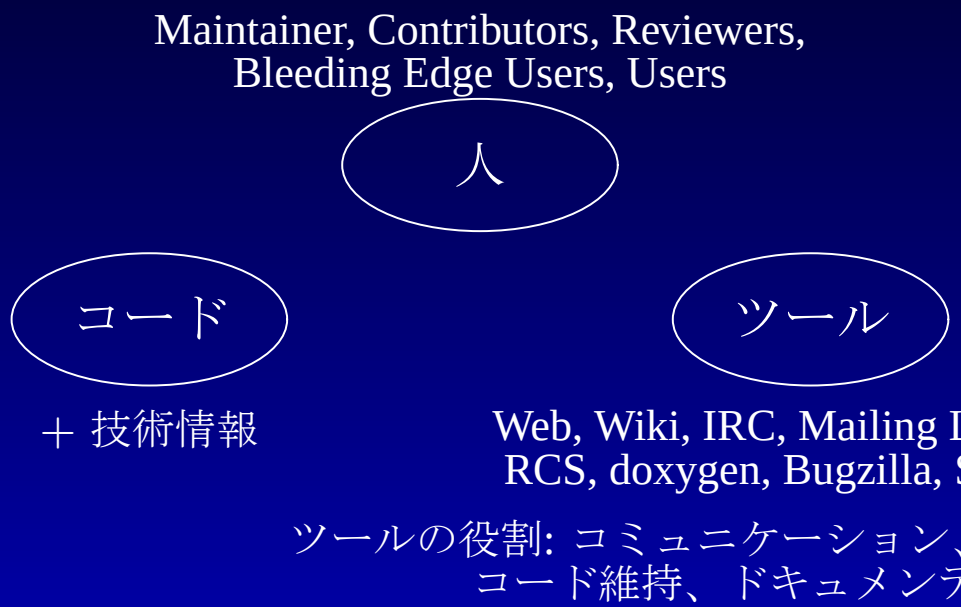
- 携帯電話ミドルウェア
  - Android, Symbian, Moblin, MeeGo の 100 %が OSS
  - Maemo の 95 %, LiMo の 91 %が OSS
  - 年間 11 億台の携帯市場の 40 %を出荷する Nokia でもミドル維持コストに耐えられない

OSSとは: 開発手法

- 開発者とユーザのコミュニティ
  - 使い、改良し、共有し、貢献する原則
  - 共有されるウェブベースプロジェクトとして視覚化可能
  - 会社に入る、と同じ、ある作法に従う開発の仕方の選択
- 要求抽出はコミュニティ主導
  - 要求条件はよく理解されているという前提 (国際標準の場合等)
  - 多くの議論は実装にかかわるもの
- 実装
  - 小さなチームでインクリメンタルに
    - 大きなプロジェクトは少ない
    - 例外: SCTP (Stream Control Transmission Protocol), IPv6, Crypto, SELinux (Security Enhanced Linux), LVM2(Logical Volume Manager Version2)
    - 個人とひととなりをよく知っていることが重要

OSS とは: コミュニティ

プロジェクトの 3 要素



ツールの役割: コミュニケーション、開発、バグ管理、
コード維持、ドキュメンテーション

- よい OSS プロジェクト
  - 強いコミュニティ、確固たるユーザベース、メジャーディストリビューションでの採用、安定したリリースサイクル

コミュニティとの関わり

- オープンソース化する知識が企業に不足: 証明、仲間としての認知
- OSS はギルドであり、掟であり、属人社会: IRC, メーリングリスト, Bugzilla, プロジェクト貢献
  - 例: 偶数バージョンは安定版、奇数バージョンは試用版: プロジェクトによって違う
- 時間を使い、仲間を知り、負担を担って信用される
- 望ましい例
  - レビューやバグ修正に参加
  - 修正予定を周囲に連絡
  - 小さな改変たくさん、がよい → よい住人
  - 不安定でも何回も貢献
- 半年作業して 15 万行突然一括貢献 → 他者作業の無視、反発呼ぶ

[参考]仲間にはいる方法

- 自分を知ってもらう (1)
  - IRC(チャット) にはいって話しかける
  - メーリングリストに入ってメール交換する
  - 自分がどのようなソフトを作ろうとしているかをチャットやメールで話す
  - 変なおたくジョークをチャットで披露
- 自分を知ってもらう (2)
  - Bugzilla(バグ管理ソフト) の中からバグを取り上げて直す
  - Project の作業項目の中から何かを貢献する
- 相手を知る
  - Maintainer, Contributors を知る
- 文化、言語、時差、の壁

