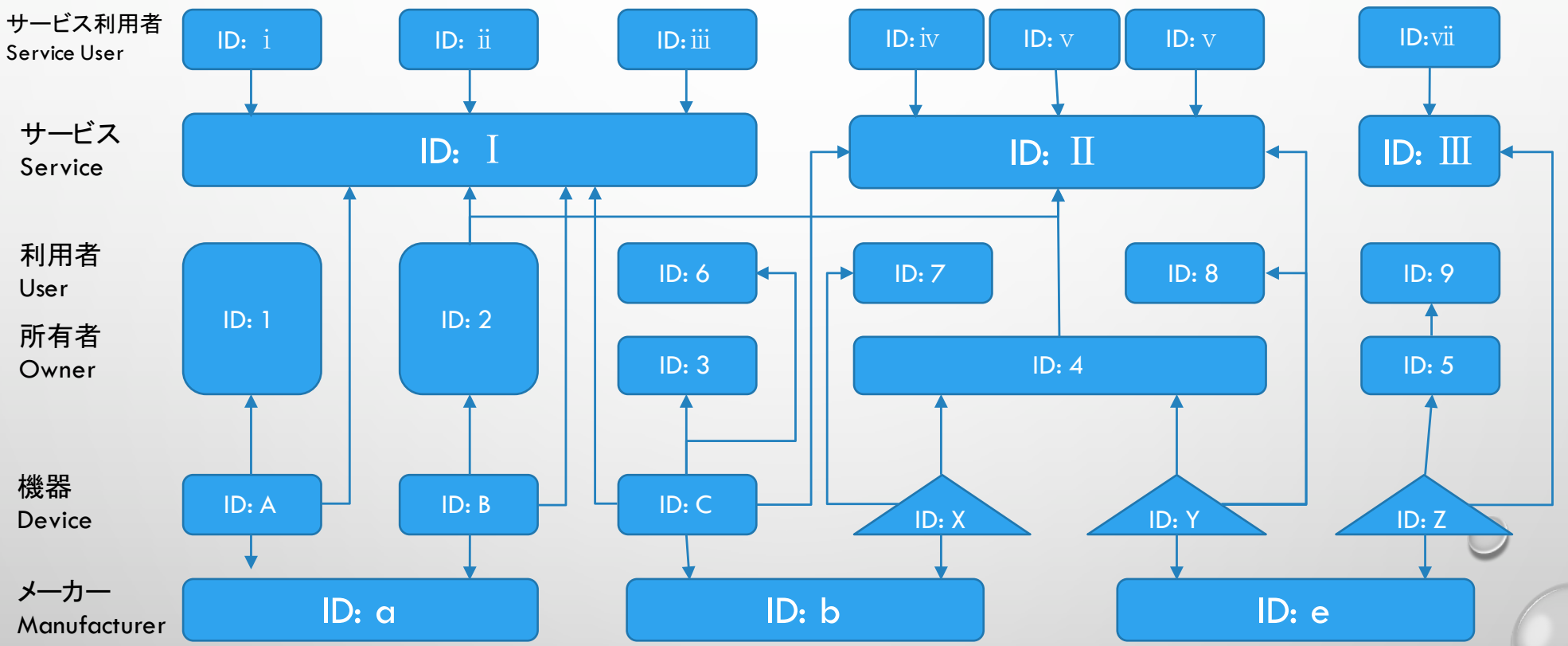
# 二木　真明　Masaaki Futagi, CISSP

- 独立セキュリティコンサルタント　IT Security Consultant (Independent)
- 日本クラウドセキュリティアライアンス　IoT WGリーダー
- Leader, IoT Working Group , Cloud Security Alliance Japan Chapter Inc.
- NPO 日本ネットワークセキュリティ協会(JNSA)　幹事
- Secretary, NPO Japan Network Security Association (JNSA)
- 経歴　（Experience）
  - ８０年代いにしえの組み込み系プログラマ
  - ‘80s Embedded System Programmer (Old fashioned )
  - ９０年代のUNIXシステム・アプリケーションプログラマ
  - ‘90s UNIX System and Application Programmer
  - ２０００年代のセキュリティエンジニア
  - ‘00s Security Engineer
  - ２０１０年代のセキュリティコンサルタント
  - ‘10s Security Consultant

