今日のお題
「最新のテクノロジー x SIビジネス」

セゾン情報システムズ常務取締役
CTO、アプレッソ代表取締役社長

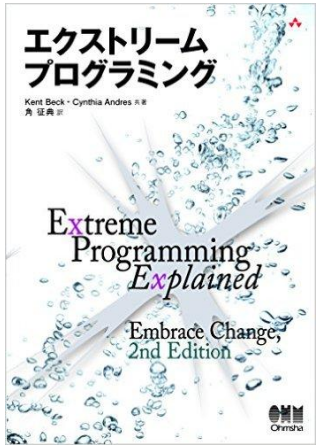
小野和俊

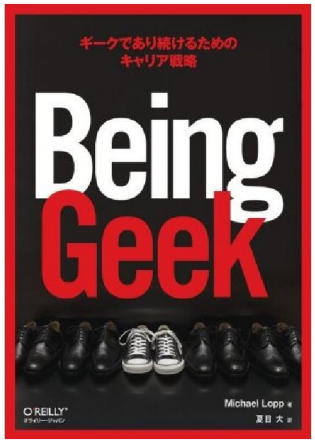
自己紹介

- 1976年 生まれ
  - 小学3年生から趣味でプログラミングを開始
- 1995-1999年 慶應義塾大学SFC 環境情報学部
  - NRIにてナレッジマネジメントシステムを企画・開発
- 1999-2000年 サン・マイクロシステムズ
  - シリコンバレーの米国本社にてJava + XMLを用いたサイジングアプリケーションを開発、E3 Award受賞
  - カリフォルニア州立大学サクラメント校大学院でのJava+XMLの授業を担当
- 2000年 アプレッソ起業
  - 「日本にエンジニアの楽園のような場所をつくりたい！」
  - 起業と同時にDataSpiderの企画・開発を開始、半年間で最初のバージョンが完成
  - DataSpiderは現在3年連続顧客満足度No.1、3000社を超える導入実績
- 2013年 セゾン情報システムズとアプレッソが資本業務提携
  - 2013 HULFT CTO → 2014 CTO → 2015 取締役CTO → 2016 常務取締役CTO
  - HULFTはMFT分野で世界2位にまで成長
  - 次世代SIerのあり方を模索し、自社で仮説検証

自己紹介

- 未踏ソフトウェア創造事業
  - PICSY(伝播貨幣) シミュレータークライアント設計＆実装(2002年)
  - Galapagos(XM会議ツール) 共同開発者 (2004年)
- 日経ソフトウェア
  - 巻頭連載「小野和俊のプログラマ独立独歩」(2007-2010年)
- 九州大学 工学部大学院
  - 「高度ICTリーダーシップ特論」非常勤講師 (2008-2011年)





SIerのデジタルトランスフォーメーションはなぜ難しいか？

1. 先行投資に対して及び腰になりがち
   - 「人月ビジネスなら確実に稼げるのになんでリスクのあることをやるのか？」
   - 「新しいことをやりたいようだが、ROIは？エビデンスは？第三者アセスメントは？」
   - 「そもそも新領域はやったことがない。そんなリスクのあることをやるのか？当社社員にできるのか？」
2. 現場のスキルトランスフォーメーションの難易度
   - 「COBOLerの俺らにブロックチェーンのシステム構築なんてできるの？」
   - 「なんだかんだでメインフレームはなくならない。案件だって溢れるほどある」
   - 急にFinTech担当、DX担当になる人同士の不毛な情報交換

SIerのデジタルトランスフォーメーションはなぜ難しいか？

3. BP依存体質
   - 「料理のレシピとソフトウェアの仕様書は似ている」
   - 単価の問題: プログラマーは早く卒業すべき低レベル職種
   - 技術的負債は、美しいコードは。
4. 技術力は証明しづらい
   - 資格を取る人 = 資格保有者としての最低限の能力を証明しているに過ぎない
   - PMP資格保有者 ≠ できるPM
5. レビュー文化、性悪説的アプローチ
   - モニタリング強化
   - 第三者アセスメント
   - IPAバグ密度

総じて・・・モード1→モード2への転換ができない。あるいは両立（バイモーダル)ができない

ユーザーとSIerをつなげる「美しいコード」の物語

DB Online

「美しいコード」という概念のない世界で

システム開発　ソフトウェア開発　SIer

小野 和俊[著] / 有馬三郎[著]

2017/06/05 06:00

「美しいコード」が大切なのは当たり前という考え方で生きてきた小野和俊さんが、4年前にセゾン情報システムズとの業務提携で目の当たりにすることになった、SIerの世界。そこではソースコードは特定の人の目にしか触れないような軽んじられた存在でした。SIerの世界に「美しいコード」の概念を定着させることはできるのか？セゾン情報システムズで小野さんと一緒に働く有馬さんとの対話をお届けします。

小野　今回、「美しいコード」をテーマに考えていこうということで、まず僕が興味を持ったきっかけについて話します。僕はこれまでパッケージをベンチャーでやってきた中で、プログラマーというのは美しいコードを目指すのが当たり前だとずっと思って生きてきました。ところが、4年前にセゾン情報と業務提携をして、そこでSIの実際の現場みたいなものに触れた時に、「美しいコードが当たり前」という考え方とは大きく異なる価値観で、いろんなことが動いている生態系を目の当たりにして。コードがあまり重視されてないというか、作業的に見られていることにショックを受けたわけです。これまで芸術性を極めて、大切にするべきもの、崇高なるソースコードへの探求の道みたいに思っていたものが、なんか雑作業

技術的負債
(Wikipediaより)

- 技術的負債（英: Technical debt）とは、行き当たりばったりなソフトウェアアーキテクチャと、余裕のないソフトウェア開発が引き起こす結果のことを指す新しい比喩である。「設計上の負債(design debt)」とも言う。
- 開発の中で先送りされるのは、文書化、テストコードの記述、ソースコード中の積み残し(TODO)項目の解決やコンパイラの警告、静的コード解析ツールの解析結果への対応などである。その他にも、技術的負債の例として、組織で共有されない知識や、複雑すぎて変更が難しいコードなどがある。

技術的負債
(Wikipediaより)

- 最初のコードを出荷することは、借金をしに行くのと同じである。小さな負債は、代価を得て即座に書き直す機会を得るまでの開発を加速する。危険なのは、借金が返済されなかった場合である。品質の良くないコードを使い続けることは、借金の利息としてとらえることができる。技術部門は、欠陥のある実装や、不完全なオブジェクト指向などによる借金を目の前にして、立ち尽くす羽目になる。

「ソフトウェアの仕様書は料理のレシピと似ている」(中島聡さんブログより)

- まず第一に「プログラムを書く」という仕事は簡単な仕事ではない。数学的な頭を持っていないとかなり辛いし、基礎がしっかりと出来ていないとろくなソフトウェアは作れない。
- これに関しては、自信を持って言えるのだが、「どんなに優秀なエンジニアでも、決してプログラムを自分自身で書かずに良い詳細仕様を作ることは出来ない」という絶対的な法則があるのだ。私の知っている優秀なエンジニアは、皆それを知っており自ら実行している。もちろん、彼らはプログラムを書き始める前に大まかな設計をするのだが、十分な経験を積んだエンジニアは、その段階でのものが「仮設計」でしかないことを良く知っている。だから、その段階で詳細設計書を書くような時間の無駄使いはせず、すぐにプログラム(もしくはプロトタイプ)の作成にかかるのである。