OSSとは: 普及方法

- 部分的にオープンで周辺領域で稼ぐ: 自然なビジネスモデル
  - フリークライアント、オーサリング、民放、検索エンジン、試供品、アイテム課金、情報通信国際標準、QR コード、科学、...
  - オープンの利点
    - 利用者による情報伝播
    - ステークホルダーの結集
    - 競争相手の排除
    - 急速な市場拡大
  - ソフトの特殊性: ディジタル、経験財
  - 情報通信コスト激減時代に適応
- よく考えると私有ソフトで稼ぐほうが不思議
  - OSS が不適当、OSS と戦って勝てる理由の明確化が必要
  - 過去のツールやプロジェクト管理の優位性は揺らぐ
- イノベーション (社会技術変革) の一手法

[参考] OSS 商用利用の欧米の見解差

	欧州	米国
オープンソース採用の主な理由	ベンダー・ロックインを回避	コスト削減
商用オープンソースを推進する主要因	ローカル・ソフトウェア産業の創出	ベンチャー・キャピタルや企業主導。ビジネスをつくりだし、投資家が儲ける
デュアル・ライセンスに対する意識	疑問。オープンソースを PR やマーケティングに使用した商用指向なビジネスモデル	オープンソースビジネスモデルとして広く受け入れ
セールスモデル	流通主導。VAR や SI	直販
オープンソースビジネスモデル	サービスに焦点を置き、ソフトウェアは 100%オープンソース	焦点は製品に置き、商用アドオン、またはエンタープライズ版をオープンソース版製品と組み合わせ
オープンソース製品に期待されること	すべてのコードがオープンソースで提供。コミュニティ参加モデルのコミュニティ統治	欧州と基本は同じだが、すべての製品がオープンソースライセンスで提供されなくてもよい。プロジェクトは商業ベンダーが管理

http://lmaugustin.typepad.com/lma/2008/09/commercial-open-source-in-europe-verses-the-us.html
Larry Augustin blog

[参考] コミュニティ進化もいろいろ

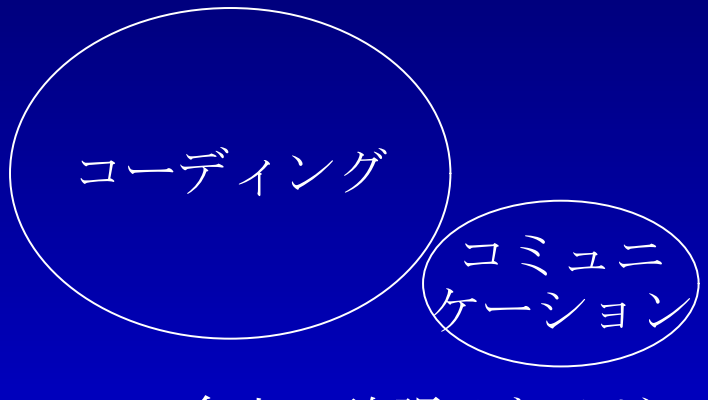
プロジェクト	発展系
GNU	GPL を生む思想運動。Mach マイクロカーネルによりフリーの UNIX 互換 OS GNU Hurd を作るという目的は四半世紀たっても他と比較しえるレベルでは達成していない。
Linux	フリーの UNIX をスクラッチから作る。いまでもメーリングリストベース。Linus のリリースを世界が尊重。定期的リリース (2.6.33 at 2010.2.27)。2000 年頃から企業フルタイム従業員 (HP, SGI, Intel) の貢献大。 Kernel Mailing List には誰でも修正を投稿できる。
GNU/Linux	Linux を動作させるには、ライブラリ、サブシステム、サーバ、アプリケーションが必要。その多くが GNU で提供されており、事実上、相互依存している。実態として一体で OS として機能する。
Eclipse	IBM プロジェクトから脱皮し、Eclipse Foundation へ脱皮。 当初から IDE のプラットフォームを作ることを目指していた。ＭＳＰゴシック有力企業が戦略メンバとして評議委員会に参加し、コミット。

[参考] 成功した OSS プロジェクト例 (Apache)

