OCP Test and Validation: Optimizing Repair Strategies with OCP Diagnostics

EMPOWERING OPEN.

OCP GLOBAL SUMMIT
OCTOBER 18-20, 2022
SAN JOSE, CA

OCP Test and Validation: Optimizing Repair Strategies with OCP Diagnostics

TEST & VALIDATION

Dan Frame, SWE Manager, Google
Yuanlin Wen, Tech Lead Manager, Google
Ming Xia, SWE, Google
Richa Mishra, Data Science Technical Lead, Meta

OPEN PLATINUM™

OCP Diagnostics and Optimizing Repair Strategies

Today's Presentation

- Why is repair optimization important?
- What makes this problem difficult?
- What are the two common approaches to creating a repair strategy for a new hardware platform?
- How is the OCP Test and Validation Track helping to solve the problem?
- ***How can you participate?***

Why is this problem important?

- When customers move their applications to the cloud, they expect not only lower costs, but also greater availability.
    - If machines are offline, they are not contributing to redundancy.

- The Capex and Opex costs of machines continue to accrue when the machine is not serving.

- Downtime is ***expensive***.
    - Cost of the repair
    - Depreciation of the asset that is no longer serving.
    - Average downtime for machine repairs must be incorporated into the spare capacity modeling.

Error Characterization Workflow

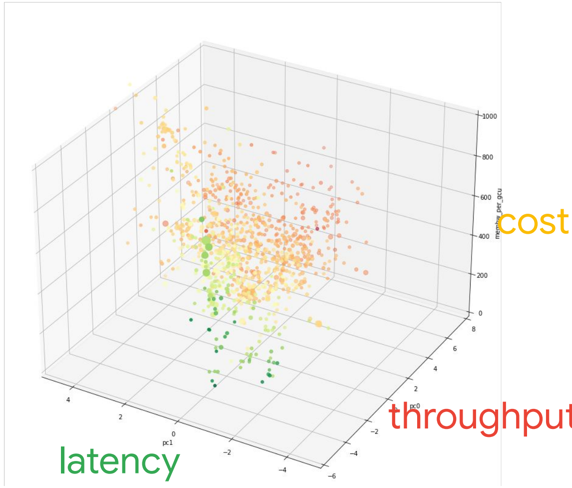
Why?
Serving large scale social media applications translates to dense workloads running across Infra hardware. In these environments HW failures are inevitable, however characterizing these errors makes it easier for debugging and crucial for efficient analyses to diagnose faults.

Fleet Heterogeneity makes the problem harder

Fleet Heterogeneity is increasing due to workload demands

- Static repair strategies are harder to maintain and keep up to date for different hardware types.

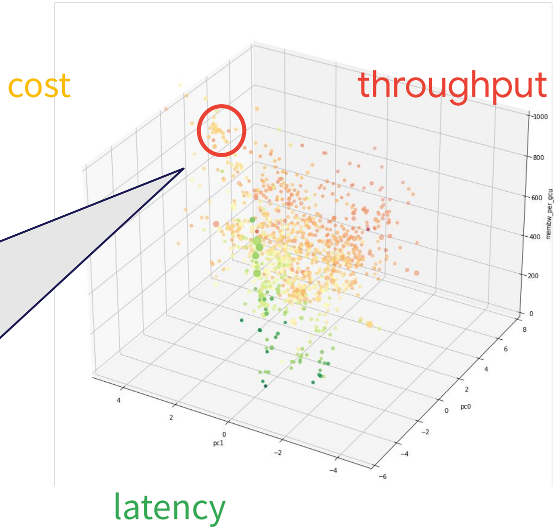
- Each different machine configuration/type requires a spare set of identical machines to replace them.
    - As the number of machine types increases the number of total 'spare' machines also increases.
    - Efficiency of returning the machine back to service is essential.



Fleet Heterogeneity makes the problem more urgent…

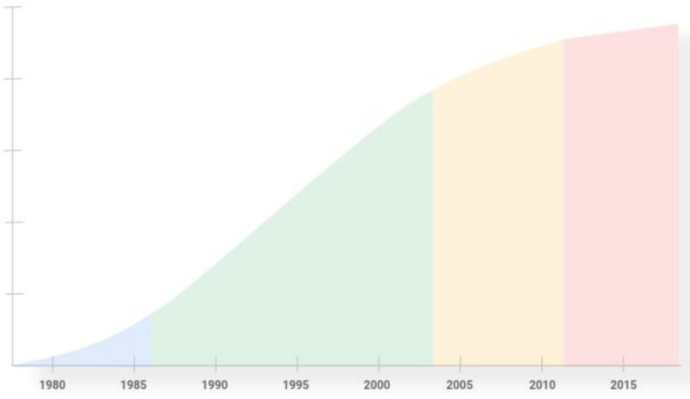
Rare Machines are the most essential to repair quickly.

- They are rare which means replacement machines are also rare.
- Rare machines are typically the most expensive for specialized applications.



"Scale Out" increases repair strategy complexity

As CPU Performance plateaued due to the End of Moore's Law, this led to 'scale out' vs. 'scale up' architectures, typically with dedicated ASIC's for specialized workloads such as highly parallelized ML. These scale-out architectures have more defect opportunities, and are more difficult to generate repair strategies.

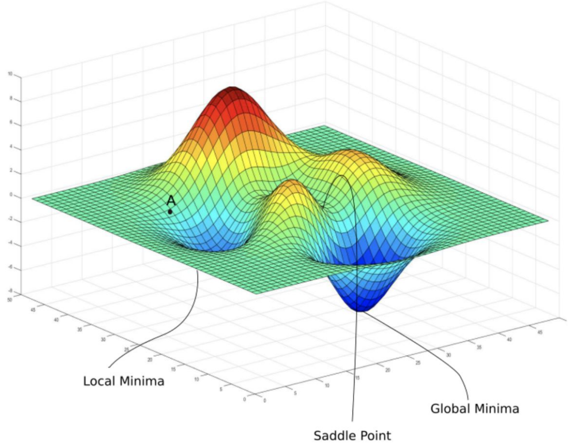


CPU Performance Plateau
(Courtesy- S422 Computer Architecture 5th Edition, Hennessy/Patterson)

Massive Parallel Interconnected Architectures

Repair Strategies

A repair strategy is an attempt at finding an optimal minimum to the total overall *cost* of the diagnosis and repair of a defective device. This challenge of *repair at scale* is common in manufacturing operations, and large data center fleets.



The *cost of a repair* may be evaluated differently for various situations:

- Minimize the overall "dollar cost" to repair the device.
- Minimize the overall "time to repair" cost.
- Balance the "dollar cost" and "time to repair" costs.

What goes into a Repair Strategy?