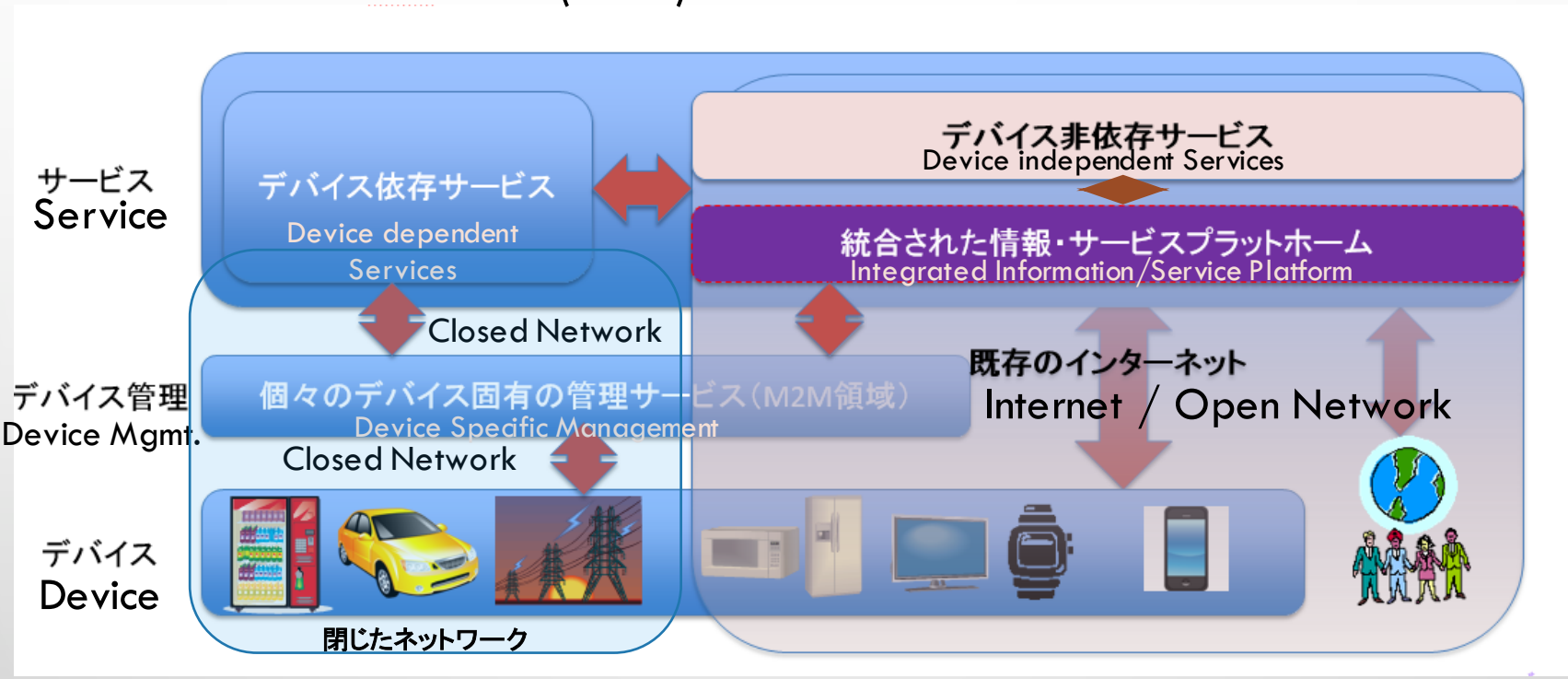
# たとえばアイデンティティ管理

Identity Management of Internet of Things



# 近未来の IoT（WoT?）ワールド

IoT (WoT?) world in near future




API連携

From: 'Identity of Things', Masaaki Futagi
ID and IT Mangement Conference 2015

# 雲の上から見たIoT

Bird's View of 'Things' from/over the Cloud

- IoT は単なるプロダクト（デバイス）のネットワークではなく、複雑なシステムもしくはサプライチェインである
- 'IoT' is not simply a network of product or devices but a complexed system and/or supply chain.
- IoTはもはや「閉じた世界」ではなく、際限なく「繋がって」いく
- 'Internet' of Things (IoT) is no longer a 'CLOSED' world but unlimitedly expanding "CONNECTED' world.
- 様々な接続における「想定外」を排除出来るかが、セキュリティ屋の心配事
- Whether we can or can not to eliminate errors in connected environment beyond our expectation. It is one of concern of 'SECURITY' practitioners.

# セキュリティ V.S 品質

Security V.S Quality

- 製品やサービスの セキュリティ ∈ 品質

  Security of Product/Service is a part of 'Quality'

- 脆弱性 ＝ バグ（その影響で認証やアクセス制御・・などに悪影響があるもの）

Vulnerabilitys are 'bug's affect to, and weaken or bypass Authentication, Access Control…

- 設計時にセキュリティ機構を組み込む→正しく機能するかテスト

  Considering Security features into Design and test them properly

