



usp lab.

アジャイルジャパン 2009
株式会社良品計画 事例セッション

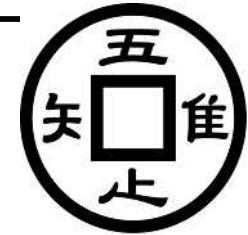
スピードがすべてを駆逐する

良品計画の情報システムを支える
「ユニケージ開発手法」とは

2009/4/22

ユニバーサル・シェル・プログラミング研究所
當仲寛哲

Universal Shell Programming Laboratory



usp lab.

1. ユニケーション開発手法とは

・ニーズをいかに早く具現化するかに特化した手法

パッケージ(機能があらかじめパッキングされている)に対して、
ユニケーション(UNIX Uniq Unify)はスクラッチ開発の高効率化手法である。

・「安く、早く、柔らかく」がキーワード

コストを「安く」

- ・安いハードウェア(PCベース)
- ・安いOS(Linux)
- ・安いアプリケーション(シェルスクリプト)
- ・安いデータ管理(テキストベース)
- ・システム更新無し(移植性抜群)

期間を「早く」

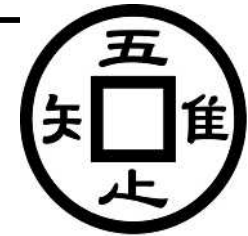
- ・短いアプリケーション
- ・試作型(テキストデータがあればすぐ作れる)
- ・情報系、業務系、更新系すべてシェルスクリプト
- ・オールインワン方式 (シェルの依存関係をつくらない)

「柔らかく」

- ・ハード・ソフト・データの完全分散型配置(コンピュータズ・コミュニティ)
- ・Realbase®

Universal Shell Programming Laboratory

2. ユニケーゼ開発手法の歴史



usp lab.

- ・大手流通業の業革プロジェクトチーム
- ・金なし権限なし素人だけ
- ・生き抜くための工夫の数々
- ・大型汎用機を3000台の分散サーバに置換
- ・衣料品の業務改革
- ・世のため人のために働く決意