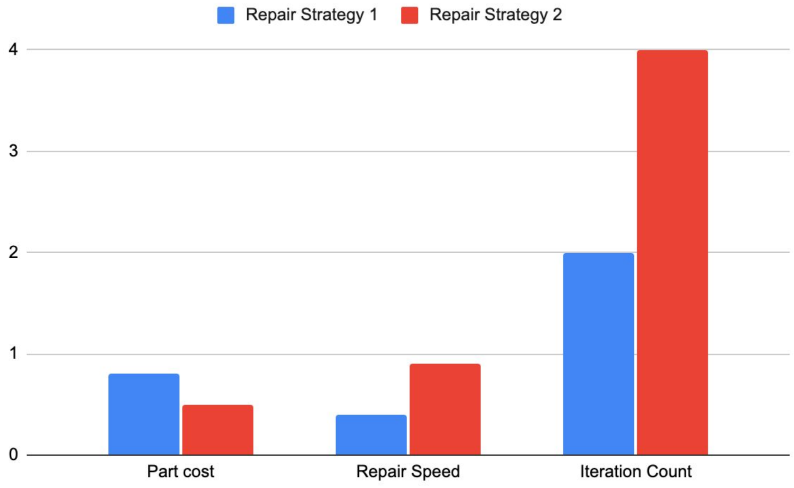
"What is the probability of particular components failing?"	*"What are the symptoms that have been observed that indicate a non-conformance?"*	*"How would a hardware expert recommend resolving this symptom?"*	*"How should I optimize the repair strategy for this environment? (Cost, Speed, Minimum Iterations)"*
"What has been previously repaired on the device?"	*"What components are failing at the same time?"*	*"How are the components connected together?"*	*"What environmental factors should I consider? (Location, Temperature, etc)*
"What repair typically fixes this problem?"	*"What do I know is working correctly in the system?"*	*"What test coverage do I know I have?"*	*"What are the interconnected devices with this device under test?"*
Historical Information	Test Result Information	Product Information	Environmental Information

Example of Cost Factors for Repairs

Assume "cost" of a repair strategy is calculated as:

```
Cost_repair = Cost_part +
              Cost_labor +
              Cost_downtime +
              Cost_migration
```

Different repair strategies can achieve the same goal, but may come at a different cost. Depending on the business need and business-driven cost figures for downtime and migration, an optimization can be sought between the part cost and repair speed (labor, downtime, migration)



2 Approaches to Generating a Repair Strategy

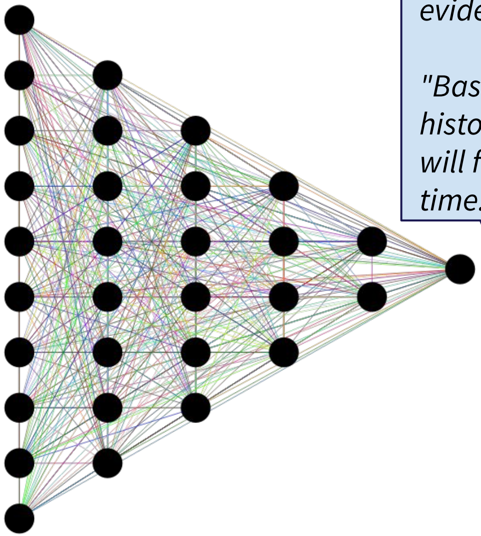
Hardware expert encodes knowledge in a grammar:

"If 'CPU0' is hot, then you first check the heatsink torque, then check the fan, then check the BMC, then check the CPU…"

Rule-Based Systems

Model evolves in response to new evidence:

"Based on the test results, and historical data, reseating the CPU will fix the problem 89% of the time."



Historical Inference

Repair Strategies: Rule Based

Rule based repair or debug strategies have been in use since the early days of electronics. Sometimes these rulesets are referred to as 'debug trees', or 'debug actions'.

These debug trees usually link a symptom to an ordered list of repair actions which may be prioritized by the likelihood to fix the problem or a different weighting mechanism such as repair cost/difficulty.

Typically, they include some type of additional weighting based on product history, cost factors, or risk of damage to the product.

FAIL-LC-CONN

Action:

1). Check console cable connection from line card to console cable. If failed go to step 2

2). Swap console cable with another line card and re-test. Go to step 3

3). If failure moves with line card, replace console cable. If failure stay on the same line card, reseat line card. Go to step 4

4). If failed, replace line card and run RFT to update line card Go to step 5

5). re-test and If failed, go to step 1

NCSI-FAIL

Action:

1). Change loopback and reseat sliver cable and re-test Go to step 2

2). If failed, replace Sliver cable on the line card with a loopback and re-test. Go to step 3

3). If failed, replace line card and run RFT to update line card. Re-test after RFT line card update. Go to step 4

4). If failed, go to step 1

OBO-PATH-FAIL

Action:

1). Re-seat obo fiber on card, clean loopback fiber, and re-test. Go to step 2

2). Open up distribution panel, clean MPO fiber, reseat connectors, and re-test. Go to step 3

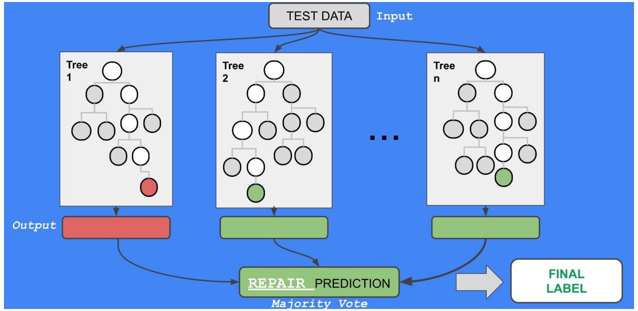
3). If failed replace OBO fiber and re-test. Go to step 4

4). If failed, go to step 1

Repair Strategies: Inference Models

Inference Based Repair Strategies use training datasets of labeled data to build models that can be used to generate repair predictions based on live data.

There are many types of inference models, but we have found that classic ensemble methods such as *random forests* have been proven to work very well and mature implementations are available in many open source ML libraries and frameworks.



Which approach works best?

Rule-Based Systems

- + Simple and direct linkage from domain knowledge to repair logic
- + Reasonably quick to ramp up to a good coverage with little tuning effort
- + Straight-forward validation and debugging

- - Requires incremental development for new symptoms and failure modes
- - Requires maintenance on growing rule set and repair logics as fleet size increases
- - Depends on domain knowledge to compose good rules
- - Software bugs/issues can complicate rule composition

Historical Inference

- + Evolves as more information is available.
- + Handles combinations of scenarios well.
- + Automatically determines what is 'signal' and what is 'noise'

- - Doesn't work when limited data is available.
- - Not always obvious why some features strongly outweigh others.

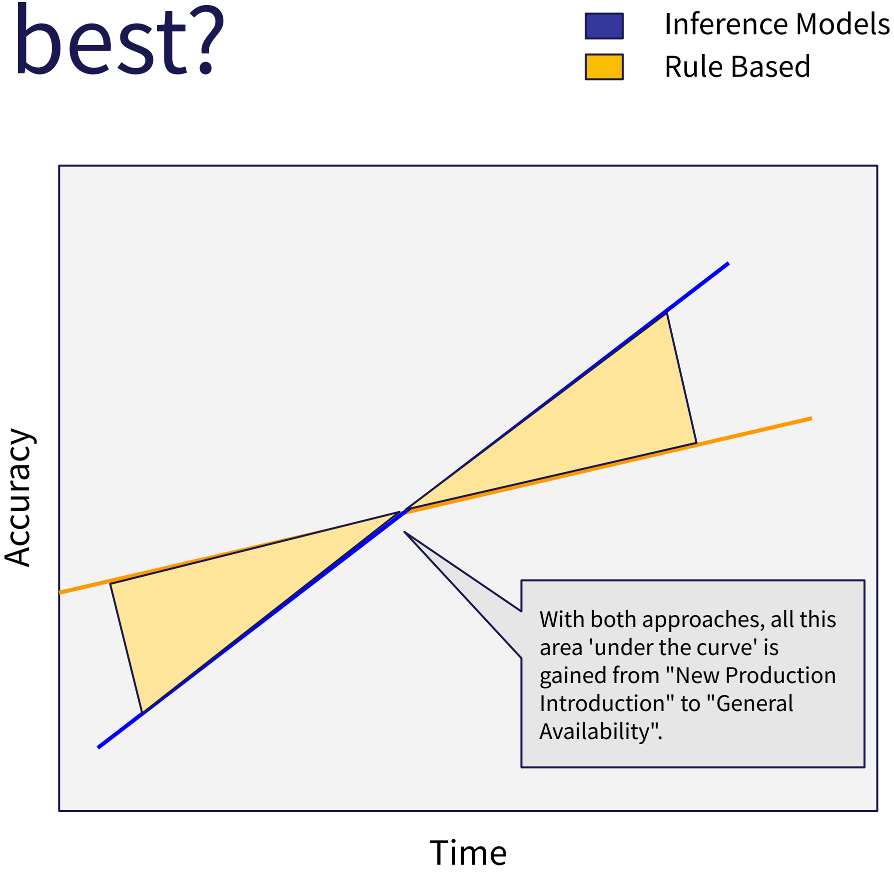
Which approach works best?