「ソフトウェアの仕様書は料理のレシピと似ている」(中島聡さんブログより)

- 実際にプログラムを書き始めて初めて見えてくること、思いつくことが沢山あるので、それを元に柔軟に設計を変更しながらプログラムを書き進めるのである。作っているプログラムが予定通りに動き始めてやっと、設計も完成に近づくのである。(ただし、そんな作り方で作ったプログラムはソースコードが汚くなってしまうケースが多いので、この段階から出来上がったプログラムを、読みやすさ・メンテナンスの高さを重視して大幅に書き直すことを強く薦める。エンジニアによっては、ここで一度作ったプログラムを全部捨ててしまってもう一度全部作り直す人もいるぐらい、この作業は重要だ。)
- 世界を又にかけてソフトウェア・ビジネスをしている米国の会社は、MicrosoftにしてもGoogleにしても、この法則にのっとって、アーキテクト自らがプログラムを書いている。

「ソフトウェアの仕様書は料理のレシピと似ている」(中島聡さんブログより)

- 私には、この「自分でプログラムを書かない上流のエンジニアが詳細設計書を作り、下流のエンジニアがコーディングをする」という工程そのものが、根本的に間違っているとしか思えないのだ。「下請け」という弱い立場にあり、経験も少ない下流のエンジニアが、「仕事を発注してくれる大切なお客様」である上流のエンジニアに対して、「この部分は、設計を少し変更をした方がプログラムがシンプルに書けるし実行効率もあがると思うんですが、変えちゃっていいでしょうか」などと言うことが出来るとは思えない。下流のソフトウェアハウスの経営者にとっても、そんな余計なことを言うエンジニアよりも、仕様書通りのプログラムを納期以内に黙って作るエンジニアの方が使いやすいのではないだろうか。
- 日本のエンタープライズ系のソフトウェア業界は、そんな根本的に間違ったソフトウェアの作り方を長年してきたために、まるで建築業界のような下請け・孫請け構造が出来てしまい、下流のエンジニア達が十分な経験も得ることが出来ずに低賃金でこき使われ、業界全体として国際競争力をなくしてしまう、という状況に陥ってしまったのではないか、というのが今回の私の仮説である。

「技術的負債」と「レシピの話」から学ぶべきこと

- プログラミングは数年で習得できるような下級職種的なスキルではなく、人によって何十倍、何百倍も差がでるような高度な技能である。
- プログラムを書ける人にしか良い設計書は書けない。
- したがって、自分でプログラムを書かないエンジニアが設計をし、下級エンジニアやBPが実装をする、という構造は根本的に間違っている。
- 複雑度の高いソフトウェア開発に於いては、最終的な設計は作り始めてみないと確定しない。
- BPさんと協力して仕事をしていくにせよ、自社で内製化し、技術的負債を背負わずにソフトウェア開発する力がなければプロジェクトは失敗する

技術力向上、とは？

- 資格を取れば技術力が上がる、というものでもない
- 新しい技術を使えばすべてが良くなる、というものでもない
- 採用すべき開発プロセス、取り入れるべき技術も、お客様やプロジェクトのメンバーの特性を考慮して決めていくべきもの
- パッケージ開発におけるやり方が常にSIにおいても正しいわけでもない

技術力向上、とは？

- 開発効率や生産性を高めたり、問題を早期に発見するためのツールは多数ある。しかし変化の速度が激しく、各事業部の各プロジェクトでこれらを個別に追いかけるのは効率的ではない
- 課題があるプロジェクトもあるが、最新技術をバランス良く使い、その結果、システムの品質も向上し、お客様も社員もとてもハッピーになっている、というプロジェクトもある
- ならば、先端技術を把握する役割を担うテクノベーションセンターに新しくチームを作り、事業部のプロジェクトに入って、プロジェクトがより良いものになるよう、各種支援を行うのが良いのでは？

→ モダン開発推進チーム発足

2016年10月
モダン開発推進チーム発足

With Flying Colors
～ 成功体験を広げていく ～

モダン開発推進チーム
～「いいな」と思えるやり方を広める～

- ITの世界はめまぐるしいスピードで変化しており、「効率的かつスピーディー、かつ楽しみながらシステムを作る」ための様々な工夫やツールが日々生まれている。そしてこうしたツールを有効活用することはプロジェクトの成功確率向上に大きく寄与する。
- こうしたツールや開発プロセス、設計や実装の技術によって得られる効果や感動を、モダン開発推進チームが現場に寄り添い、プロジェクトに一緒に参加してプロジェクトメンバーにそのすばらしさを体感してもらう。
- そしてファンになった人がインフルエンサーとしてそのやり方を社内に普及させていくことで、モダンな開発手法をねずみ算式に浸透させていく。

技術の変化

- HTML5を軸とするモダンWebアプリケーション
- ネイティブに近いUXを実現する“SPA”への移行
- APIベースのアーキテクチャが主流
- Webブラウザの進化
- OSSをはじめするフレームワークの進化
- DevOpsを支えるツールの進化
- CI/CDによる継続的品質改善の必要性
- DRY/CoC/DSL/Annotaionなどにより、スクラッチ開発でのコード量が大幅に減少
- クラウドを積極的に利用した開発の必要性
- デジタルトランスフォーメーション(デジタルマーケティング、クラウド、フィンテック、IoT)

モダン開発推進メンバーが
SIのプロジェクトに一緒に参加し、
次のような支援を行います。

支援内容

設計支援

- アーキテクチャ構築、フレームワーク選定等

開発環境

- ソースコード管理、CI / CD、BTS、テスト自動化等

実装支援

- デザインパターン、コードレビュー、ライブラリ選定、技術トラブル相談等

モダン開発推進支援プロジェクト
従来型の開発手法との比較
(流通系、2016年10月～12月)

ソースコード管理

- 従来
  - ST前に一斉コミット
- 今回
  - 随時(デイリー)
    - 作業の定量的な把握が可能に
    - デイリーコミットが習慣化することで**個人で抱え込まずにチーム全体で共有することができる**ようになり、**問題の早期発見につながる**

      ※　支援初期段階では**「自信のなさ」や「恥ずかしさ」**からデイリーコミットが進まなかったが、共同所有の有用性を体験し、理解してもらうことで最終的には習慣化することができた

脆弱性診断 / 静的解析

- 従来
  - ST完了時に実施
- 今回
  - 随時(デイリー)
    - **コミット単位での脆弱性診断が可能**なため、修正に伴う戻り効果は必要最低限に抑えることができた。
    - 脆弱性診断の他、各種静的解析ツールも同様であり、**段階的且つ確実な品質の作り込みが可能となった**。

テストコード

- 従来
  - 作成していなかった
- 今回
  - **実装箇所の優先順位にもとづき、機械的に実行可能な単体テストコードを実装**。
  - バグ修正やリファクタリングの際に既存コードの品質が担保可能となった。

コミュニケーション

- 従来
  - メール・電話による情報伝達
- 今回
  - Slackを活用したコミュニケーション
  - メールに比べてリアルタイム且つライトなツールのため、コミュニケーションのオーバーヘッドが削減。冗長になりがちなメールに対して、問題の具体的な内容をシンプルに議論しやすかった。
  - オープンなコミュニケーションツールのため状況が全員で共有可能。**課題発生時に有識者を交えて議論し、短時間でのトラブルシューティングにつなげることができた**。
  - 各自のコミット状況やジョブの実行結果などを自動的に通知。**出社時にチャンネルを確認するだけで、前日のテスト状況の把握と必要に応じて修正を素早く行うことができた**。結果として問題発生時の対応スピードが向上した。