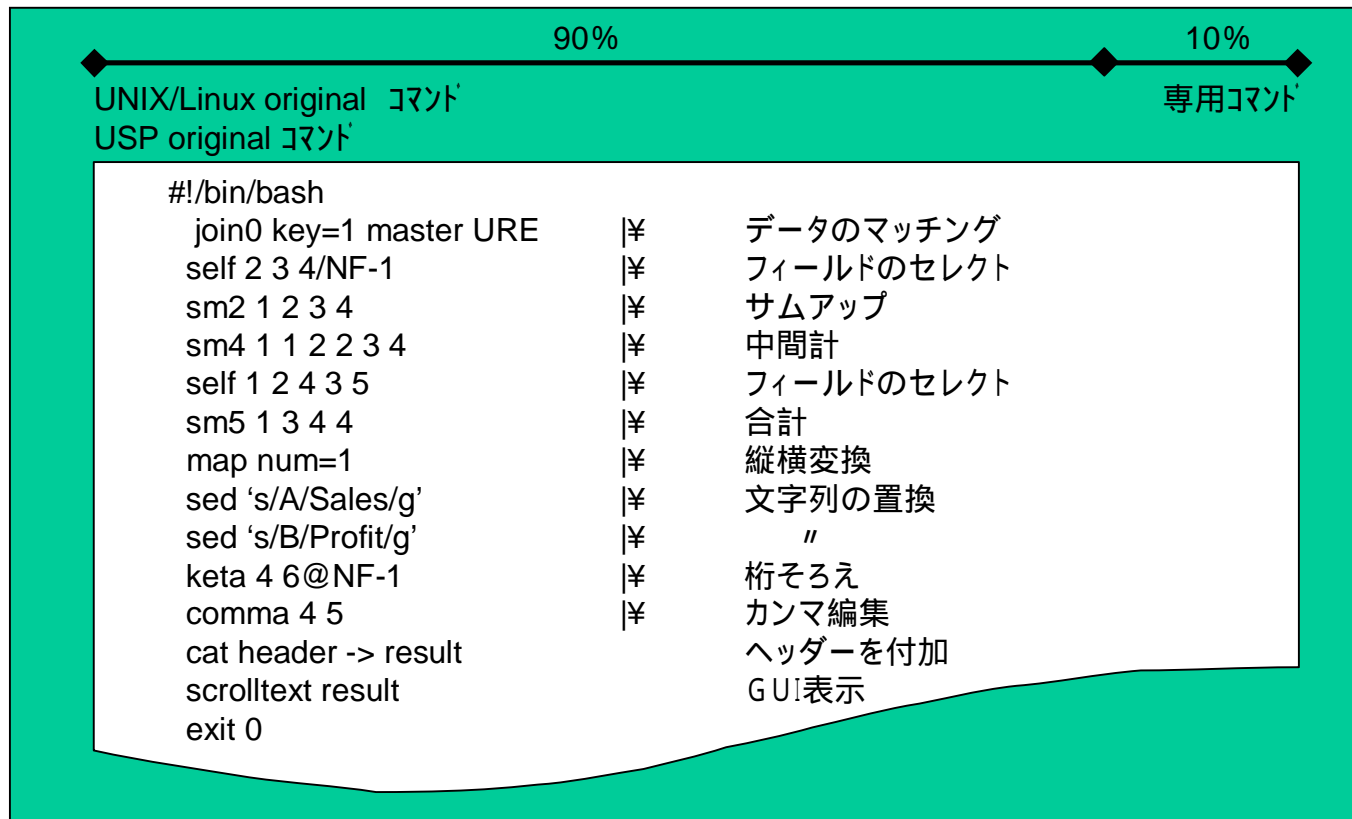
Universal Shell Programming Laboratory



usp lab.

3. シェルスクリプトでシステムを作る

約40年間変わることのないUNIX/Linuxの基本的な技術



90%の汎用コマンドと必要に応じて開発したアプリ専用コマンドをシェルスクリプトで使用する

Universal Shell Programming Laboratory



usp lab.

4. コマンドを自作する

すべてのプログラムはフィルタである 一番明快なフィルタはコマンドである

よく使うものは100個もない

| | | |
|-------|-----------|------------------------|
| cp | コピー | Linux オリジナル コマンド |
| find | ファイルの探索 | |
| uniq | 重複を除く | |
| sort | 並べ替え | |
| ⋮ | | |
| self | フィールドセレクト | USP オリジナル コマンド |
| join0 | データのマッチング | |
| sm2 | サムアップ | |
| waku | 枠付け | |
| ⋮ | | |

- ・単機能を極めた独立コマンド群
- ・簡単な使用法(マニュアルレス)
- ・コマンドをC,Java,PHP,perl,awk等で作る
- ・単純なコマンドを組み合わせることで簡単に複雑な業務処理を行えるコマンドセット

Universal Shell Programming Laboratory



usp lab.

5. テキストファイルベース

思想: データが表現できる情報はなるべく自由であるべき



テキストにすれば自由な情報があつかえる
タテ型テキストデータ誕生

```
<name>  
<age>  
</age>  
</name>
```

XML形式

```
_name: Suzuki  
_age: 30  
_hobby: sports
```

タグ形式

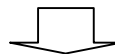
```
Suzuki 30 sports  
Yamada 23 guitar  
Tanaka 41 golf
```