Rule based models are typically more effective early when no historical data is present, but they help build the dataset which allows the training of the historical model which quickly surpasses the accuracy of the original rules.

You need ***both*** approaches:

1) You need to initial heuristics to "kick start" inference based approaches.

2) When the historical dataset has become large enough, inference based approaches provide continuous improvement until a plateau on hardware diagnosis accuracy is reached.

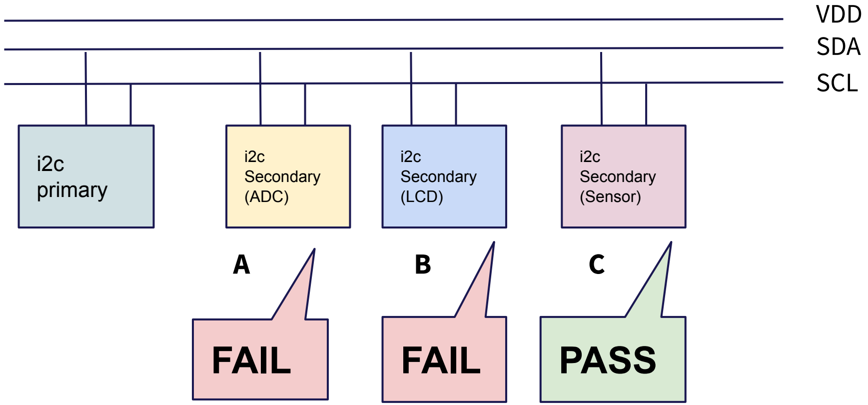


Why are repair heuristics difficult to generate?

In this common case, we have an I2C primary device, and three secondary devices.

Two of the secondary devices are failing connectivity tests, and one is passing.

Root Cause: The 'Passing' device is holding the bus and causing the two properly working secondary devices from utilizing the bus leading to failures.



Repair Logic:

If !A && B && C: Repair A
If A && !B && C: Repair B
If A && B && !C: Repair C
If A && !B && !C: Repair A
If !A && B && !C: Repair B
…
….

The combinatorial explosion of symptom combinations make heuristics very difficult to generate for non-trivial scenarios.

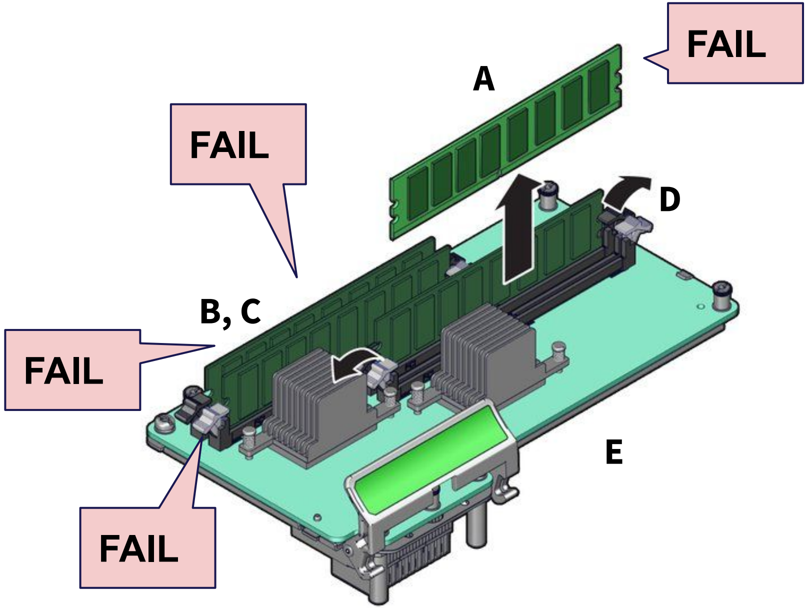
Why are repair heuristics difficult to generate?

In this scenario, we have a DIMM interposer card, and 4 failing DIMMs, on separate channels.

Root Cause: The DIMMs are all functioning properly, but the 'upstream' interposer is defective.

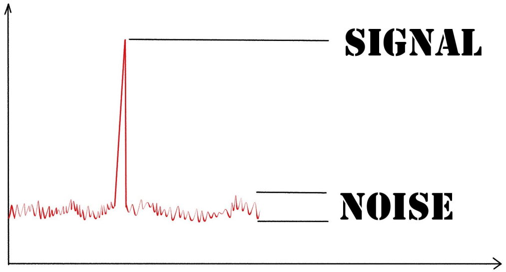
Repair Logic:

If !A && !B && !C && !D && E: Repair E
If !A && B && C && D && E: Repair A
If A && !B && C && D && E: Repair B
…
…



Why are repair heuristics difficult to generate?

There are so many signals that are collected from testing that are discarded as 'noise' in typical rule based systems, but these all provide useful signal:



- Software Errors
- Kernel Logs
- Crash dumps
- Explicit Test Results
- Execution Times
- Parametric Information (Temps, Fans, CPU Loads)
- System Event Log Messages
- BIOS Post Codes
- Schmoo Timing

All of these can provide extremely valuable signal for repair prediction, but it's infeasible to develop heuristics for all of them. This is an area where inference models can show great value.

Why are repair heuristics difficult to generate?

Lost Signal…

OCP compliant diagnostics emit both passing and failing diagnostics.

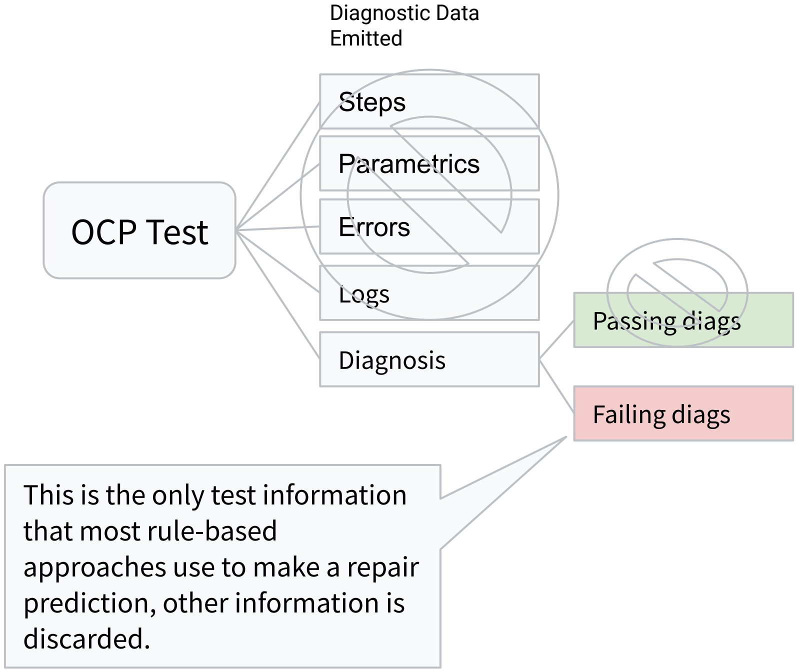
Most tests emit far more passing information than failing information.

