PFS制御ソフトウェア理想と現実

Atsushi Shimono
Univ. of Tokyo; Kavli IPMU
PFS Project Office

お題

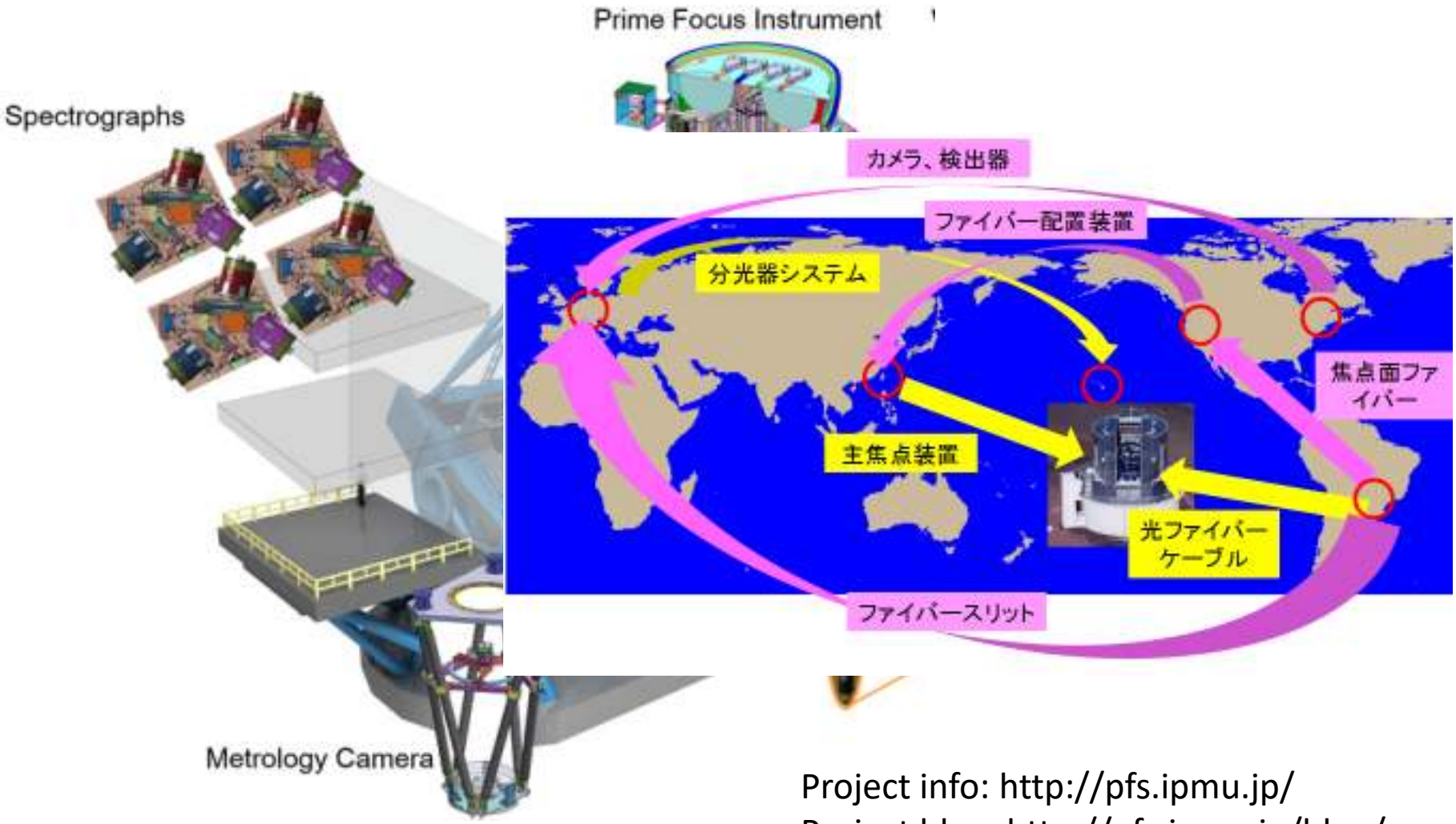
「PFS に関連して装置制御に関わるソフトウェアの最新の動向やデータベースに関わるソフトウェアの最近の話題をソフトウェアの要素技術にフォーカスして紹介」

最近[illegible]動向

- PCAST “Designi[illegible]
- 密[illegible]
  - 専[illegible]
- RDBM[illegible]VE,Spark)
- [illegible]のCI、CD、DevOpsへの取り組み
- オブジェクト指[illegible]関数型言語の導入、サービス開発企業による新言語の提供