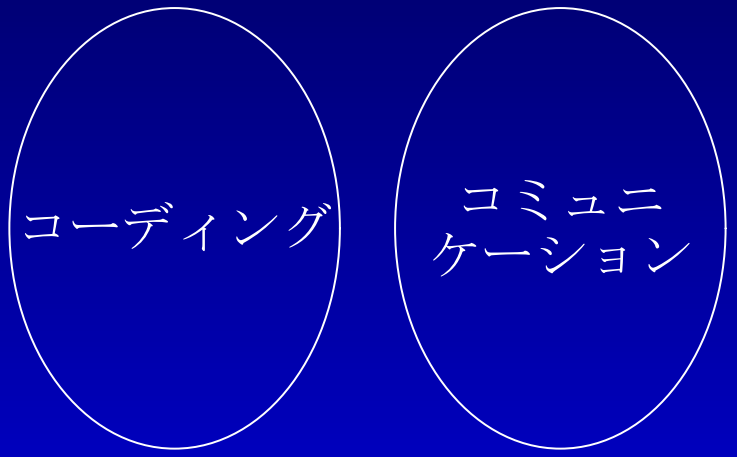
- NCSA サーバの放置、有志メンテ、再度放置
- 1999 年にメーリングリストで再開
- 非営利団体、構成員は無報酬、物理オフィスはない
- Apache Software Foundation (2010.2 末) 構成
  - ユーザ: 誰でもなれる
  - コミッタ: 活発なユーザから選ばれ、書き込み権限がある (418 人) (日本人名は Shinobu Kawai, Takashi Sato, Koji Sekiguchi, Shinsuke Sugaya)
  - オフィサ: コミッタから選ばれ、プロジェクトの管理 (90 人)
  - メンバ: 活発なコミッタから選ばれ、選挙権と被選挙権がある (266 人)
  - 理事会: メンバから選ばれる 9 人の理事。毎年選挙
- Apache HTTP server だけでなく Apache ブランドのプロジェクト (例 Apache Hadoop)
- 2008 年に Microsoft が資金提供

オープンであること

- オープンであることは痛みを伴う
- プロジェクト毎にやりかたはいろいろ
  - 汗を流した人に決める権利がある
- 大規模であれば、実は OSS でなくても課題は同じ



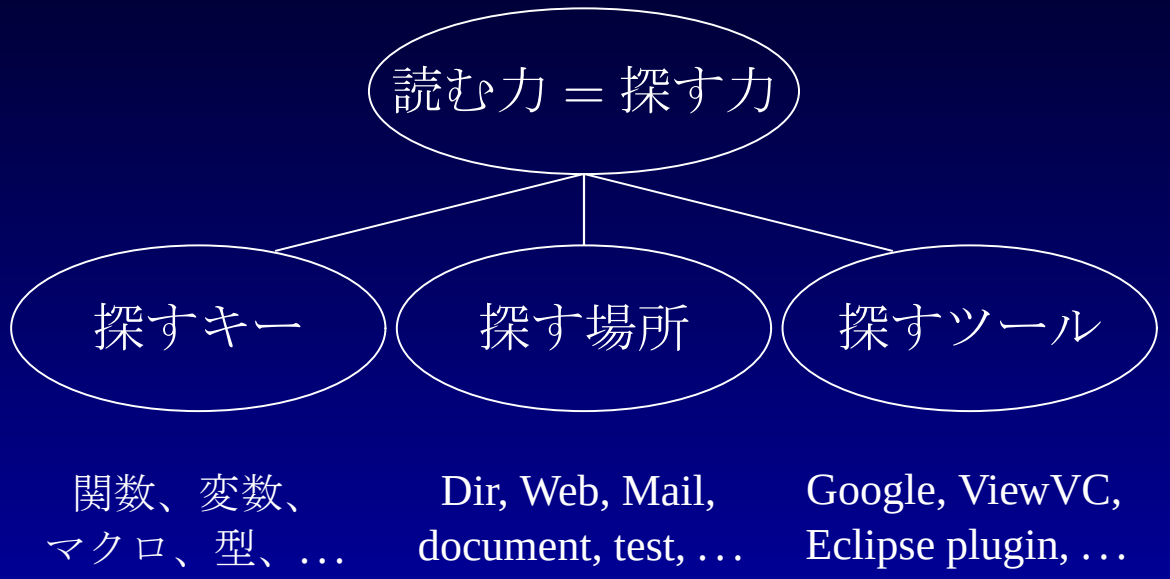
多少の誇張はあるが...