Passing Diagnostics ***exonerate*** components and mark them as not suspect.

Failing Diagnostics ***indict*** components and mark them as suspect, but this information should be combined with the entire result set of a particular test execution.

In addition, diagnostics emit 'error' symptoms, but it's proven difficult to incorporate these into heuristics due to the large scope.

This lost signal leads to unnecessary repair attempts that do not correct the problem.



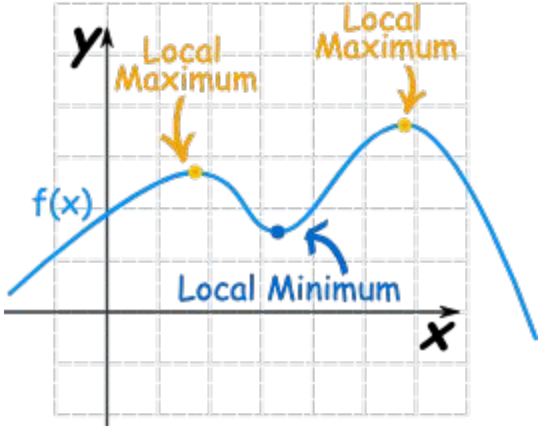
Can Rule Based Repair Strategies be optimized by algorithm?

Rule based repair strategies reach a localized minimum which is considered "good enough", and then rarely are changes made to try to achieve "long tail" optimization.

This can be addressed with optimization techniques such as simulated annealing.

Changes that result in large reductions in the efficiency are allowed initially, but as the system under optimization "cools", fewer bad decisions are allowed. This can allow a system to fall out of a 'localized minimum' to a more ideal set of rules.

In some instances, rule-set optimization can yield significant improvements, but in many cases, the annealing of the ruleset is too costly for implementation except in very high volume, low cost scenarios.



Case Study on Rule-Based Repair Strategies

Machine configuration: A new combination of hardware.
Fully Automated Repair Rate: The percentage of time that an collection of repair rules accurately diagnosed the problem without human intervention.

Figure 1 indicates that the effectiveness of rule-based systems is lower in newer products.

Figure 2 indicates that the effectiveness of rule-based systems decreases as shared rules are applied to more heterogeneous platforms.

This might be caused by the diagnostic coverage decay, untweaked repair rules, or in general relying on granular symptoms to build repair strategy becomes less sustainable with the increasing complexity of the machine configuration.

Figure 1. Fully automated repair rate vs machine configuration age

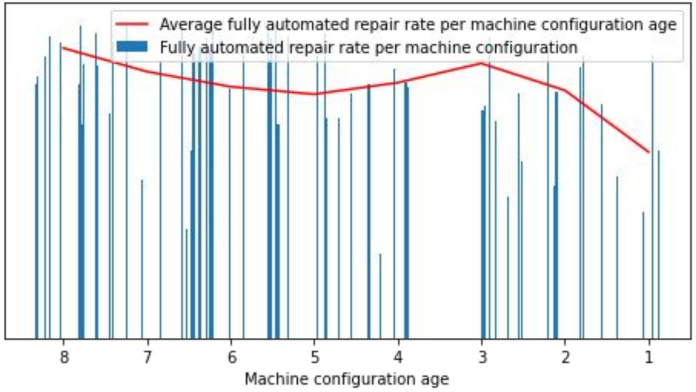
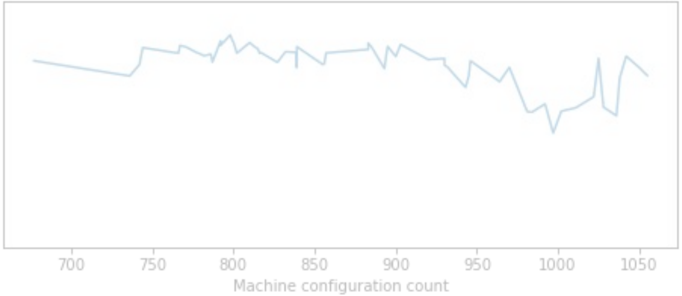


Figure 2. Fully automated repair rate vs total number of machine configurations



Case Study on Rule-Based Repair Strategies

Machine configuration: A new combination of hardware.
Fully Automated Repair Rate: The percentage of time that an collection of repair rules accurately diagnosed the problem without human intervention.

Figure 1 indicates that the effectiveness of rule-based systems is lower in newer products.

Figure 2 indicates that the effectiveness of rule-based systems decreases as shared rules are applied to more heterogeneous platforms.

This might be caused by the diagnostic coverage decay, untweaked repair rules, or in general relying on granular symptoms to build repair strategy becomes less sustainable with the increasing complexity of the machine configuration.