という話ではない、はず？

PFSとは？



Project info: http://pfs.ipmu.jp/
Project blog: http://pfs.ipmu.jp/blog/

PFSとは？

- 地理的に分散したハードウェア開発
  - ハードウェアに紐づいたソフトウェアの分散開発
  - ハワイに来て初めて全体が繋がる装置構成への対応
- 文化・技術的なバックグラウンドや地理的にも大きな断絶
  - 統一した開発手法の導入が困難
  - 地理的断絶(時差)によるコミュニケーションの非同時性
- ソフトウェアインフラとしてのハードウェアへの制約
  - 変更が困難なすばるの設備、しかも20年選手
  - 完成後5-10年の運用可能性を見据えてのインフラ設計
- 運用中の柔軟な機能更新要求に耐えうるソフトウェア
  - カプセル化などによる相互依存性を削減した設計
  - 単体での更新時動作評価試験への対応

で、現実問題どうするか？

DbC (契約による設計)

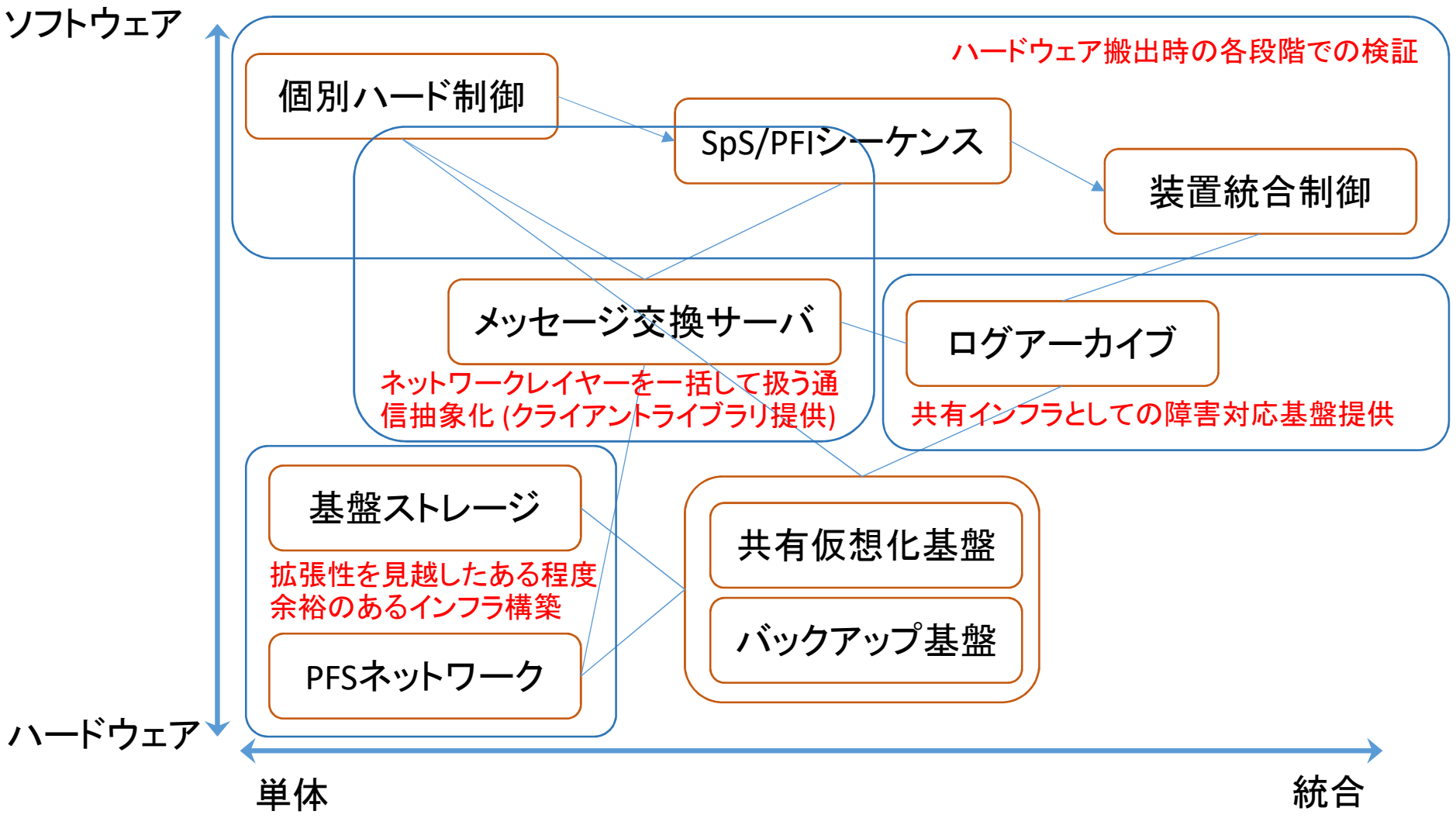
事前条件Pが成り立つときに、プログラムAを実行するとその実行後に必ず事後条件Qが成り立つならば、プログラムAは、事前条件Pと事後条件Qとに関して部分的に正当(partially correct)である。