タテ型ワープロ形式

項目の追加 ... ファイルの追加
項目長の変更 ... 可変長なので自由に変更できる
履歴の保持 ... タテ型なので履歴を持つのは簡単

ただし、これらを扱う
コマンドは必要

UNIXのコマンドは、元々文章処理用として開発されているものが多い。



機械的な処理効率は悪いが、激的なハードウェア性能の向上で解決

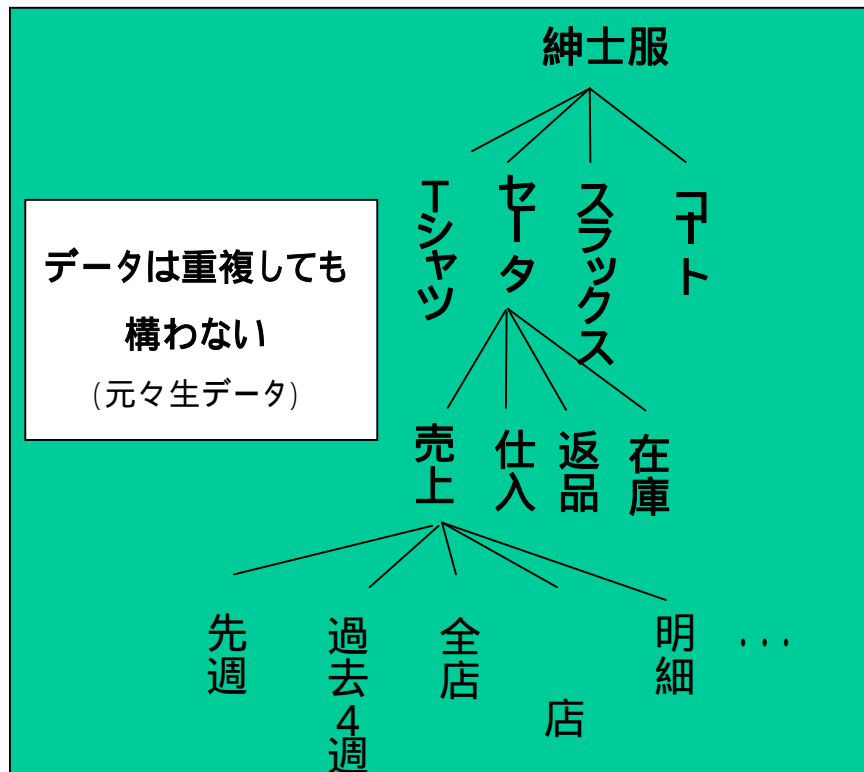
Universal Shell Programming Laboratory

6. データ整理術



usp lab.

紳士服データ分類例



冗長なデータを作り自由度を高める

Tシャツの売れは……？



先週の売れは枚です！



人が情報を口にする時は、コードではなく言葉である。



データは会社で使われている言葉を使って分類。

UNIX/Linuxディレクトリ体系を、言葉による表現形式にあてはめ、ファイル分類に使用。
(もともとUNIXはこの思想 素直に従う)

Universal Shell Programming Laboratory



usp lab.

7. 開発環境

- 開発環境 vi と cp (コンパイル、ビルド作業無し)
作成シェルスクリプトを、コマンドラインから実行する
- デバッグ手法

変数やファイルを表示 (printf手法)
して内容を確認する

```
#!/bin/bash  
-----処理-----
```

```
echo $variable  
cat $file  
tee コマンド
```

コマンドエラーステータスの表示

```
echo $?  
echo ${PIPESTATUS[@]}
```

パイプの切断 (切り分け)

```
#!/bin/bash  
-----処理-----
```

```
command |¥  
tee debugfile |¥  
command |¥  
or
```

```
command |¥  
cat > debugfile ; cat debugfile |¥  
command |¥
```

ユニケーシ開発手法では、情報は変数ではなく、ファイルの中にある。
実データを使ったデータプログラミングで開発とテストを同時進行させる。

Universal Shell Programming Laboratory

8 . 運用監視



usp lab.

運用監視に必要なソフトウェア要件は、

すべてシェルスクリプトで記述可能

・走行ログ

```
#!/bin/bash -xv
exec 2> LOGFILE
```

```
-----
#!/bin/bash -xv
SHELL_PROGRAM 2 > LOGFILE
```

```
-----
tail -f コマンド
```

・通常ログ

loggerコマンド (syslog)

・処理報告

```
scp MESSAGE $remote_host
```

```
-----
while ["$TIME" -lt "$LIMT_TIME"];
do
  sleep 1 ;
done
```

・二重化、バックアップ

H/W RAID , rsync ,ロードバランサ

・処理順番制御

```
while [ ! -e SEMAPHORE ]; then
do
  sleep 10
done
```

・ジョブ制御

```
&,$!,kill,wait,setsid
```

・エラーステータスによる制御

```
test,&&, ||
```

・時間起動

```
cron,at
```

・トラブル報告

```
email,swatch
```

・定期リブート

```
reboot,fsck
```

Universal Shell Programming Laboratory

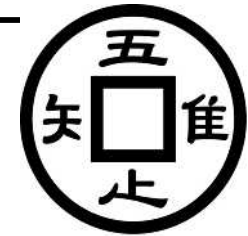
9 . 技術的なFAQ



usp lab.