考察

- モダン開発推進として下期から活動を開始し、3ヶ月間で第一弾の支援が完了。
- モダンな開発手法の導入により、**総じて「問題の早期発見」が行いやすくなる**ことが改めて確認できた。とりわけ、**これまでの定例会議では発見できなかったような問題が自然に見つかる**ようになることに大きな効果があると思われる。例えば今回のプロジェクトでもコミットがしばらく行われていない、静的解析ツールの解析結果が数日経っても改善に向かっていない等の兆候がデジタルアラートとして機能した。(定例会議ではこれらをひとりで抱え込んでいる場合にそれが見えない)
- チーム内のカルチャーが**「一人ではなくチームで開発」**という文化に自然に引き寄せられていくため、できていない、どうしよう、悪い報告はできない、そして報告が遅れて更に・・・というネガティブスパイラルに陥りづらい。
- セゾン情報のSI事業はかなりレガシーな開発手法で行われており、したがって改善に向けた伸び代が大きいとも言える。

PMジェダイ評議会

- 再発防止に向けた「PM力強化」のための最重要施策として、実践力重視で招集された「PMジェダイ」で構成されるPMジェダイ評議会を発足する。要注意プロジェクトに対してPMジェダイを派遣し、課題の整理と解決に当たる。

そうだ、開発合宿をしよう！

文化の問題

SoRとSoE

- 「キャズム」で知られるジェフリー・ムーア氏が2011年から提唱している考え方

SoRとSoE

- SoR
  - System of Record
  - 事実を記録するためのシステム
  - 例. ERP、SCM
- SoE
  - System of Engagement
  - 絆をつくるためのシステム
  - 例. CRM、MA

バイモーダルとは

- バイモーダル
  - バイモーダル = ふたつの流儀
  - ユニモーダル = ひとつの流儀

- バイモーダルの「ふたつの流儀」
  - モード1
  - モード2

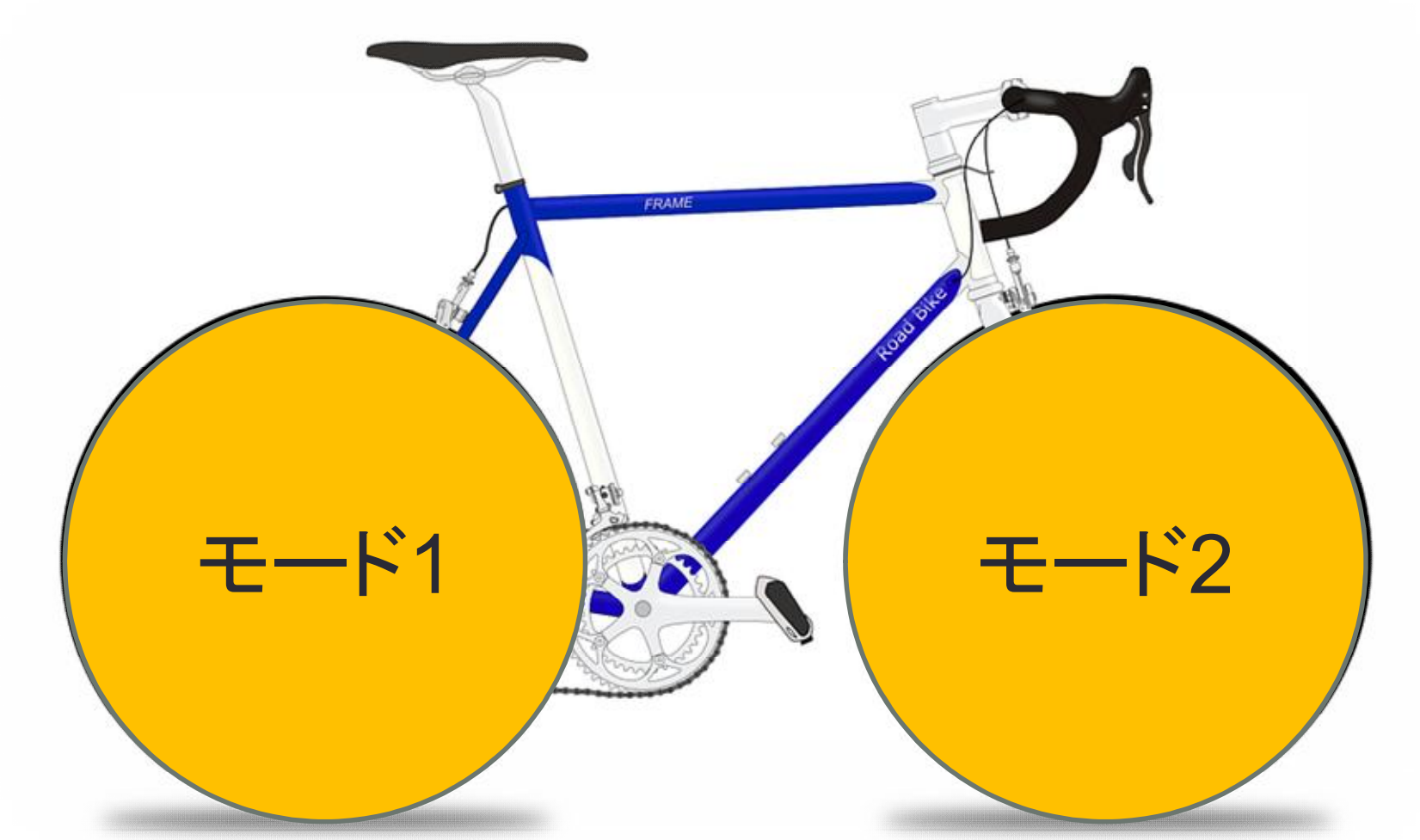
–	モード1	モード2
タイミング	事後的	事前的
性向	安定性重視	速度重視
開発手法	ウォーターフォール	アジャイル
アプリ例	ERP,SCM	CRM,MA
管理部門	IT部門が集中管理	ユーザー部門が分散管理
対象業務	予測可能業務	探索型業務
例えるなら	武士:領地や報酬を死守	忍者: 何が有効なのかを探る
誰のためのもの	運用者(オペレーター)	革新者(イノベーター)
重視すること	効率性、ROI	新規性、大きなリターン
車の運転で言うと	リスクを抑えて安全運転	スピード重視で運転
経営	トップダウン	ボトムアップ
規模	大規模	小規模
強み	統率力、実行力	機動力、柔軟性

一部、ZDNet「経済のデジタル化がもたらす企業ITの“バイモーダル”が目指すもの」から引用
http://japan.zdnet.com/article/35075658/

バイモーダルな組織

- 「バイモーダル」が意味するもの
  - モード1とモード2のどちらが良い、という話ではない
  - 両方を兼ね備える組織が強い
- 守護者(ガーディアン)
  - モード1とモード2の調整役
- ガートナーの予測
  - 「2017年までにIT組織の75%がバイモーダルの能力を備えるようになる。そのうち半数の組織で混乱が生じる。その原因は文化的な問題に対処しないこと。」
  - 「二重人格的要素を取り入れることが必要。」

自転車(Bicycle)とバイモーダル(Bimodal)の類似性



自転車(Bicycle)とバイモーダル(Bimodal)の類似性

- 両方が存在することで安定する
- 一方(前輪、モード2)が方向を変えながら進むべき方向性を模索する
- もう一方(後輪、モード1)は方向性が決まったら、前輪に少し遅れて、自転車や事業をまっすぐに力強く進める