- ソフトウェアモジュールを階層化してそれぞれの間にDbCコンセプトを導入
- 個別試験・段階別統合の実現可能性を追求
- ソフトウェア・ハードウェアインフラレイヤーについても疎結合の階層化
- パフォーマンス要求には数割～数倍の余裕をみて (ピーク要求を含み)
- 各階層での単体試験・結合試験を独立にできるように可能な限り配慮

「賢明なソフトウェア技術者になるための第一歩は、動くプログラムを書くことと正しいプログラムを適切に作成することの違いを認識すること」(M.A. Jackson 1975)

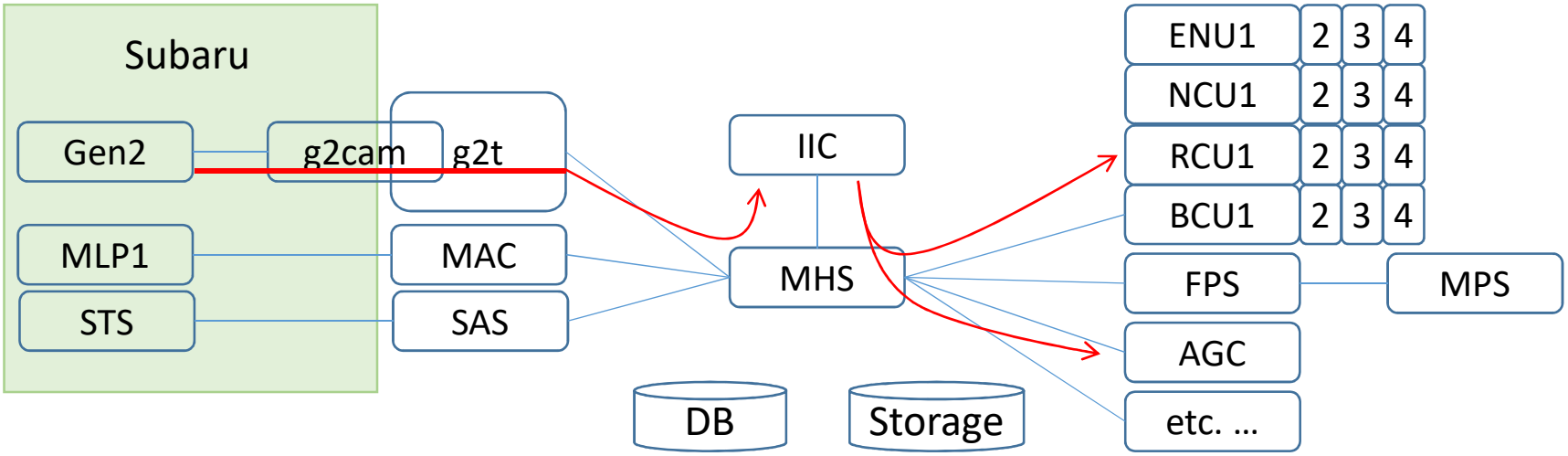
とはいえ理想と現実はいろいろあるわけで・・・・

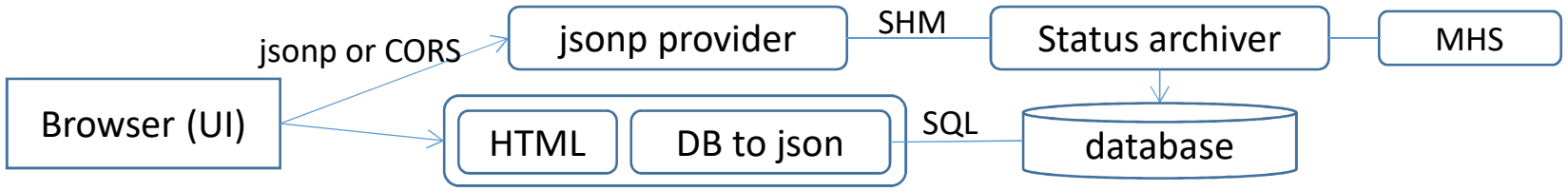
コンセプト


ソフトウェア
ハードウェア
単体
統合
ハードウェア搬出時の各段階での検証
個別ハード制御
SpS/PFIシーケンス
装置統合制御
メッセージ交換サーバ
ネットワークレイヤーを一括して扱う通
信抽象化 (クライアントライブラリ提供)
ログアーカイブ
共有インフラとしての障害対応基盤提供
基盤ストレージ
拡張性を見越したある程度
余裕のあるインフラ構築
PFSネットワーク
共有仮想化基盤
バックアップ基盤

コンセプト (II)

- カプセル化などによる相互依存性を削減した設計
  - クライアントライブラリを提供し通信レイヤーを完全に隠蔽
  - 各ハードウェア制御モジュールではローカル関数呼出で対外接続
  - 高度に分離された個別制御モジュールの集合としての統合制御
- 単体での更新時動作評価試験への対応
- ハードウェアに紐づいたソフトウェアの分散開発
- ハワイに来て初めて全体が繋がる装置構成への対応