大量のコードを読む

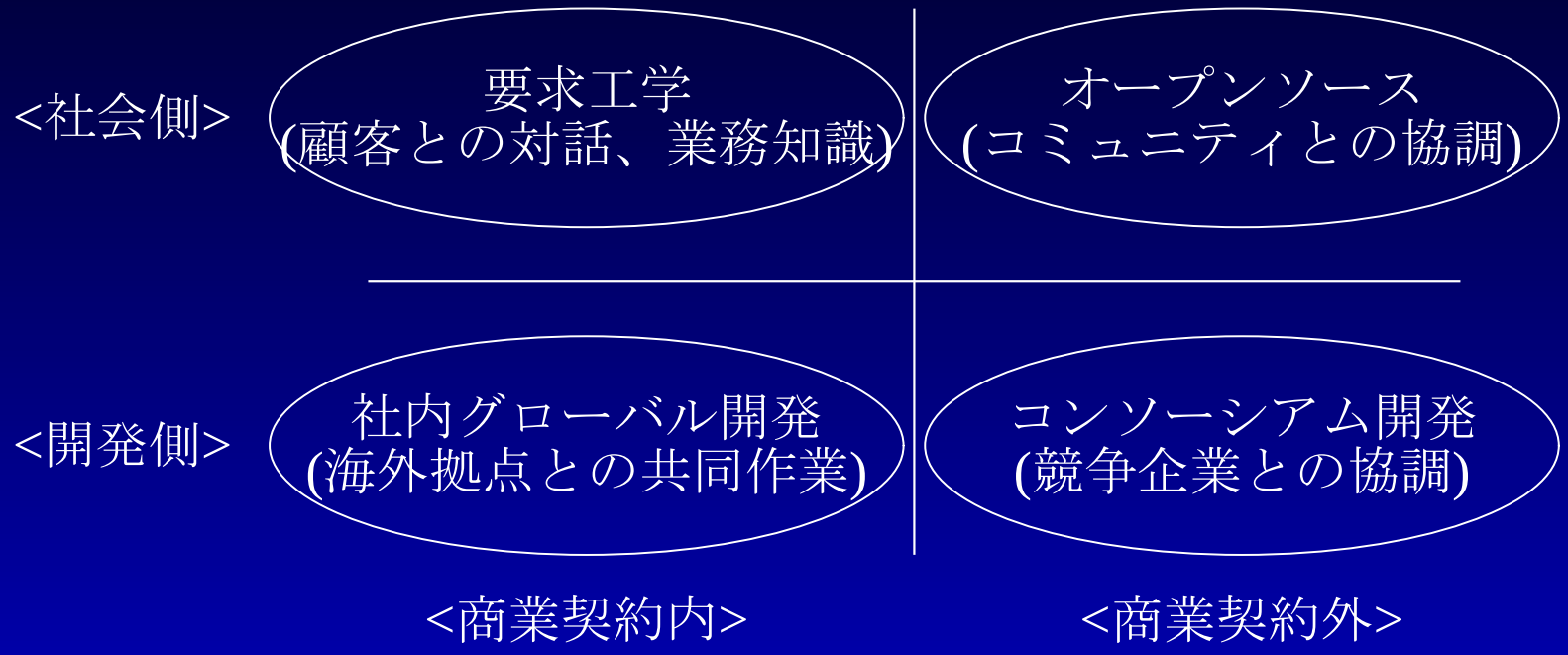
- 大量のコードを読む
  - そうはいっても何読むの？
    - Linux Kernel? GNU? GNOME? Android?
  - Adobe Open Source / http://sf.net/projects/adobe-source (C++ です)
    - 石黒邦宏 (ACCESS CTO) 曰く、「凄いコードなんでびっくりした。凄いハッカーが作った凄いコード。ここ五年間に見たコードの中でいちばん凄かった。皆、これを勉強しなきゃ」
  - 知らない OSS は怖い → ルータソフト Zebra http://www.zebra.org/
    - 石黒邦宏 (ACCESS CTO) 作成の OSS
- 読む経験
- 読む勘所

大量のコードを読む: 読む力＝探す力



- コード専用検索エンジンもある: Koders (Koders.com), Krugle (http://www.krugle.com/), Google (http://www.google.com/codesearch)
- フレームワークの理解: フレームワーク対応ツールで読む (現代プログラミングの 80 %以上はフレームワーク)