「バイモーダルIT」への道

最初の変化はHULFTから起こった

- HULFTの一般的なイメージ
  - 「メインフレームやUnixのための連携ソフト」
- とするとHULFTの将来は・・・？
  - メインフレームやUnixの相対的位置が下がっていくと、一緒に落ちていくソフトウェア・・・？

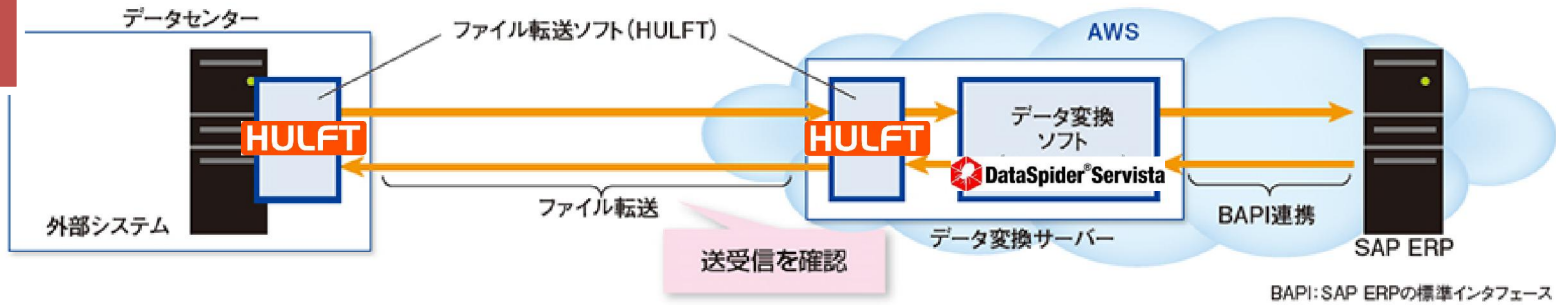
- しかしHULFTはむしろ今まで以上に成長し始めた。それはなぜか？
  - 「自らを食い殺す領域を、逆に取りに行く」
  - マイクロソフトやJTの事例
  - 「痛みのある箇所 = ペインポイント」を探す

AWS Leadership Award
「Think Big」賞を受賞

クラウドでのHULFT活用パターン

活用例　：　クラウドとオンプレミスのI/FにHULFTを利用

ケンコーコム様 事例



出典：http://itpro.nikkeibp.co.jp/article/Active/20131118/518729/

インテージ様 事例

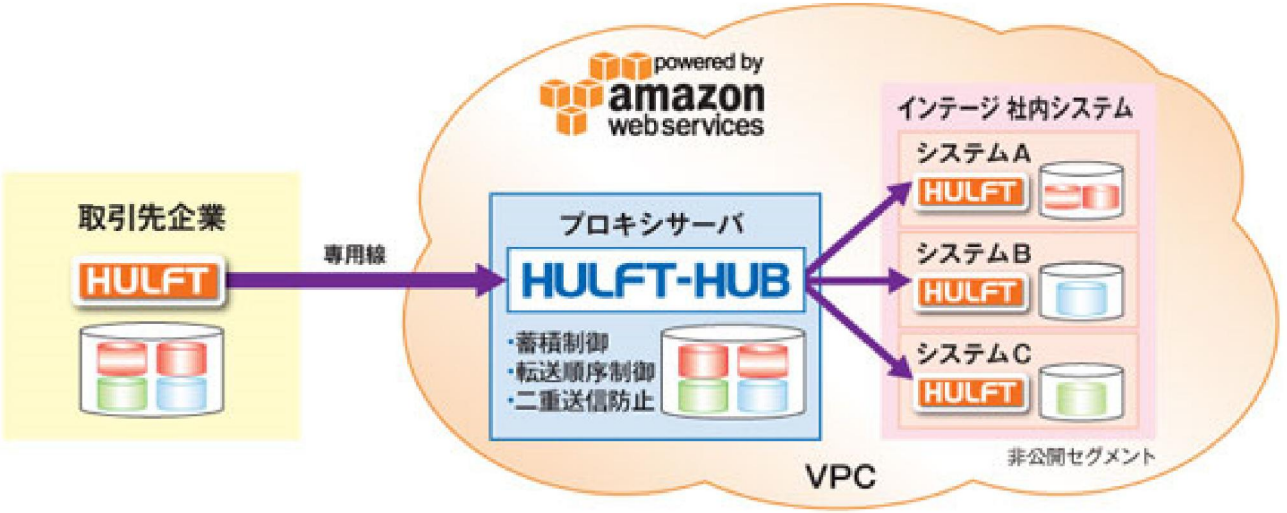


図　トータルリサーチシステムの概要

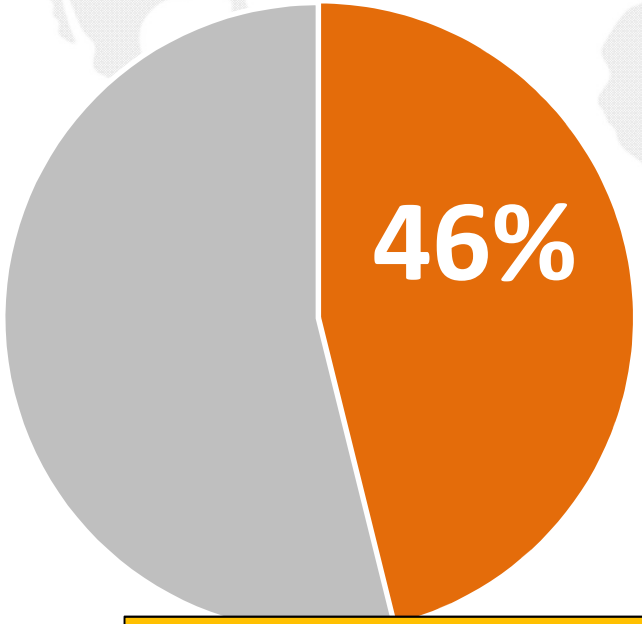
AWS上にプロキシサーバとして機能するHULFT-HUBを活用した中継サーバを構築。転送データはHULFT-HUBに蓄積され、転送開始の順序通りに送信する追越禁止の制御や、受信側サーバに障害が発生しても自動的に再送される機能がある。顧客システムとAWSは専用線でつながっている。