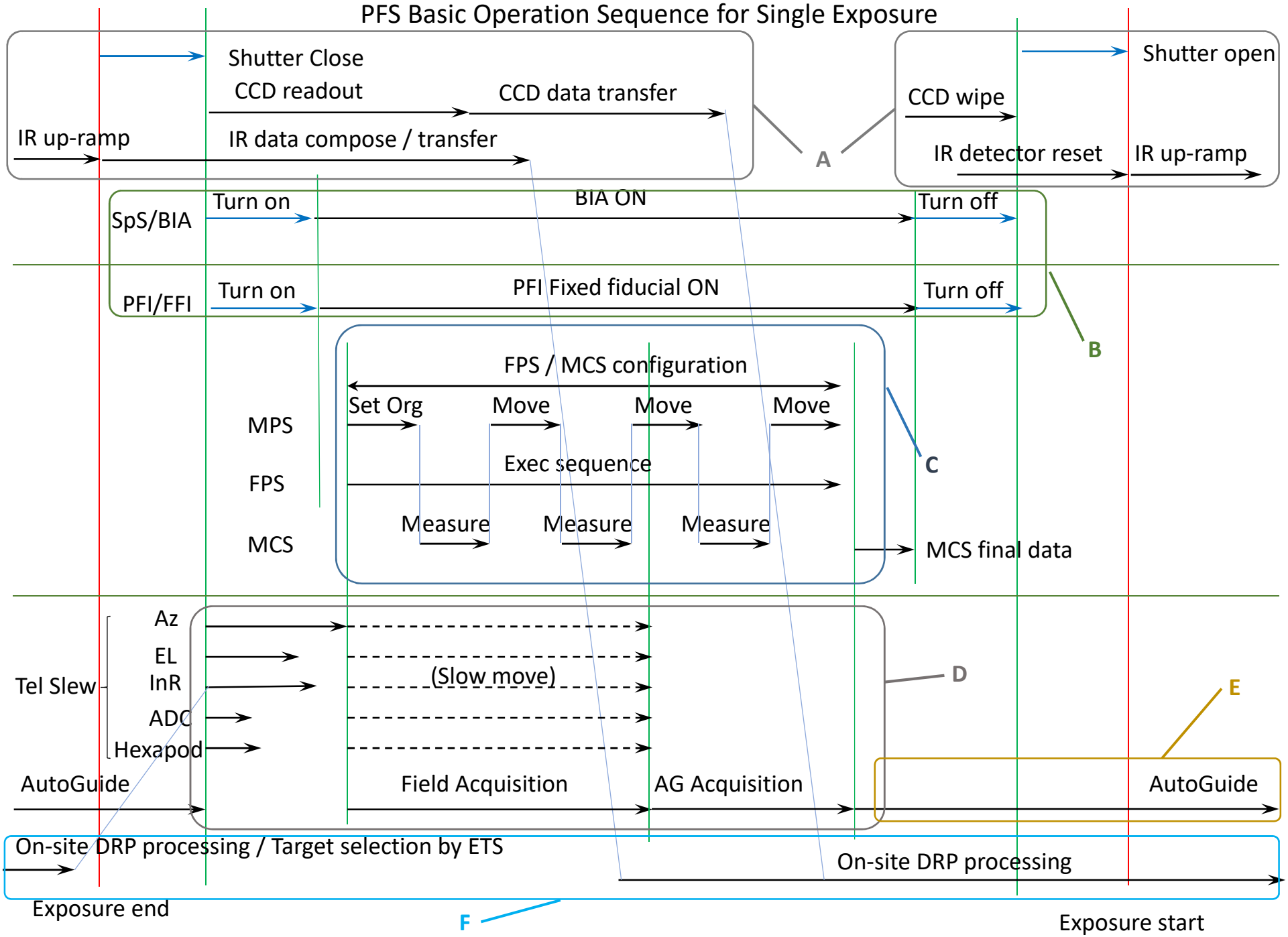

Subaru
Gen2
g2cam
g2t
MLP1
STS
MAC
SAS
IIC
MHS
DB
Storage
ENU1
NCU1
RCU1
BCU1
2
3
4
FPS
MPS
AGC
etc. …


jsonp or CORS
Browser (UI)
jsonp provider
SHM
Status archiver
MHS
HTML
DB to json
SQL
database

コンセプト (II)

- カプセル化などによる相互依存性を削減した設計
  - クライアントライブラリを提供し通信レイヤーを完全に隠蔽
  - 各ハードウェア制御モジュールではローカル関数呼出で対外接続
  - 高度に分離された個別制御モジュールの集合としての統合制御
- 単体での更新時動作評価試験への対応
  - モジュールが公開する関数でのドメイン分離が第一目標
  - かつ、制御モジュール間連携シーケンスの階層化によりドメインの狭域化
- ハードウェアに紐づいたソフトウェアの分散開発

- ハワイに来て初めて全体が繋がる装置構成への対応


PFS Basic Operation Sequence for Single Exposure
Shutter Close