- **排他制御**
ロックファイルをセマフォとする処理単位の制御
(排他ファイルオープン、fcntlシステムコール)
ファイルは直接更新しないので、ファイルやレコード自体にロックはかけない。
- **順番制御**
繰り返しサーバによる処理順番制御
- **コミット**
ファイル書き込みコマンドのステータスで判断
(ファイルシステムを信頼 MySQL、PostgreSQLと同じレベル)
- **ロールバック**
明細の履歴が保存されているので、基本的に再生成可能
- **高速検索**
レベル4データ

Universal Shell Programming Laboratory



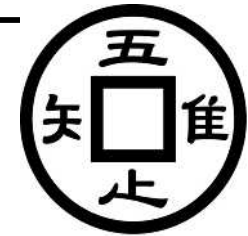
usp lab.

10. ユニケーシ開発手法の思想

- ・大規模で複雑なシステムを分担せず効率よく開発したい
セル型開発方式
- ・プログラムは作り捨てにしてもいいぐらい簡単にしたい
シェルスクリプト化
- ・仕様変更柔軟に対応したい
ハード、ソフト、データの分散化(依存関係を無くす)
RealBase
- ・少しでも楽に開発したい
現地駐在型(顧客との一体化)
ペアプログラミング(気づきと啓発)
ツール化
開発環境レス(コピー容易性)

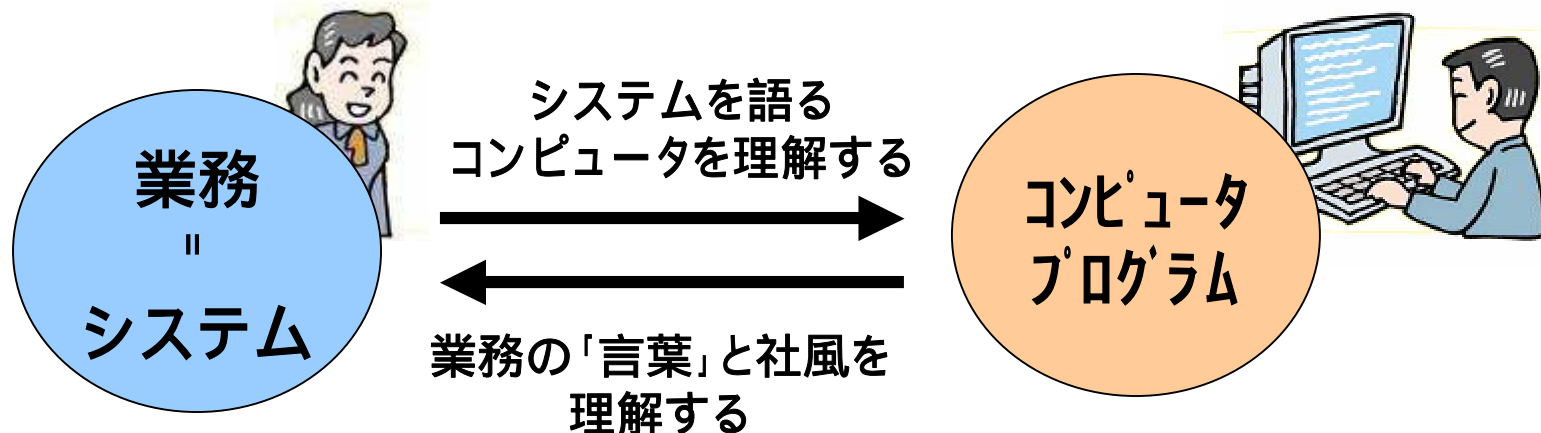
Universal Shell Programming Laboratory

11. ユーザー一体型開発



usp lab.

- 業務とシステムは表裏一体
- システムは業務にある
- 業務を理解している人は、システムを語り、コンピュータを理解するよう努める
- コンピュータを操る者は、企業の言葉を身につけ、業務とシステムを理解するよう努める



Universal Shell Programming Laboratory

12. ユーザー一体型開発



usp lab.



良品計画の開発現場の様子

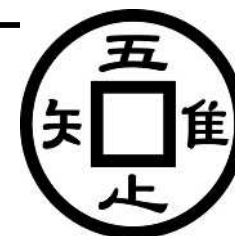


必ずお客様の部署の一員として仕事をします。



開発はお客様と一緒に…。

Universal Shell Programming Laboratory



usp lab.