世界でも選ばれるHULFT

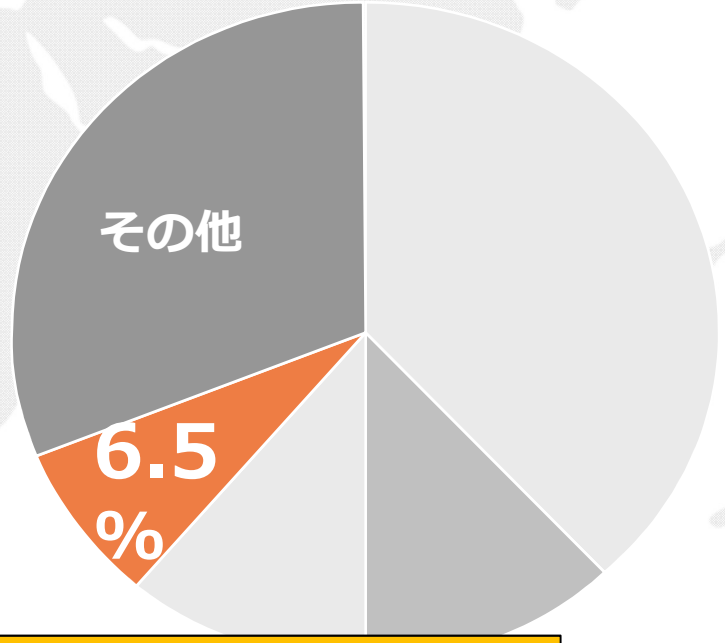
アジアシェア **No.1**

メガバンクをはじめ主要企業に選ばれています。

アジアシェア第1位



世界シェア第４位



2017年夏、更にシェアを伸ばし世界2位に！！

出典：IDC調べ

ミッションクリティカル領域 で 20年以上 の経験

全国銀行協会 会員銀行

導入率 **100%**

一般社団法人全国銀行協会ホームページ掲載の全銀協会員（正会員）に基づく

日本自動車工業会 会員企業

導入率 **100%**

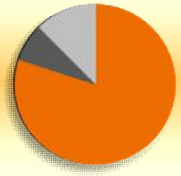
一般社団法人日本自動車工業会ホームページ掲載の会員に基づく

導入実績

8,400社

181,000本

2016年3月末現在



日本国内 市場 シェア

12年連続 **第1位**

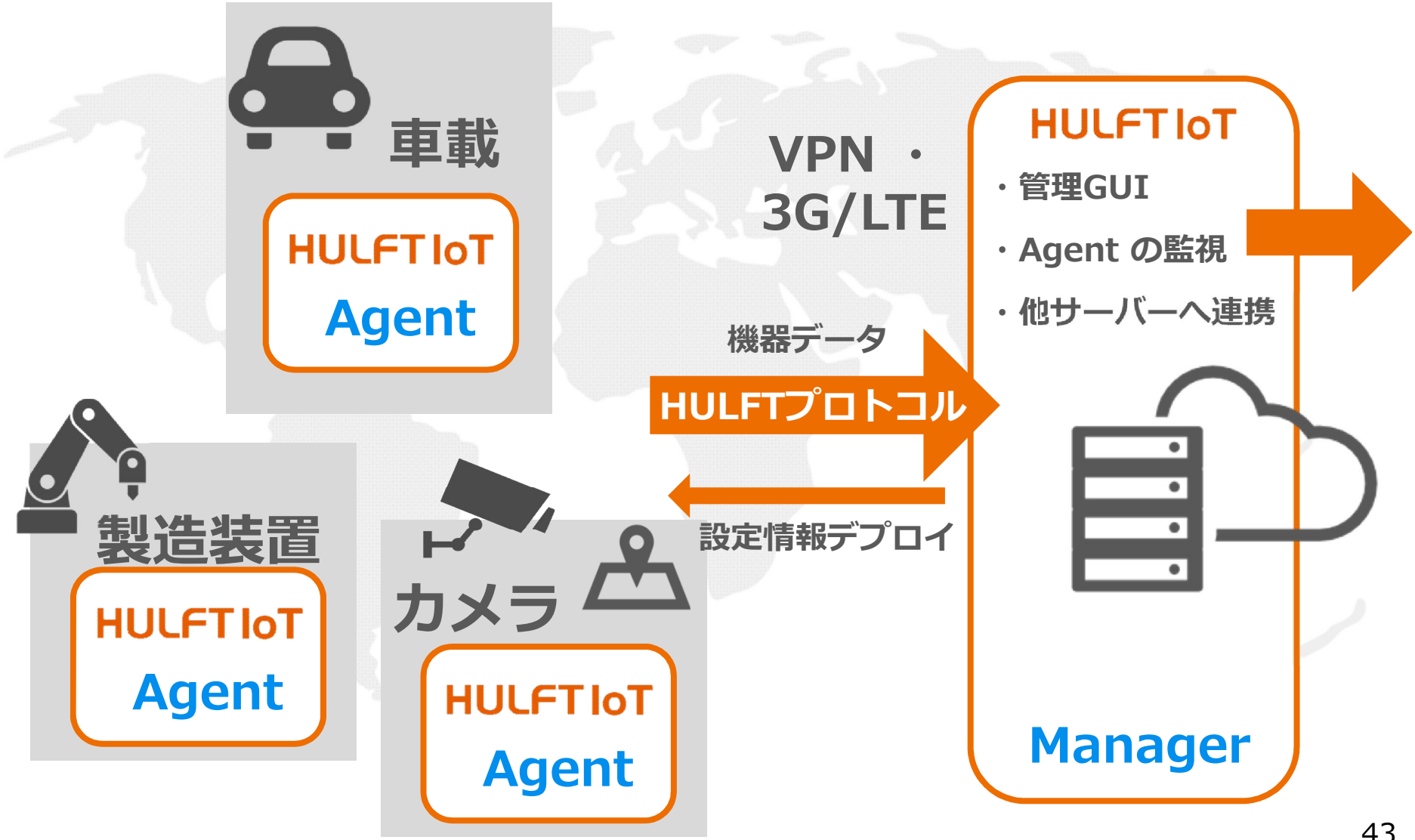
市場占有率 **77%**

富士キメラ総研 「ソフトウェアビジネス新市場 2015年度版」

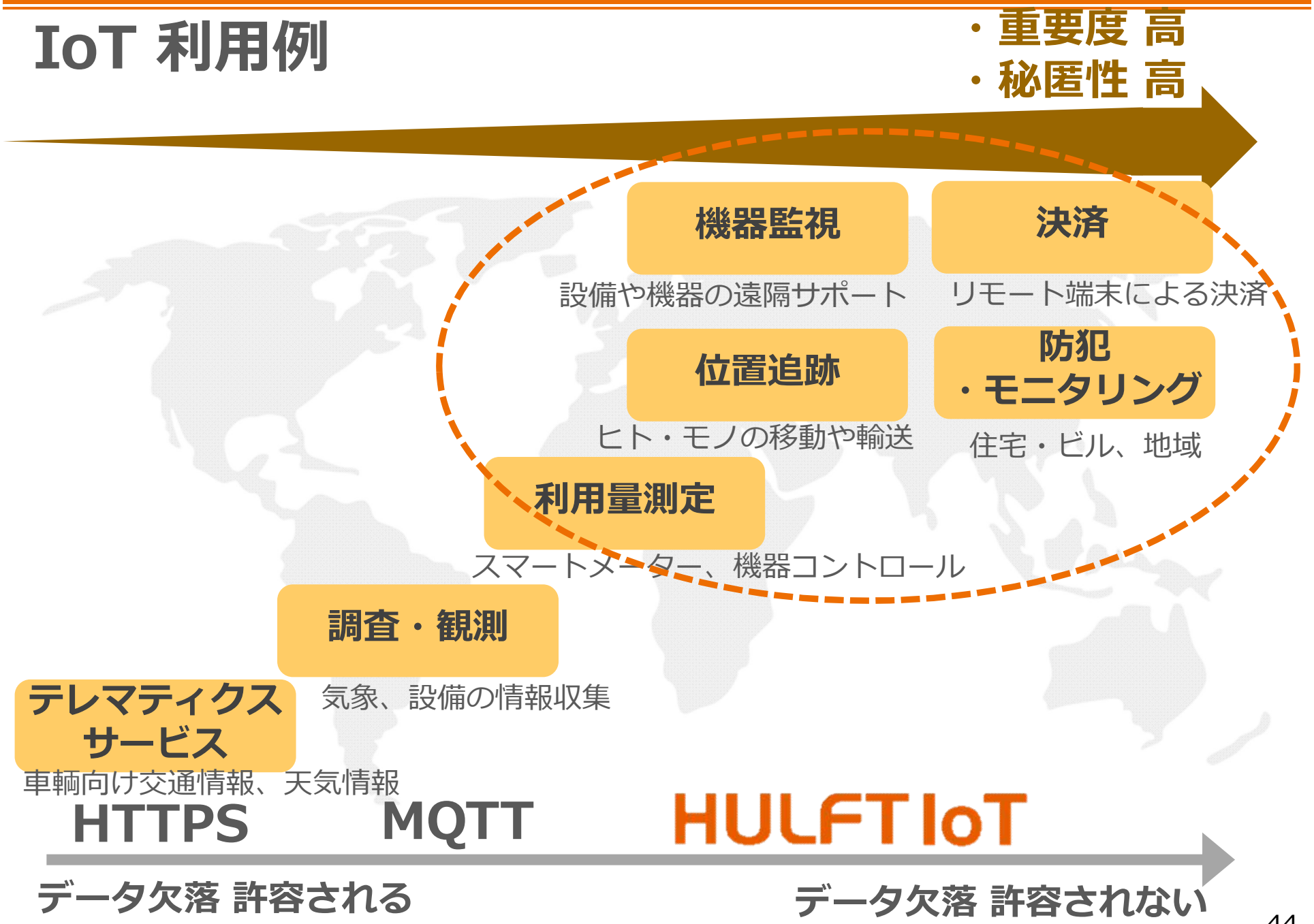
HULFT IoT

HULFT IoT 構成イメージ

HULFT IoT 利用条件
- **IPアドレス** を持つ
- **OS (Linux/ Windows)** がある
- **ファイル** になっている

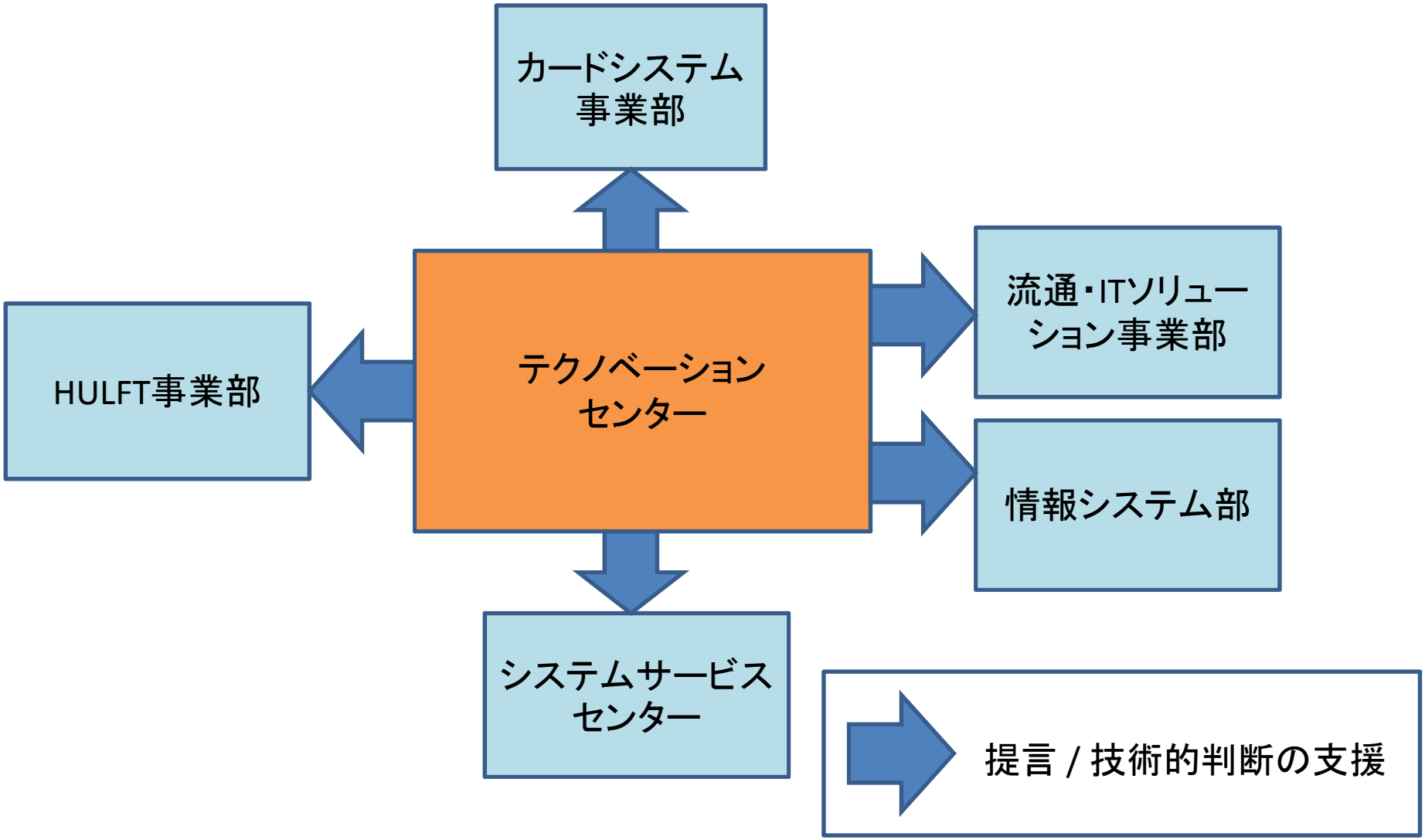


IoT 利用例



テクノベーションセンター設立

- HULFTで起こったようなイノベーションを、SI部門も含めた全社に展開できないか？
- テクノベーションセンター設立