CCD readout
CCD data transfer
IR up-ramp
IR data compose / transfer
A
CCD wipe
Shutter open
IR detector reset
IR up-ramp
SpS/BIA
Turn on
BIA ON
Turn off
PFI/FFI
Turn on
PFI Fixed fiducial ON
Turn off
B
FPS / MCS configuration
Set Org
Move
Move
Move
MPS
Exec sequence
FPS
Measure
Measure
Measure
MCS
C
MCS final data
Az
EL
InR
ADC
Hexapod
Tel Slew
(Slow move)
D
E
AutoGuide
Field Acquisition
AG Acquisition
AutoGuide
On-site DRP processing / Target selection by ETS
On-site DRP processing
Exposure end
F
Exposure start

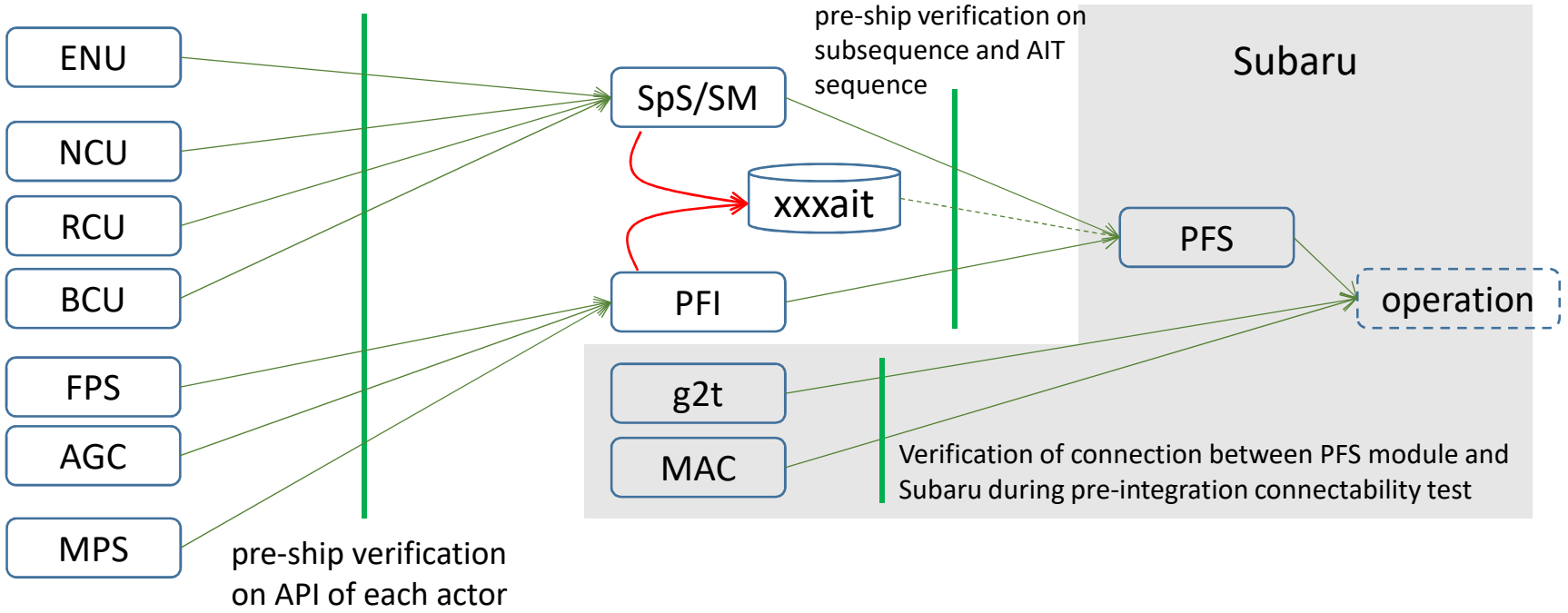
コンセプト (II)