Figure 1. Fully automated repair rate vs machine configuration age

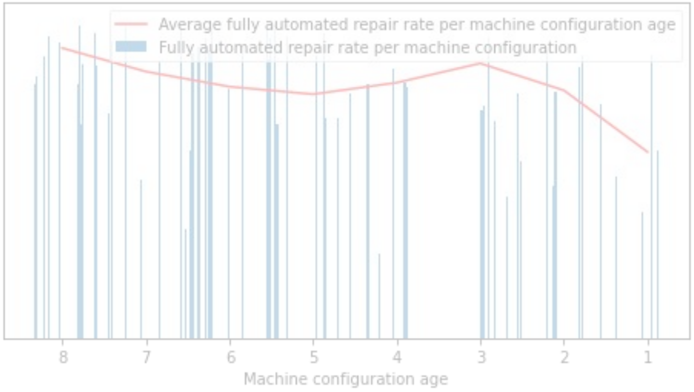
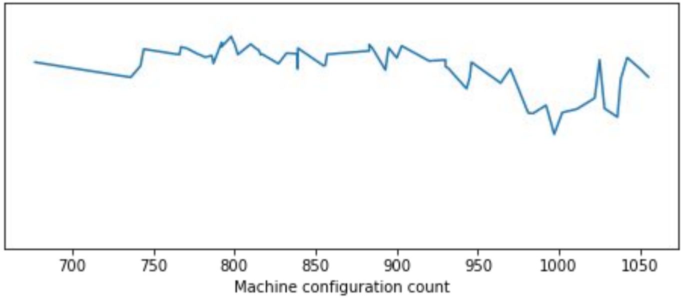
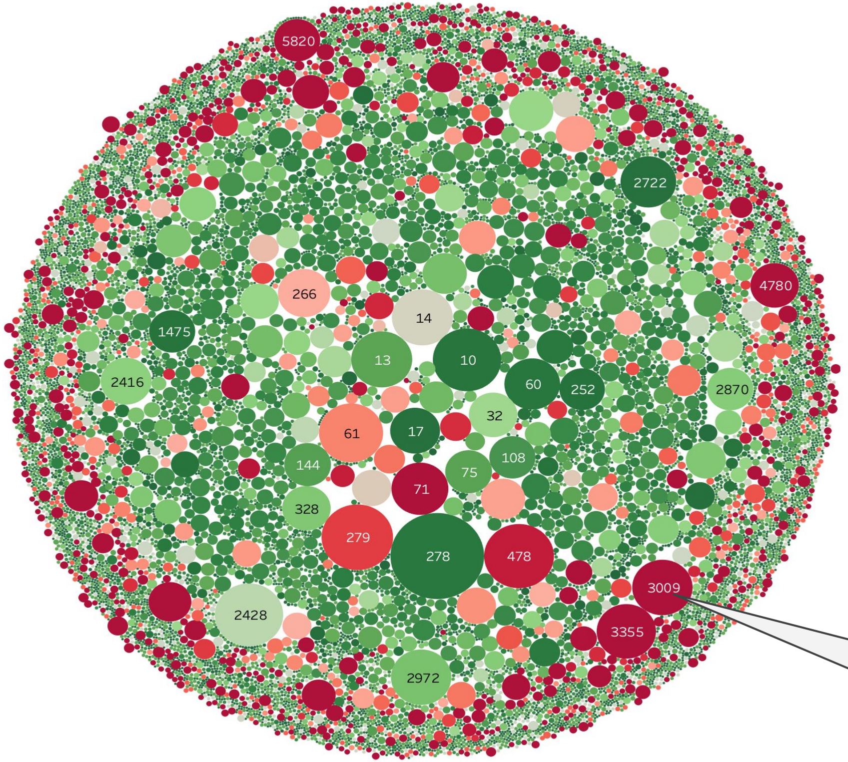


Figure 2. Fully automated repair rate vs total number of machine configurations



Case Study on Rule-Based Repair Strategies



A ***syndrome*** is a unique set of all the different outcomes that a failing test can exhibit based on the diagnosed components, and the results of those diagnoses.

Size = Number of Occurrences of the Syndrome
Color = Accuracy of Recommended Repair

- An average of less than 2 repair attempts to arrive at a successful repair is considered neutral (white)
- < 2 repair attempts is green.
- > 2 repair attempts is progressively red.

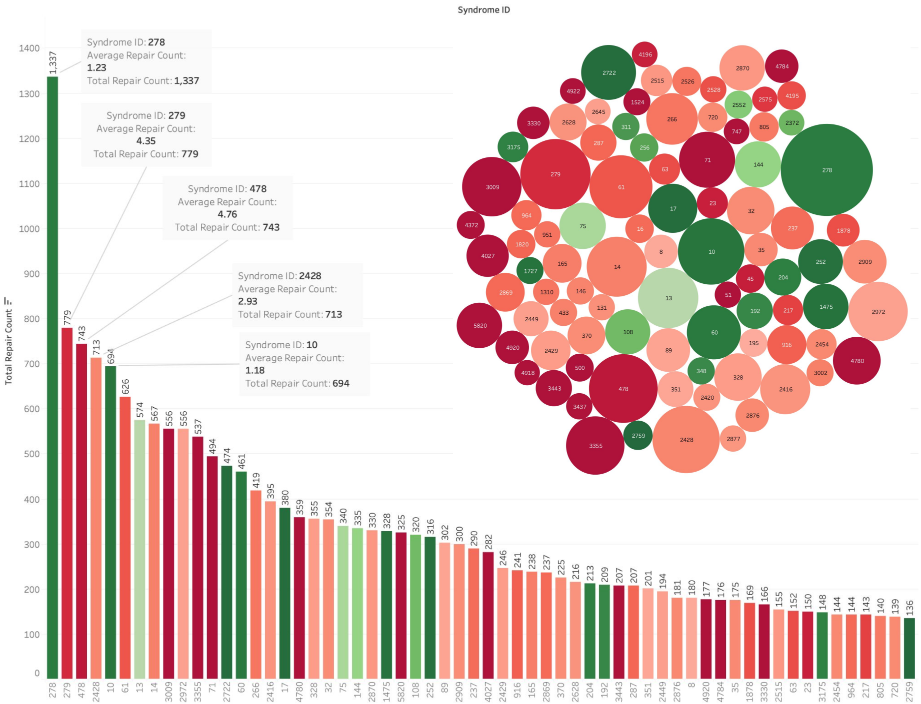
Sample…

Syndrome ID:	3009
Avg. Total Repair Count:	556
Average Repair Count:	7.13

Without inference-based repairs, inefficiencies exist in even *effective* sets of repair heuristics…

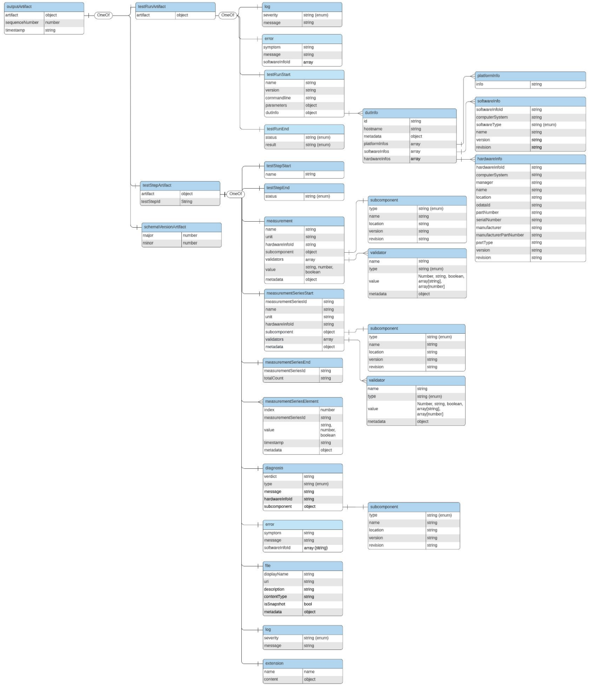
Syndrome	Occurrence Rate	Average Repair Attempts Required for Resolution
278	1337	1.23
279	779	4.35
478	743	4.76
2428	713	2.93
10	694	1.18

- *Of the 5 most common failure syndromes, 3 have large opportunities for optimization.*



What do we need to build a model?

Too much data makes models difficult to manage.
Too little data makes them ineffective.



We need structured data.

What is the OCP Test and Diagnostic Output Specification?

A standard template for diagnostic output that has been proven to provide essential data necessary for anomaly detection, statistical process control, straight forward ML modeling for repairs, and compatibility with rule-based repair strategies used in manufacturing and operations for hyperscalers.

https://github.com/opencomputeproject/ocp-diag-core

Applying the OCP Diagnostic Specification for Repair Prediction

Hardware Details	Diagnosis and Verdicts	Parametrics
Software Details	Errors and Logs	Time Series Data

Provides training features for ML, and branch conditions for heuristics.

Diagnosis/Verdicts form the core of most rule-based conditions, which can also be used as ML inputs.

Provides statistical process control and anomaly detection.

OCP T&V Output: Device Under Test Features

DUTInfo record for each test record provides data for repair rule logic.

- What type of device was tested?
- What is the software configuration?
- What is the hardware configuration?