越境 (Cross-boundary) ソフト工学



- 境界越えを加速する要因：情報通信コストの劇的低下

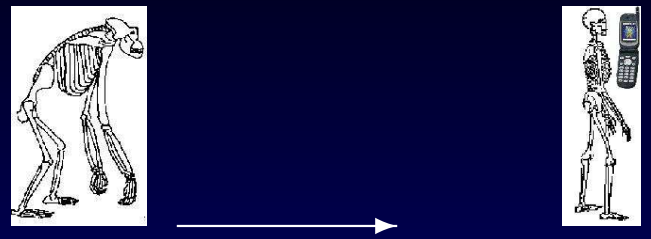
OSSへの取り組み

転換経営
ビジネスモデル競争
OSS 専門家の定着支援
一部の教育はユニバーサル
OSS の組織化に対応

知の質的転換の教育
フレームワークに基づく共同作業教育
リアルに完成動作する喜び
オープングローバルに前向きな姿勢
OSS アライアンス開拓

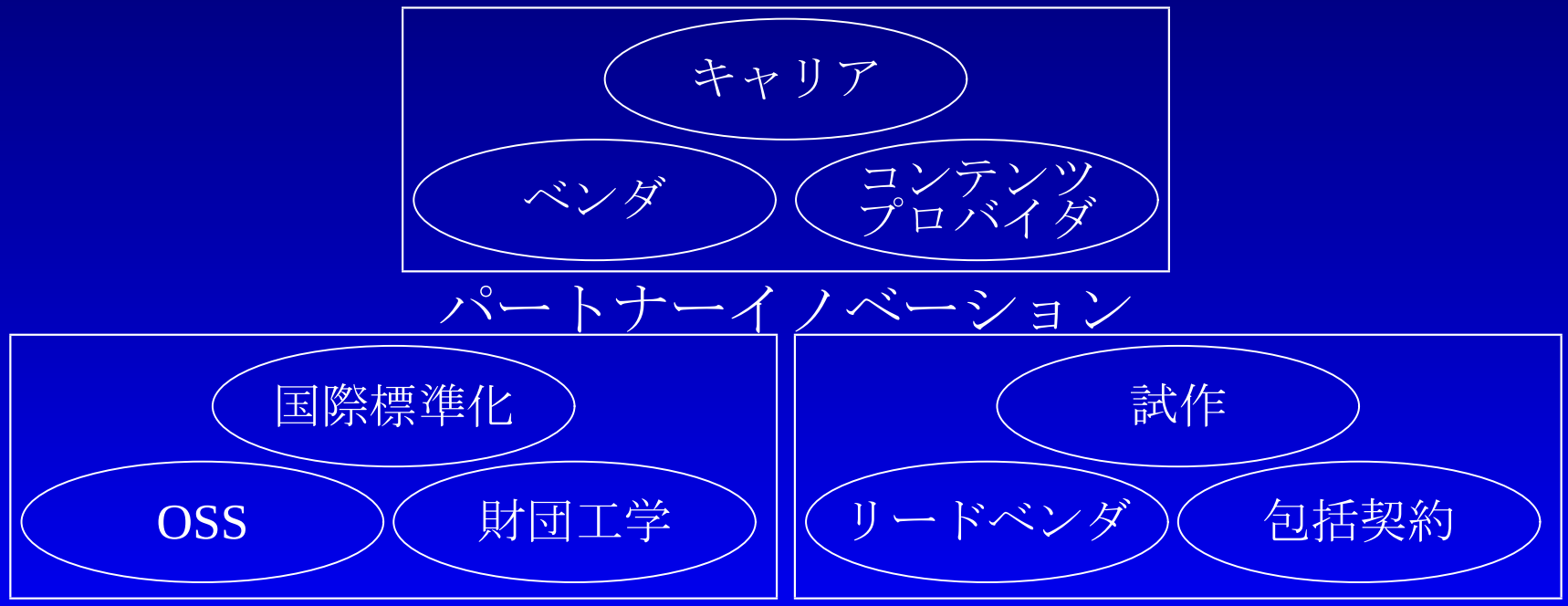
- 個人的には疑問: 現在のギルドや独裁主義のような OSS
- OSS の未来形への試行錯誤は今後も続くだろう
- 商用化の波: インターネット商用転換期 (90 年代前半) のデジャブ