- カプセル化などによる相互依存性を削減した設計
  - クライアントライブラリを提供し通信レイヤーを完全に隠蔽
  - 各ハードウェア制御モジュールではローカル関数呼出で対外接続
  - 高度に分離された個別制御モジュールの集合としての統合制御
- 単体での更新時動作評価試験への対応
  - モジュールが公開する関数でのドメイン分離が第一目標
  - かつ、制御モジュール間連携シーケンスの階層化によりドメインの狭域化
- ハードウェアに紐づいたソフトウェアの分散開発
  - 最もハードウェアに近い個別モジュールは各ハードウェア開発サイトで検証
  - 分光器・主焦点部などに閉じたミニシーケンスによる各部分での検証
- ハワイに来て初めて全体が繋がる装置構成への対応

System verification and integration

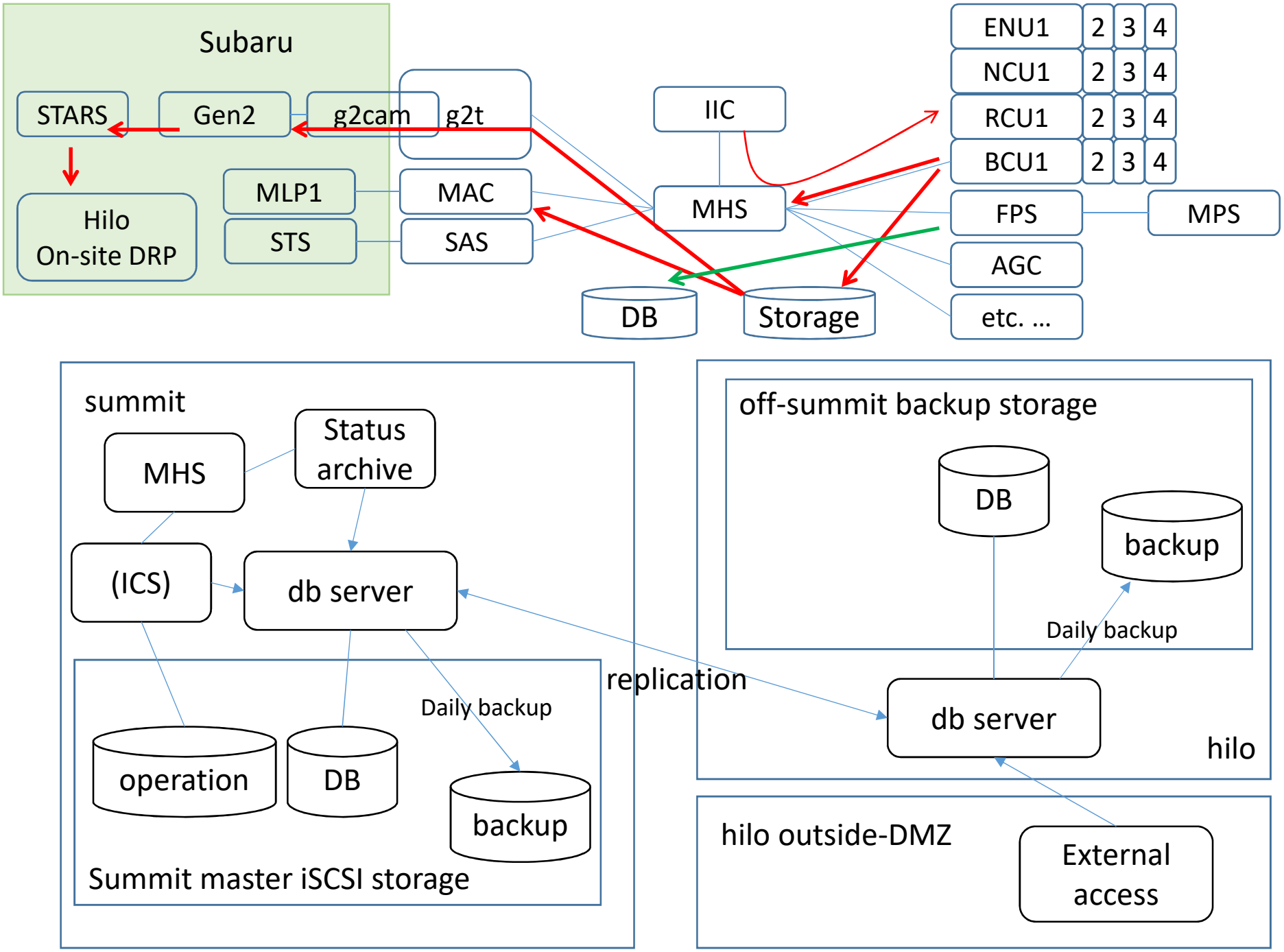
If design follows concepts well, PFS ICS software verification and integration flow will be hierarchic, continue validation at each level and to go next level of integration. Thanks to loosely coupled module based design using MHS, validation of each actor or combination of actors within one (sub-)sequence could be done without any other software module.