- 製作・実装時のバグ潰し→脆弱性の排除
  - 特にAPIに対する異常な取扱（呼び出し手順、パラメータ・・）に対する耐性のチェックが重要 （不可解なエラー、誤動作、停止はダークサイドにつけ込まれる）

Eliminating 'bug's also eliminates 'Vulnerabilities'.

Especially, Checking tolerance for abnormal handling ( calling, irregular parameters …) of APIs is crucial. (Unexpected error, Malfunction.. summons 'Darkside' to the system.

# テストというよりも設計の問題かもしれないが

Perhaps, it is not a 'TEST' issue but 'DESIGN' issue …. ?

- デバイス、サービス、データのすべてにおいて最初の想定どおりの使い方（使われ方）が続くとは限らない
- Usage of 'device', 'service', 'data', all aspect of system may be changing unexpectedly.
- リスクの高い使われ方を防ぐ「サーキットブレーカー」が必要になるかもしれない（そしてそのテストも）
- A kind of 'Circuit Breaker' may be necessary to prevent 'risky' usage. (and test of it)

# どこまでやれば・・・・

How deeply or completely should we test them??

- リスクに応じて・・・・とセキュリティ屋は言うけれど・・・

  Depends on ‘Risks’, Security Practitioners say ….. but ….

- いったい「リスク」って何？？ 誰の「リスク」？

  What are ‘Risks’ , and to whom ??

# システムが与える「影響」を認識する

- いくつかの切り口　　Some aspects of 'Impact'
  - 【デバイスの特性】 デバイス単体が利用者や環境に与える影響

    [Device] Impact that depends on device feature.
  - 【サービスの特性】 デバイスをたばねるサービスが与える影響