  - テクノベーション = テクノロジー + イノベーションの造語
  - 先端技術の調査研究
  - 事業戦略、技術戦略の企画立案
  - 業界のキーマンとのコネクション提供
  - POCを企画・率先
  - 風通しの良い企業文化の醸成
- ミッション
  1. セゾン情報システムズとその顧客のイノベーションを加速・誘発する。
  2. セゾン情報システムズの技術力向上。


カードシステム
事業部
HULFT事業部
テクノベーション
センター
流通・ITソリューション事業部
情報システム部
システムサービス
センター
提言 / 技術的判断の支援

テクノベーションセンター担当領域

- 先端技術系
  - AI
  - クラウド
  - IoT
  - ブロックチェーン
  - コミュニケーションインフラ (Slack etc.)
- モダン開発推進
  - 開発環境/ツール (Jenkins, GitHub, JIRA, etc.)
  - 開発プロセス (アジャイル / スクラム / WFとのハイブリッド開発)
  - プロジェクト支援
  - ソースコードリファクタリング支援

「ラストマン」の育成

- 「ラストマン戦略」は小野自身が実践してきたスキルアップ戦略で、アプレッソは16年間この仕組みで高い技術力を持つ人材を生み出し続けている
  http://blog.livedoor.jp/lalha/archives/50140618.html
- ４月からテクノベーションセンターもできたので、セゾン情報にもこの仕組みを導入

Uhuru IoT Partner Community WG6

宅配BOX × IoT × ブロックチェーン

SAISON INFORMATION SYSTEMS CO.,LTD.