These subsystem or module(s) verification are written as sequences and developed as special actor – spsait or pfiait. Using them, software integration and test flow will be 1) per module (actor), 2) per subsystem, whose hardware are at one place before shipping to Subaru (A-E in a last slide), 3) full operational sequence at Subaru.

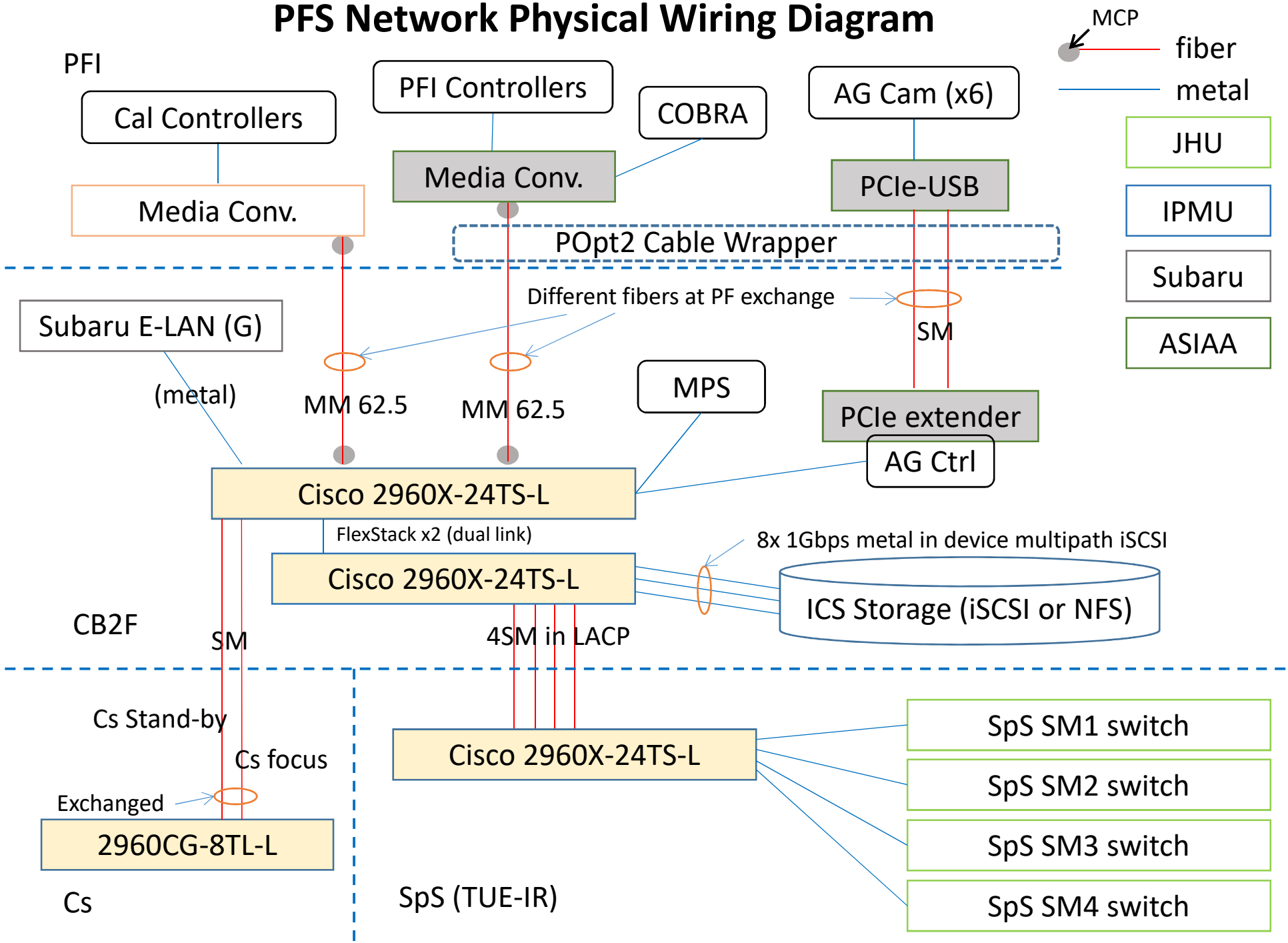


コンセプト (II)

- カプセル化などによる相互依存性を削減した設計
  - クライアントライブラリを提供し通信レイヤーを完全に隠蔽
  - 各ハードウェア制御モジュールではローカル関数呼出で対外接続
  - 高度に分離された個別制御モジュールの集合としての統合制御
- 単体での更新時動作評価試験への対応
  - モジュールが公開する関数でのドメイン分離が第一目標
  - かつ、制御モジュール間連携シーケンスの階層化によりドメインの狭域化
- ハードウェアに紐づいたソフトウェアの分散開発
  - 最もハードウェアに近い個別モジュールは各ハードウェア開発サイトで検証
  - 分光器・主焦点部などに閉じたミニシーケンスによる各部分での検証
- ハワイに来て初めて全体が繋がる装置構成への対応
  - データ送受信などインフラ部分は別途パフォーマンス検証
  - シーケンス中の大容量データの流れはモジュール内制御と独立して検証
  - 可能な部分はデータ変換・処理・転送モジュールを単機能分離する構成


Subaru
STARS
Gen2
g2cam
g2t
Hilo