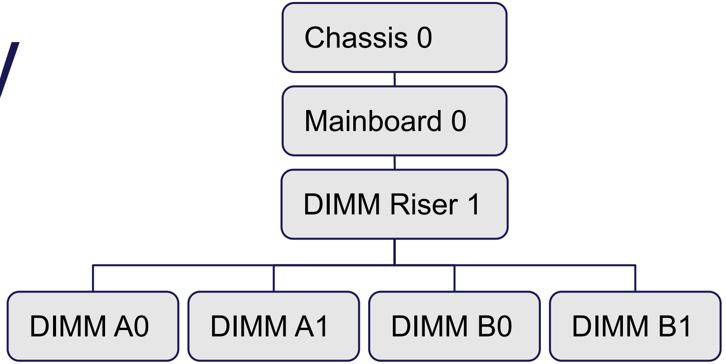
dutInfo	
id	string
hostname	string
metadata	object
platformInfos	array
softwareInfos	array
hardwareInfos	array

platformInfo	
info	string

softwareInfo	
softwareInfoId	string
computerSystem	string
softwareType	string (enum)
name	string
version	string
revision	string

hardwareInfo	
hardwareInfoId	string
computerSystem	string
manager	string
name	string
location	string
odataId	string
partNumber	string
serialNumber	string
manufacturer	string
manufacturerPartNumber	string
partType	string
version	string
revision	string

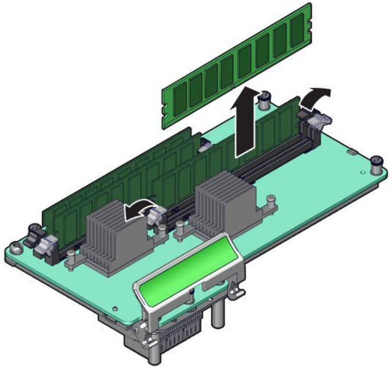
Redfish Device Connectivity



In order to factor in hardware topology, it's critical that master data exist that defines this topology. In some cases, this is knowledge has been incorporated into the set of repair rules. For inference models though, this information needs to explicitly exist as ***master data***.

This is satisfies two needs:

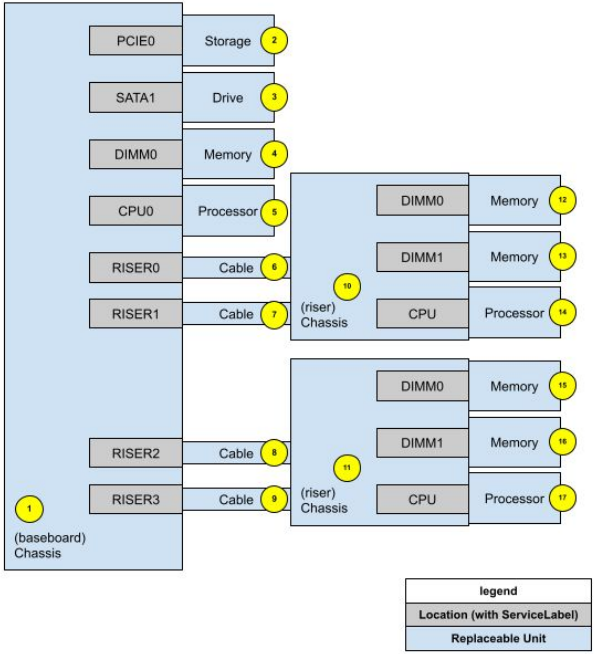
- It provides a consistent name for the hardware to reference for repairability strategies.
- It provides the topology information



Redfish Device Connectivity

```
{
  "@odata.type": "#Cable.v1_2_0.Cable",
  "Links": {
    "DownstreamChassis": {
      "@odata.id": "/redfish/v1/Chassis/some_plugin"
    },
    "UpstreamChassis": {
      "@odata.id": "/redfish/v1/Chassis/container_chassis"
    },
  },
  "Location": {
    "PartLocation": {
      "LocationType": "Slot",
      "ServiceLabel": "EXPANSION0"
    }
  }
}
```

ServiceLabel and Links in Redfish can be used to define a full location in OCP Diagnostics



1. /phys
2. /phys/PCIE0
3. /phys/SATA1
4. /phys/DIMM0
5. /phys/CPU0
6. /phys/RISER0
7. /phys/RISER1
8. /phys/RISER2
9. /phys/RISER3
10. /phys/RISER0/DOWNLINK
11. /phys/RISER2/DOWNLINK
12. /phys/RISER0/DOWNLINK/DIMM0
13. /phys/RISER0/DOWNLINK/DIMM1
14. /phys/RISER0/DOWNLINK/CPU
15. /phys/RISER2/DOWNLINK/DIMM0
16. /phys/RISER2/DOWNLINK/DIMM1
17. /phys/RISER2/DOWNLINK/CPU

These can be concatenated to form a 'device path' which represents the physical topology.

https://github.com/google/ecclesia-machine-management/blob/master/ecclesia/lib/redfish/g3doc/topology.md

OCP T&V Output: Diagnosis

Diagnosis and errors are associated to particular hardware components, or software.

This information together creates 'syndrome' which provides a repair model:

- What components in the machine are indicted as suspect, and a unique feature indicating the observed non-conformance.
- Any errors that occurred, and the software source that generated them.

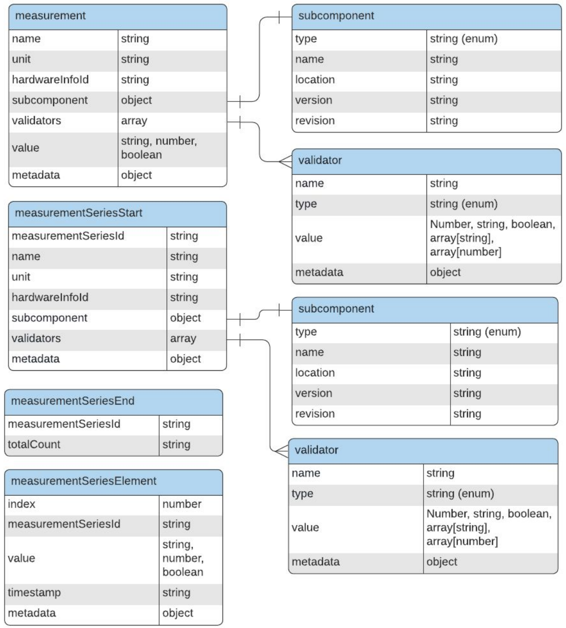
diagnosis	
verdict	string
type	string (enum)
message	string
hardwareInfoId	string
subcomponent	object

error	
symptom	string
message	string
softwareInfoId	array (string)

subcomponent	
type	string (enum)
name	string
location	string
version	string
revision	string

OCP T&V Output: Parametrics

Individual Measurements and Time Series data provide for anomaly detection for hardware and software issues observed on the system under test.



Building an Inference Model…

- Diagnostics test results are combined with historical repair information and device features captured from the diagnostics executing on the device.
- Data is encoded into boolean features as present/absent or category-encoded.
- Creates a very good training set for categorization models for supervised learning.