◆実証実験構成

【ユーザ用デバイス】

リキッドデザインを採用し
HTML/Javascriptを利用した
iOS、Androidの両方に
対応できる宅配業者、利用者
専用アプリを開発

【宅配BOX】

エスキュービズム社
スマート宅配BOX®
を利用

HULFT IoT Agent

**宅配BOX上の
RaspberryPiに
HULFT IoT Agent
を設定**

HULFT IoT Manager

AWS

**HULFT IoT Managerが
宅配BOXの稼働状態を監視
データはクラウドに集約**

【ブロックチェーン】

GMOインターネット社が提供する
PaaS型のブロックチェーンプラットフォーム「Z.com Cloud ブロックチェーン」を基盤にシステムを構築

Uhuru IoT Partner Community WG6

- 事業部長懇親会での蕎麦屋での会話がトリガーになり、WG6プロジェクトでの事業部を超えた連携が実現。
- **新しい技術に触れる楽しさ、自分たちで手を動かして作ることの喜び、他社と交流することで生まれる刺激、「自分たちにも先進的なシステムが作れる」という自信**、様々な点で社員の成長につながった。
- 株価もこの発表を受けて1日で10%上昇(翌日のJIG-SAWとのプレスリリースで更に上昇)、**市場からの期待**も感じさた。
- 他のWGでのオイシックスとの連携も含め、具体的な商談の話も出てきており**案件創出可能性も出てきた**。

新領域への取り組み事例

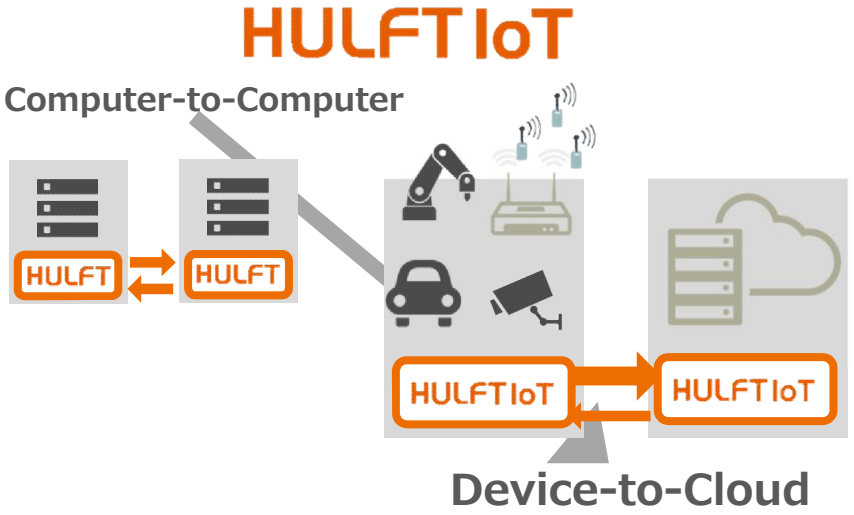
流通×IoT

1）ウフル社主催IoTパートナーコミュニティへの参画
2）HULFT IoTリリース
3）流通PoCの実施

＜IoTパートナーコミュニティへの参画＞

IoT Partner Community

＜店舗でのPoCイメージ＞



※来店者人数・属性／実購買者の情報／店内温度の
データ取得による見える化

本社移転（11/20～）

住所：107-0052
東京都港区赤坂1-8-1　赤坂インターシティAIR 19F
電話番号：03（6370）2000（代）
アクセス：銀座線/南北線「溜池山王駅」直結
　　千代田線/丸ノ内線「国会議事堂駅」直結
　　日比谷線「神谷町駅」(徒歩10分)

社内カフェ(コーヒー10種類＆BitCoin Coreベースの社内仮想通貨で決済可能！)

自由闊達で風通しの良い企業風土の醸成

- クロスファンクションプロジェクト
  - ×「隣の部署は別の会社」
  - 複数事業部をまたぐ事業企画のディスカッション
  - 「English Lunch」などの事業部を越えた交流を推奨
- TGIF開催
  - 各事業所に回って参加自由のビア・バストを開催
  - 希望者によるLT
  - その他各種イベント
- Slack運用開始
  - この効果が非常に大きかった


slack
One year in...
1 Year Anniversary!
February 12, 2015
500k
400k
300k
200k
100k
Over 500,000
Daily Active Users
Official Launch
February 12, 2014
Start of this Graph
August 12, 2013
Holidays!
The number of Daily Active Users has doubled twice in the past six months

Slack運用状況

- 導入後1年経ったの現在
  - 700名の社員のうち515名が登録、アクティブユーザー316名
  - 121のパブリックチャネル
  - 1日500-900のメッセージ
  - プロジェクトの現場でも主たるコミュニケーション手段がメールからSlackに移行してきている
- 利点
  - 堅苦しくなく気軽に聞ける
  - 知っている人や意見がある人がいればすぐ答えてくれる
  - 組織や役職を気にせず自由な意見交換ができる
  - 伝えたい内容が前面に出る


ギークであり続けるための
キャリア戦略
Being
Geek
Googleのギークたちは
いかにしてチームを作るのか
Team
Geek
本書はキャリア形成や現在の職場に関する悩みを抱えるエンジニアが
自分自身の状況に対する客観的な視点を得ることを手助けしてくれる書籍であると言えるだろう。
小野和俊（アプレッソ 代表取締役副社長 CTO）
「自分は次にどうすべきか」について考えるヒントに満ちている。
よしおかひろたか
技術・マネージメント・キャリア・転職、漠然と思っていたことが、
この本を通してかみ砕いて理解することができ、
エンジニアとしての生き方についてじっくり考えさせられた。
舘野祐一（クックパッド）
「『Being Geek』への推薦の言葉」より
Ohmsha
O'REILLY®

「HRTの原則」

- HRTの原則: 優れた開発チームでは、次の3つの価値観が大切にされている。
  1. 謙虚さ (Humility)
  2. 尊敬 (Respect)
  3. 信頼 (Trust)