13. ペアプログラミング

- 先輩と後輩、修行型で仕事を進める

先輩



後輩

データの整理
アプリケーションの
ディレクション

会話を重視
価値観や目標
技術の共有

アプリケーションの記述



Universal Shell Programming Laboratory

14. コミュニケーション技術



usp lab.

・報告

簡潔に
指示者にまっすぐに。会えば必ず
結論 理由 経緯

・連絡

誰に知らせるべきか

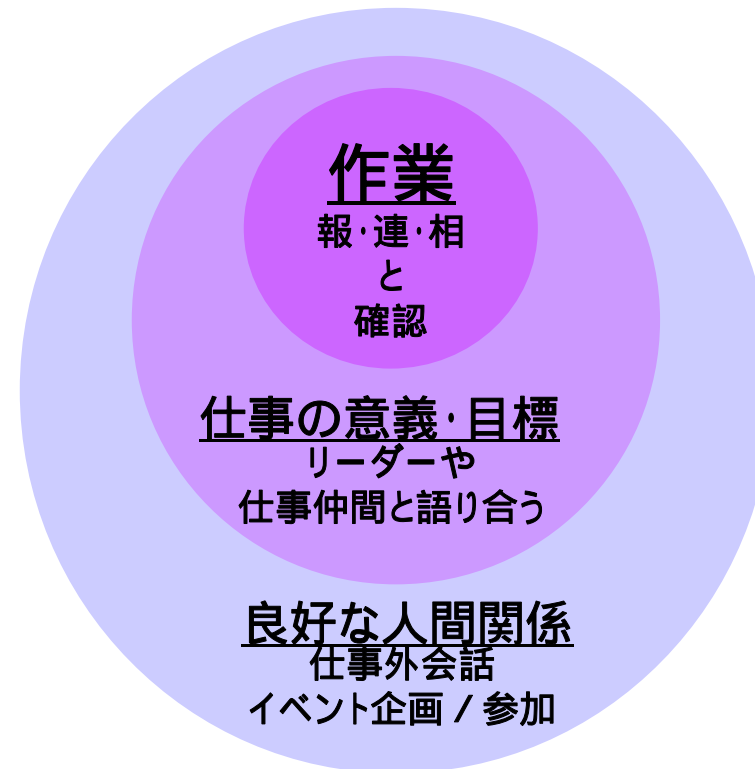
・相談

「相談があるのですが...」と冒頭に。
10分考えて分からないことは必ず
人に聞くこと。

・確認

自分の言葉で言い直し、
相手の理解を得る

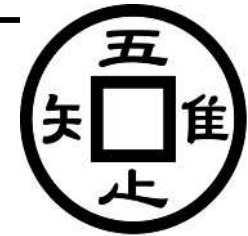
職場では、3レイヤーの
コミュニケーションが必要



公私融合・・・「七十にして心の欲するところに従えども、矩(のり)を踰(こ)えず」

Universal Shell Programming Laboratory

15. コミュニケーション技術



usp lab.

社内で様々なイベントを実施し、違う現場で働くエンジニア同士の交流を促進



春は古民家再生と農作業

夏は花火鑑賞



冬は餅つき

秋は稲刈り



Universal Shell Programming Laboratory

16. 非分業の原則



usp lab.

- 分業しない(= 自分で全てできる) に越したことは無い。
- 自分でやってみると失敗する。失敗することで学ぶ。
- 先輩のアドバイスのありがたさを知る。
- 自分で全てできる(セル開発方式)だとモチベーションが上がる。
- メンバー全員が自律的に仕事をして、チームワークや分業のよさを知る。
- プロの仕事に対するリスペクトが生まれる。

Universal Shell Programming Laboratory



usp lab.

17. 作り捨てとコピーライト

- 諦観と責任

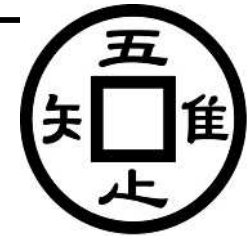
- ・アプリケーションプログラムは、過去の天才たちが作った基盤技術の上で成り立っている。
- ・だから、アプリケーションプログラムは、著作権を主張すべきでない。
自分が作ったものへのこだわりを捨てる
しかし、自分が作ったものについては責任を取る

- 新コピーライト

- ・コピーライトは「コピーをする権利」。
- ・人の書いたプログラムをコピーしたら、すべての責任はコピーした人のもの。
コピー元にあった間違いも、あなたの責任
シェルの作成者表示の書き換え
優れたプログラムしかコピーされない

Universal Shell Programming Laboratory

18. ワンプログラム・ワンフロー



usp lab.