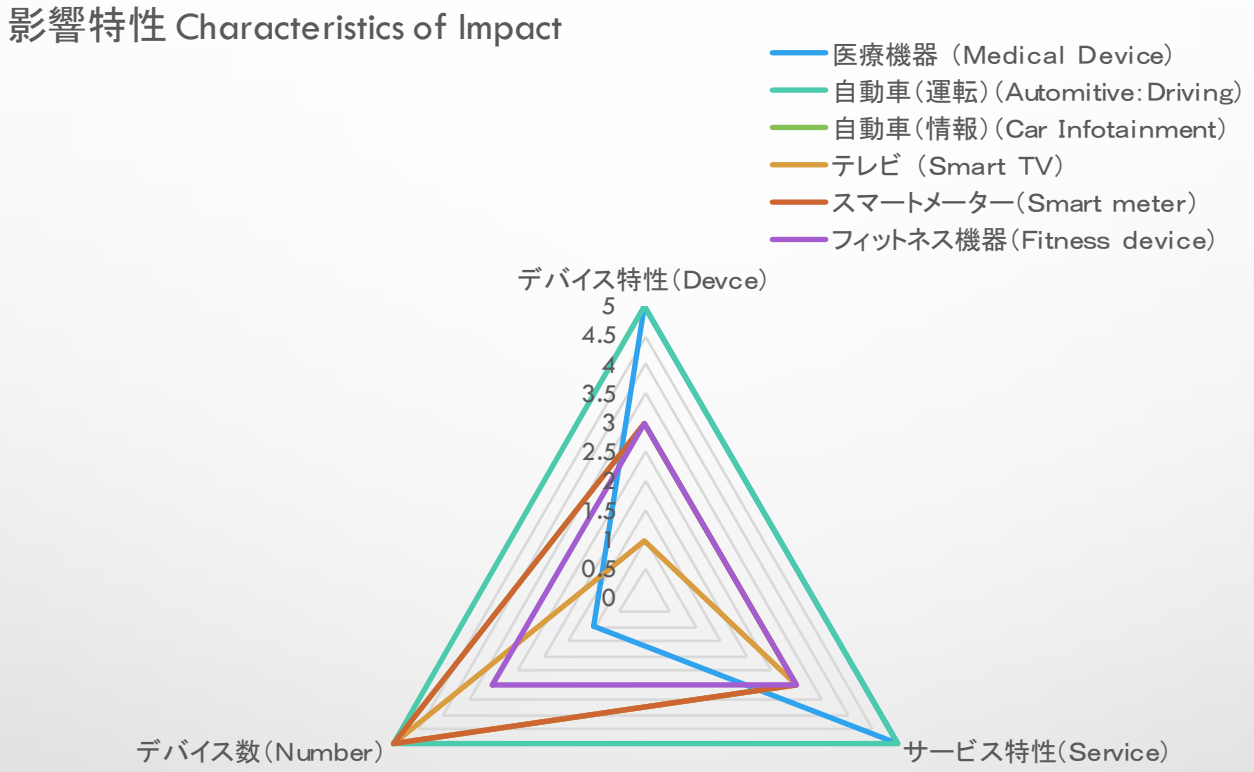
    [Service] Impact that depends on Service feature.
  - 【デバイスの数（規模）】 システムが環境や社会に与える影響

    [Scale] Impact that depands on number of devices.
    (Impact to society, environment ..)

最初の一歩・・・・・ではあるが・・・・

# 影響特性比較例

Example of impact visualization.




| | デバイス特性(Devce) | サービス特性(Service) | デバイス数(Number) |
|---|---|---|---|
| 医療機器 (Medical Device) | 5 | 5 | 1 |
| 自動車(運転)(Automitive:Driving) | 5 | 5 | 5 |
| 自動車(情報)(Car Infotainment) | 3 | 3 | 5 |
| テレビ (Smart TV) | 1 | 3 | 5 |
| スマートメーター(Smart meter) | 3 | 3 | 5 |
| フィットネス機器(Fitness device) | 3 | 3 | 4 |

＊注：厳密な評価結果ではありません ： Those scores are just for example, not well considered.

# テスト負担の軽減

- 標準化とフレームワークの確立
- Standardization, Establishing Frameworks
- 通信プロトコル、セキュリティ機能（認証、ID管理、アクセス制御・・・）REST APIのハンドリング・・・・をサポートする開発用フレームワーク、ライブラリの必要性
- Development Framework which supports Communication Protocols, Security functions, such as Authentication, ID management, Access Contol ..etc, REST API handling .. Is necessory.
- 共通機能について、よくテストされたフレームワークが利用できれば、個々の開発者の負荷は軽減できるはず
- If well tested frameworks for common functions available, it may mitigate work load of testing.