Device Features			Test Results		History		Environment	
Platform Info	Hardware Info	Software Info	Verdicts	Locations	Repair History	Part History	Location	Environment

Platform Info

is_platform_x: 1
is_platform_y: 0
…
…

Hardware Info

is_amd_milan: 0
is_intel_spr: 1
has_dimm_x343: 1
Has_drive_sas0: 0
…
…

Software Info

kernel_version: 22
bios_version: 88
bmc_version: 32
disk_firmware: 102
…
…

Test Results

dimm_cecc_error_acceptable_dimm0: 1
dimm_uecc_error_acceptable_dimm0: 0
dimm_detected_dimm0: 1
disk_reallocation_count_acceptable_disk0: 1
…
…

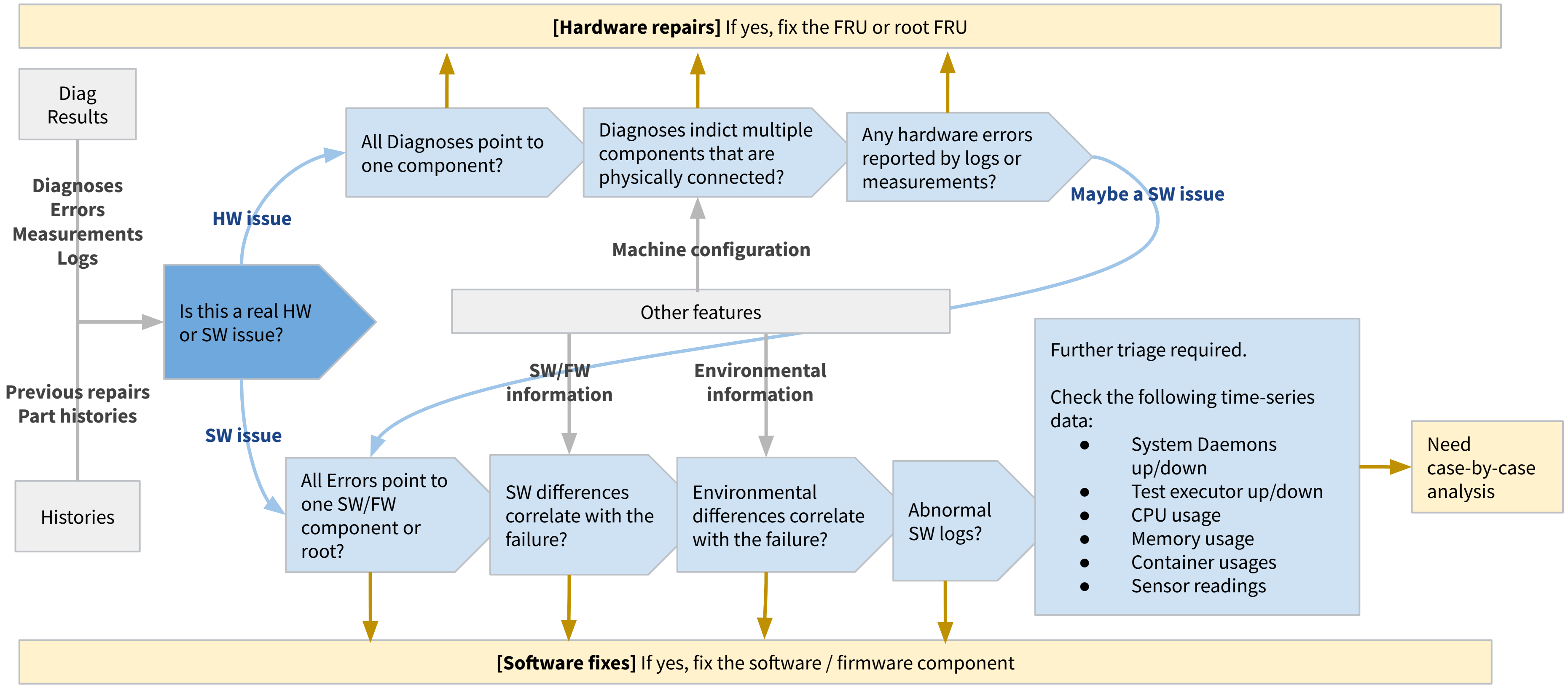
History

repair_location: dimm0
repair_type: reseat
dimm0_current_badness: 0
…
…

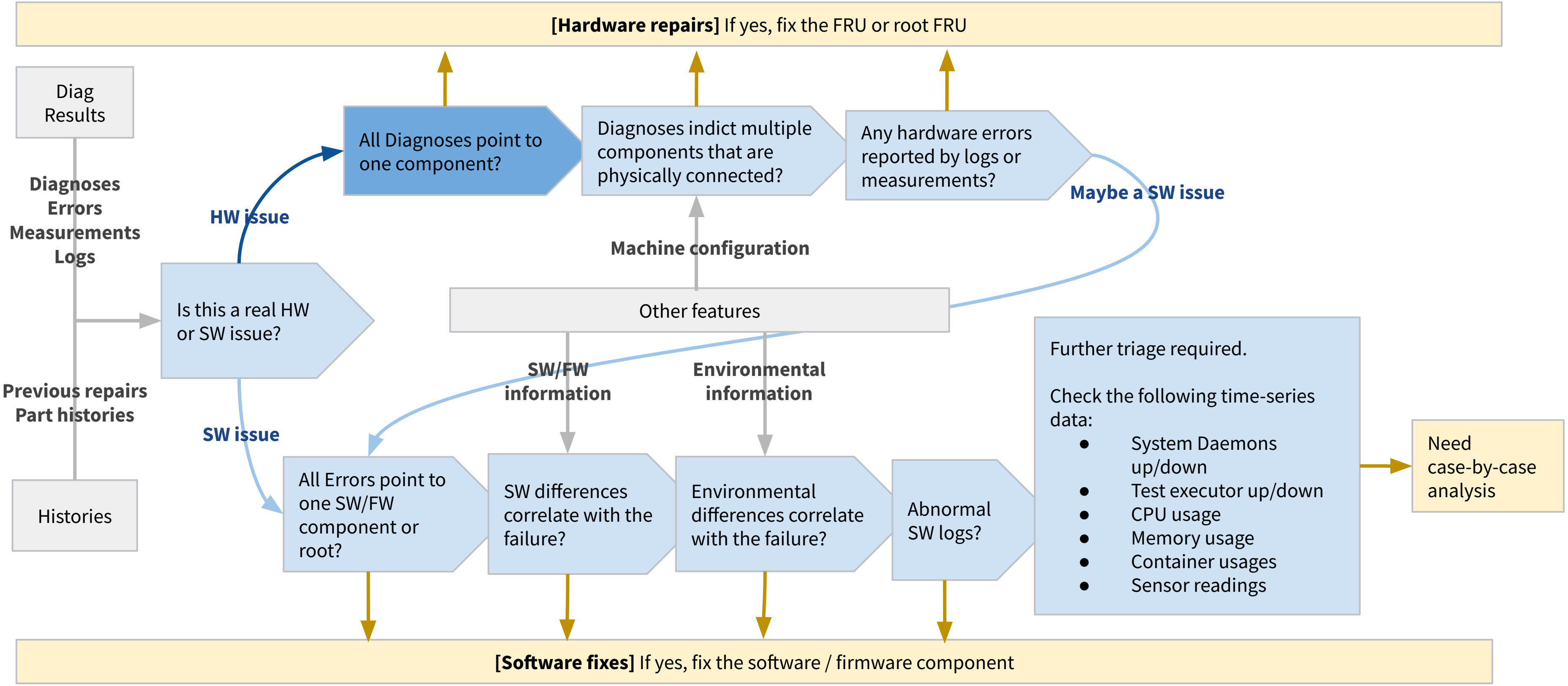
Environment

factory_site: factory123
dc_site: BLD2-A
rack_location: 0/0/1
Inlet_rack_temp: 23
outlet_rack_temp: 88
coolant_presure: 40psi
…
…