- 依存関係の排除

- ・データはファイルで分ける
 - IFや、フラグ、区分を使わないようにする
- ・インクルード処理の禁止
 - 共通モジュールなどは、基本的に無い
 - 各スクリプト内ですべて「ベタ」に記述する

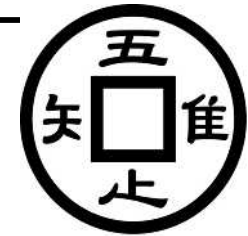
局所的改変は、全体に影響を及ぼさない

改変の自由度が増す

一斉書き換えの必要に迫られても、手間は少ない

ありうる制度上の改変はテーブルファイル化し、プログラムをできるだけ直さないで済むようにする

Universal Shell Programming Laboratory



usp lab.

19. ドキュメンテーション

・ユニケース開発手法に即した、実務的なドキュメントのみ

ネットワークやハードウェア全体図は必須

タイミングチャート、データ配置図、バッチ処理概要などは、
必要に応じて書く

不文律のシンプルなルールがあり、それに従っていれば、後はスクリプトが読めれば分かるような構成になっている

システム全体を理解するための資料は必須ではない

システム = 業務なので、業務を理解するための全体図はあったほうがいい

「その仕様書さえ見ればプログラムが書ける」ような、
「詳細仕様書」は不要

プログラマには、インとアウトを示し、フローの概略を伝えるのみ
そのための簡単な説明書を保管している

一般的に必要とされるドキュメント

システム概念図・情報一覧表・業務シナリオ一覧表・業務フロー・画面一覧表・
帳票一覧表・画面メニュー体系図・総合テストシナリオ etc.....