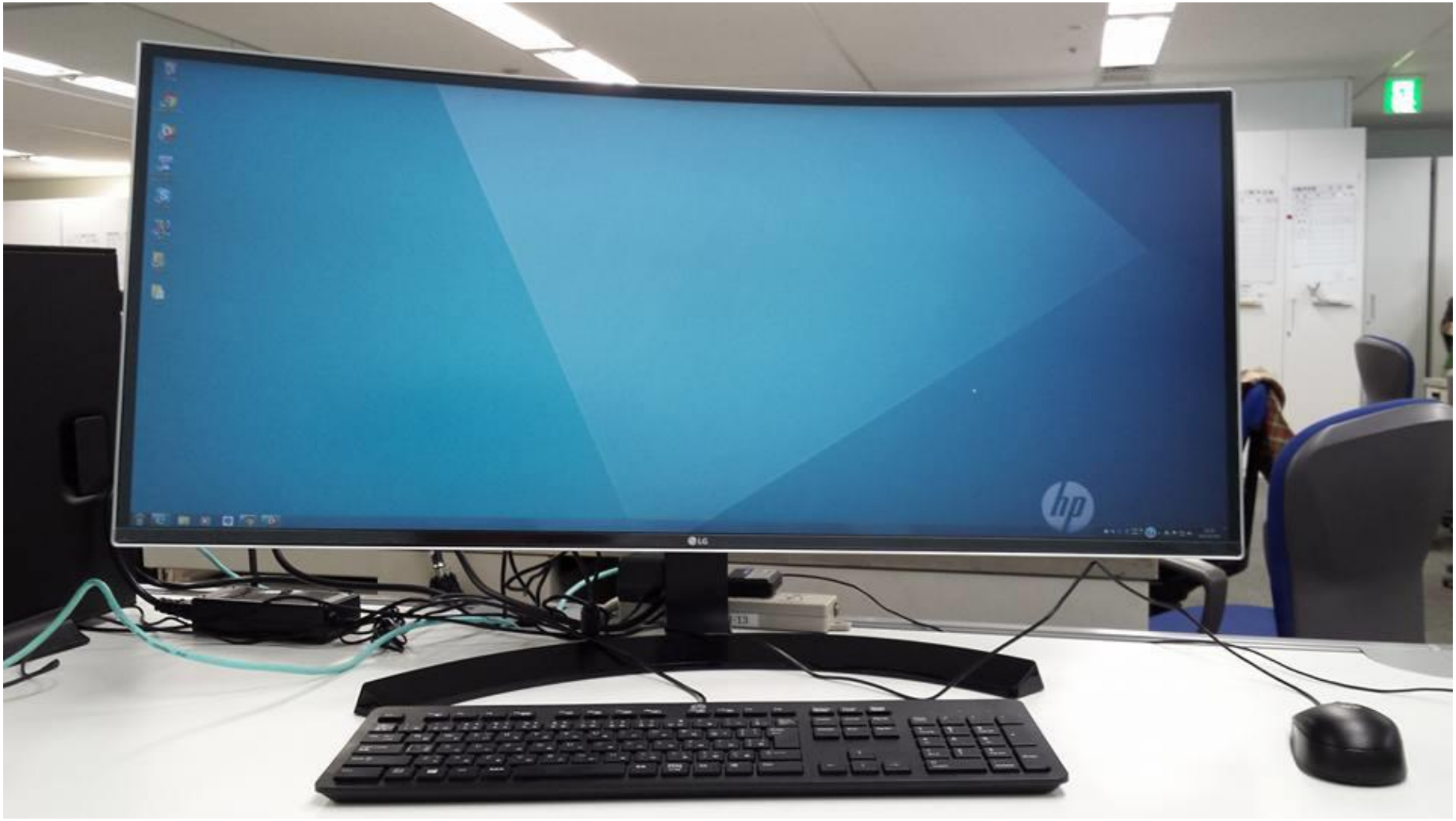
- 「あらゆる人間関係の衝突は、謙虚・尊敬・信頼の欠如によるものだ」
- 「プログラマとして成功するには、最新の言語を覚えたり高速なコードを書いたりするだけではいけない。プログラマは常にチームで仕事をする。君が思っている以上に、チームは個人の生産性や幸福に直接影響するのである。」

AI時代の9つの基本原則

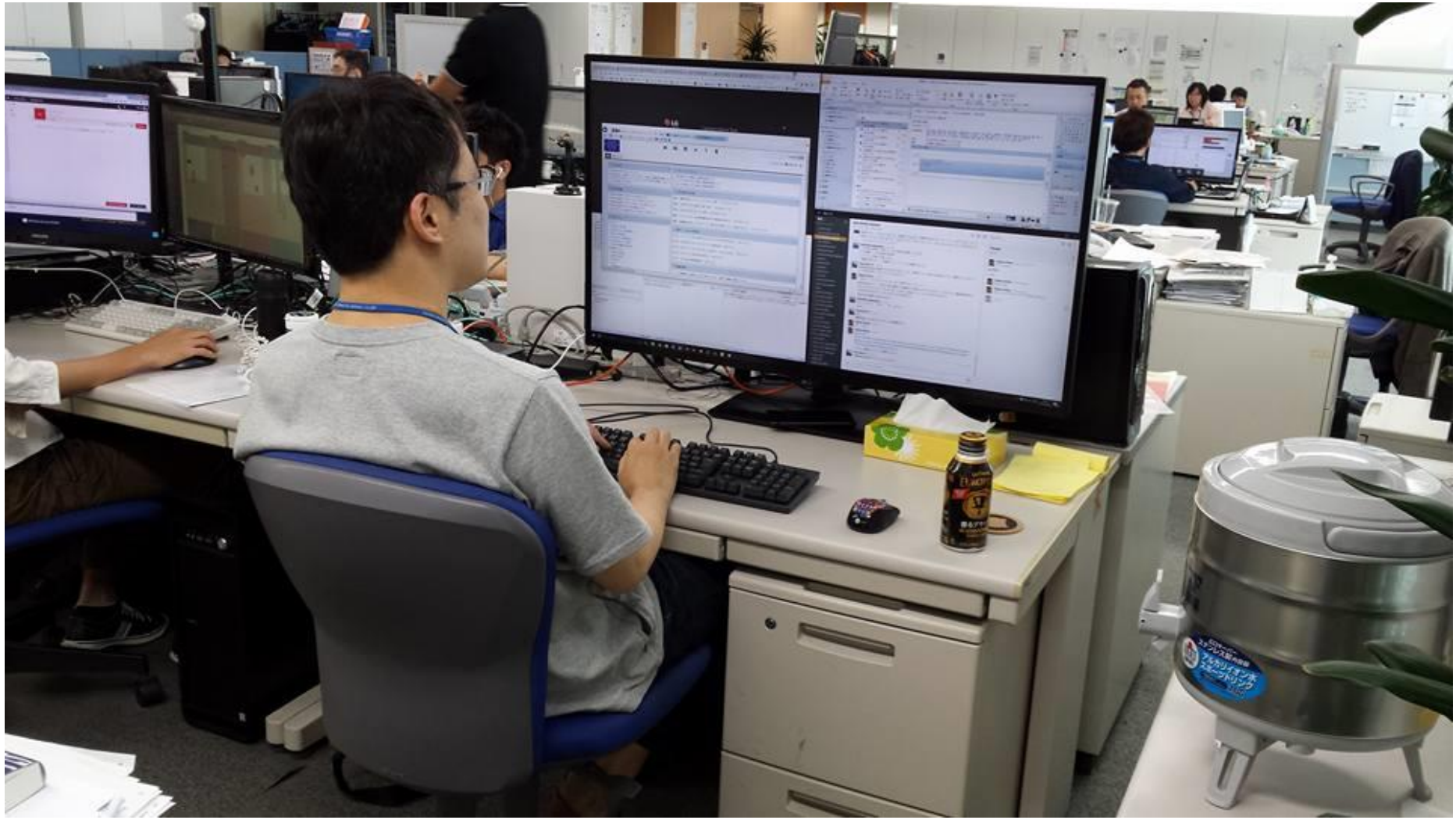
- Compass over maps
  - 複雑かつスピードの速い世界では、すぐに書き換わってしまう地図を持つよりも、優れたコンパスを持つことが大切です。強い企業はすでにコンパスを持ち、現場で素早く物事を決めていますが、多くの日本企業はいまだに地図を作るために膨大な時間とコストをかけています。見通せない未来という地図を懸命に描こうとしているのです。
- Practice over theory
  - 理論的には不可能なことでも、なぜか可能になってしまうことがある。それがAIの時代です。だからこそ、とりあえず動いてみて、プロセスの中で仮説・検証を繰り返していく。大切なのは理論を知っていることではなく、実際に機能することなのです。

引用元:「逸脱からはじまる『学び』の実践」
http://www.academyhills.com/note/opinion/13071901mitjoi.html

曲面ディスプレイ実験導入



4Kディスプレイ実験導入



さいごに

- システム開発によろこびと驚きの連鎖を
- “Fun Place to Work”