世界は変えられますか



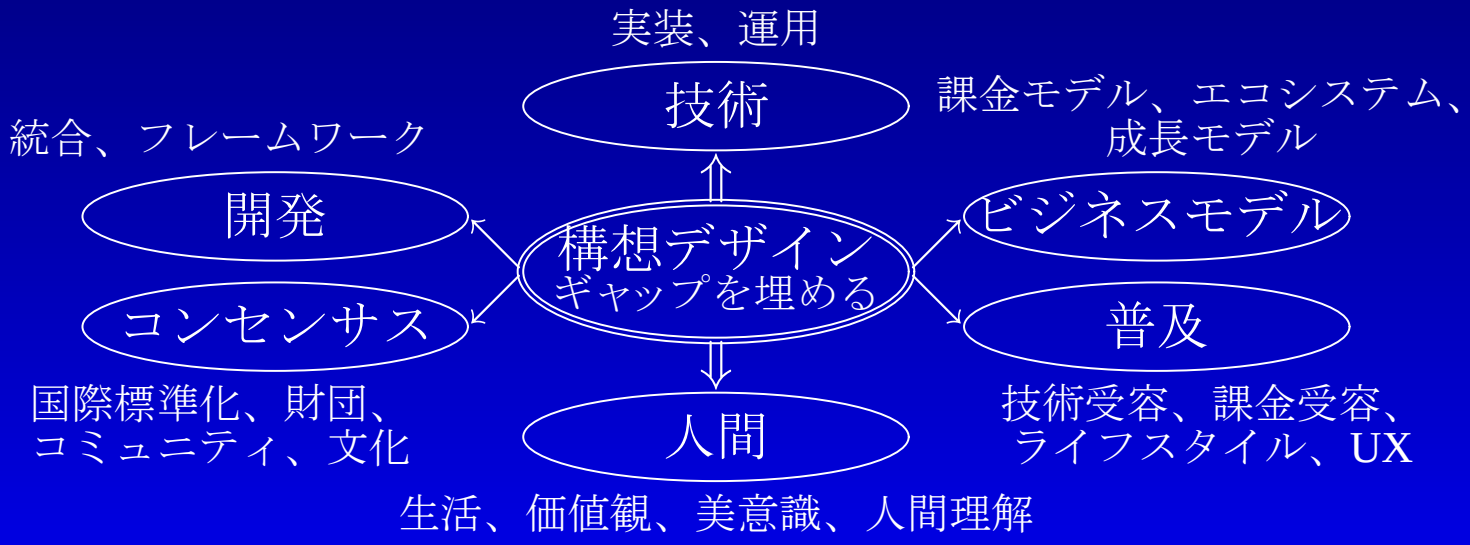
軍隊、国家、組織 → 個人
人を動かす 3 要素：武力、財力、知力
個人が情報発信：情報、金、委任、信用 (経済の源泉)

- イノベーションはさらに複雑になっている
- 簡単ではないが「やってくれ」と頼まれる位置にいればできる
- 作るのは楽しいがそれだけをやっているわけにはいかない



問題とともに生きる: 構想デザインの仕事

- 金、自由、達成感: すべては手に入らない どれかを選ぶ
- 目的と手段: 手段は目的を達成する道具、目的が達成されれば手段は何でもいい
- 誰にもできないような問題定式化をすることが重要: 研究でも
- 難しい問題を明確に捉える位置にいることが自分の価値を高める。
  難問解決は限界を超えたコラボレーションから
- 失敗を畏れない。失敗して成長する
- できない理由を考えるな。賢い人間ならできない理由はいくらでも作れる



むすび

- 情報処理は人類の本質: 科学 + 技術 + 政治 + 経済
- オープンソースや共同体エンジニアリングのインパクトは絶大 → ソフトウェア産業の歴史的転換点
- さまざまなコミュニティ的ソフトウェア開発手法が台頭、普及しているが、プロセス的にもビジネスモデル的にも改良の余地は大きい
- エンジニアリングとは目的を達成するための手法を体系化すること
  - 作るから組み合わせて使う、へシフトしているが、そのための人材育成やエンジニアリング技術の体系化は未成熟
  - 教育する人材育成から始めて 20 年かかる (書くから読むへ, 等)
- 世界最高のチーム、ビジネスモデル、製品で世界に挑む それでも ...
- 世界を変えることはできますか。できる。個人戦から団体戦へ
  - イニシアティブをとろうと思って世界は牛耳れるものでもない
- ディジタル民族大移動は本質的変化。サービスは人間理解への挑戦
  - すべての教育体系を刷新しなければいけないほどの変化