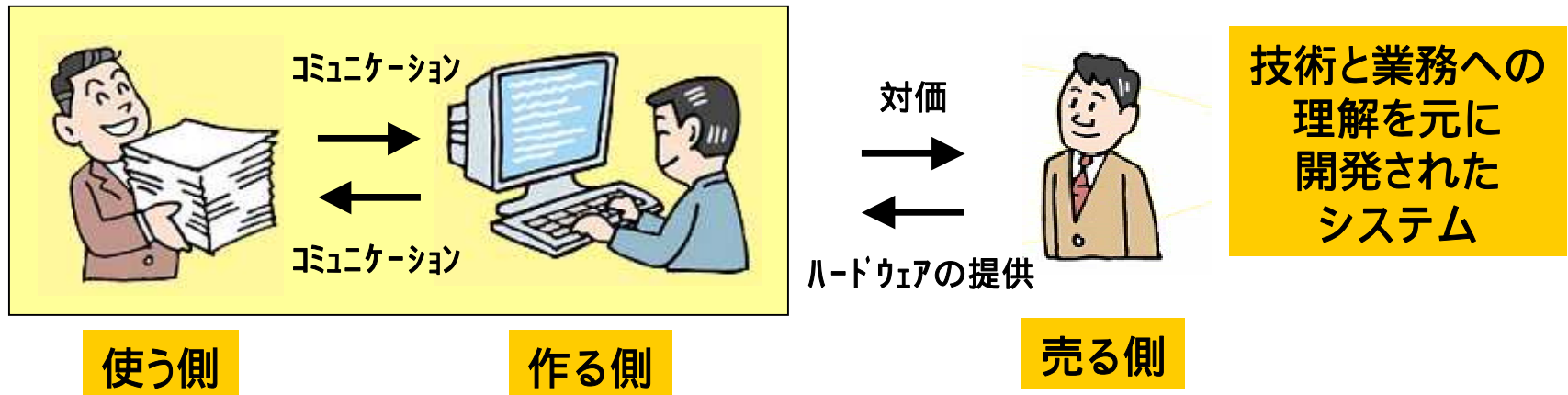
Universal Shell Programming Laboratory



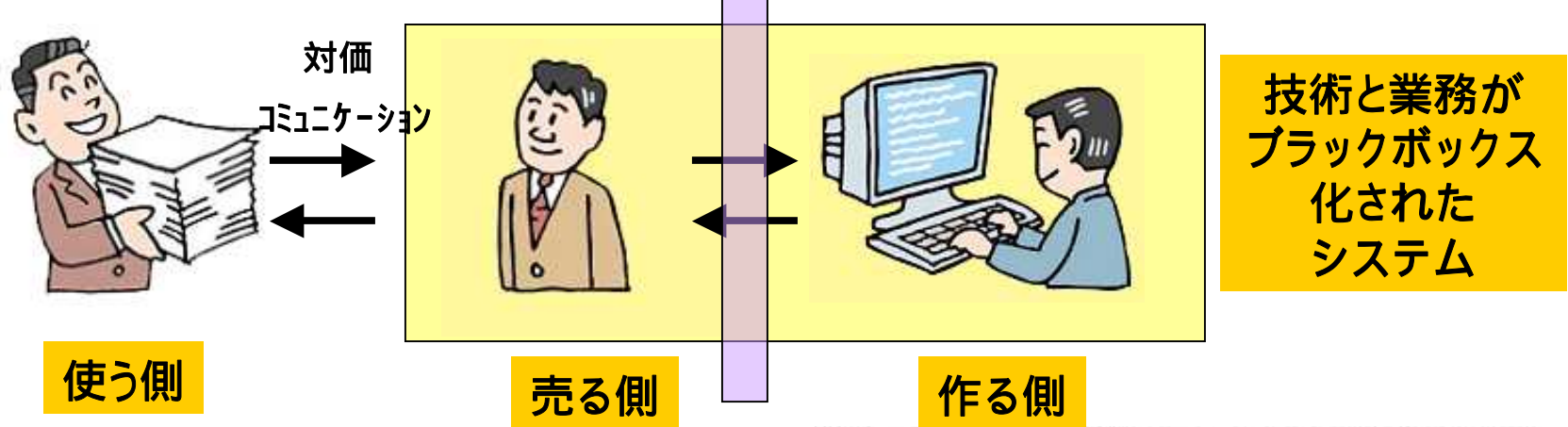
usp lab.

20. 作る側・売る側・使う側

アプリケーションソフト登場以前

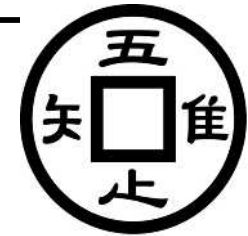


アプリケーションソフト登場以後



Universal Shell Programming Laboratory

21. 誰がソフトウェアを作るのか？



usp lab.

ソフトウェアは工業製品ではない

- ・料理のレシピと似ている
- ・正確に仕様を記述できない
- ・作り手のセンスやディテールが、良し悪しを決める

- ・規格と製造操作を示すと、繰り返し同じ品質の製品が作れる

・良いソフトウェアを作るには

プログラムのガイドライン + 優れた Super Engineer
人財をどう継承、発展させるかがポイント

Universal Shell Programming Laboratory

22. スーパーエンジニア(新SE)の育成



usp lab.

- 新7Kからの脱出
「きつい」「帰れない」「給料が安い」「休暇が取れない」
「規則が厳しい」「化粧がのらない」「結婚できない」
- スーパーエンジニア
「世情、人情、コンピュータ」に通じたスペシャルゼネラリストを育成



システムのことは
僕に任せてください！！

これがスーパーエンジニアだ！

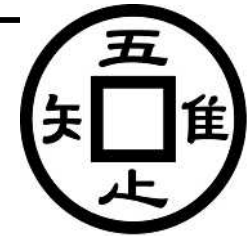
・年収:2000万円 ・趣味:サーフィン

・女性に優しい ・話題が豊富

写真はイメージです。

Universal Shell Programming Laboratory

23. 日本的アプローチと国際競争力



・技術を磨いて、システム(業務の仕組み)で競争する時代

- 日本は成熟した資本主義国家
- 単なる構造やフォーマットの違いで、企業が勝負する時代は終わった
- これからは、高度なクオリティと、細やかなサービスで優劣が決まる

**安い・早い・柔軟な技術が
国際的に評価される時代**

Universal Shell Programming Laboratory



usp lab.

ご清聴ありがとうございました。

ご質問は

tounaka@usp-lab.com

にお願いします。

Universal Shell Programming Laboratory