On-site DRP
MLP1
STS
MAC
SAS
IIC
MHS
DB
Storage
ENU1 2 3 4
NCU1 2 3 4
RCU1 2 3 4
BCU1 2 3 4
FPS
MPS
AGC
etc. …
summit
MHS
Status archive
(ICS)
db server
operation
DB
Daily backup
backup
Summit master iSCSI storage
replication
off-summit backup storage
DB
backup
Daily backup
db server
hilo
hilo outside-DMZ
External access

PFS Network Physical Wiring Diagram



さて、現実を見よう（涙）

- 文化的・技術的なバックグラウンドに大きな断絶
  - 開発支援・管理ツールの利用への温度差から共通文化の構築が困難
  - 自己開発・自己利用でない観測装置の運用に向けた意識改革必要？
  - コードレビューやりたいけれど・・・・
- 最もハードウェアに近い個別モジュールは各ハードウェア開発サイトで検証
  - 各グループの技術バックグラウンドが品質に直結(してる感じはある)
  - また、当然ながらハードウェア開発の遅れの影響はそのまま効きます
  - ハードウェアの可制御性の検証結果がドメイン境界に影響する悩みも同じ
- カプセル化などによる相互依存性を削減した設計
  - 原理的にはIFより後ろではモジュール内で何をやってもらっても大丈夫
  - というよりは、その手の放任？放置？主義をとるしかない人手不足
  - そもそも(PO/すばるで)誰がレビューやるの？
- データ送受信などインフラ部分は別途パフォーマンス検証
  - 大規模化してインフラ・セキュリティーなどの重要性が・・・。どう協働する？
  - 湯水のように性能が出るわけではない！ソフトウェア設計でも考慮しないと