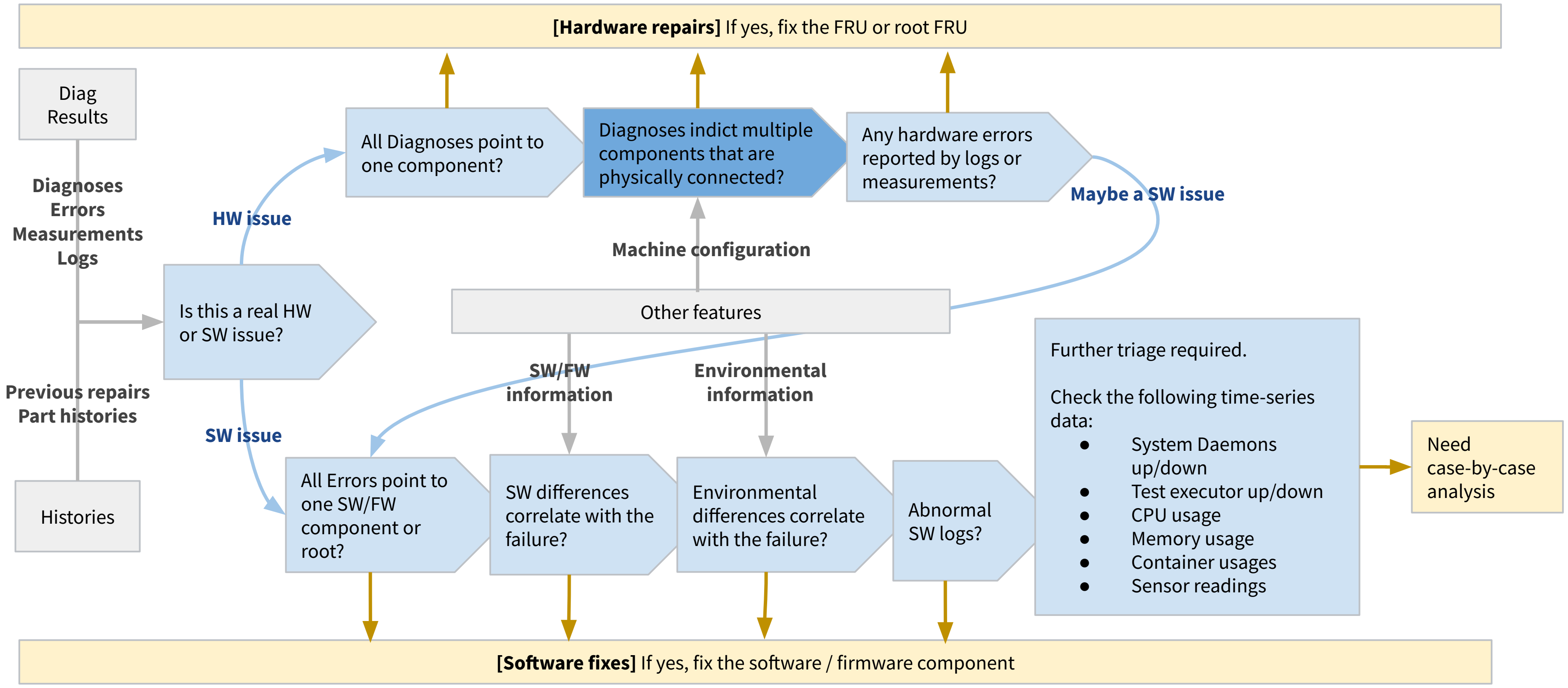
Learning from the 'Human Expert Approach'



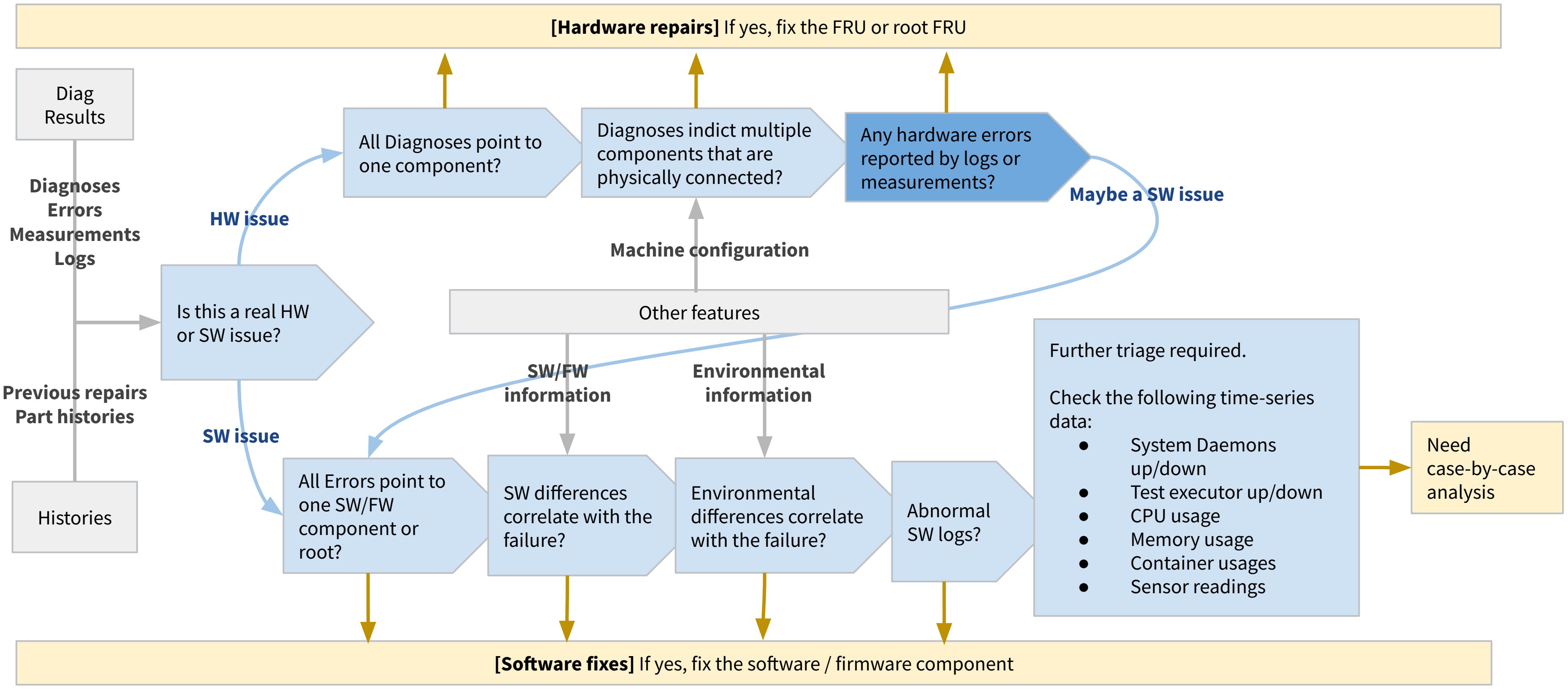
Learning from the 'Human Expert Approach'



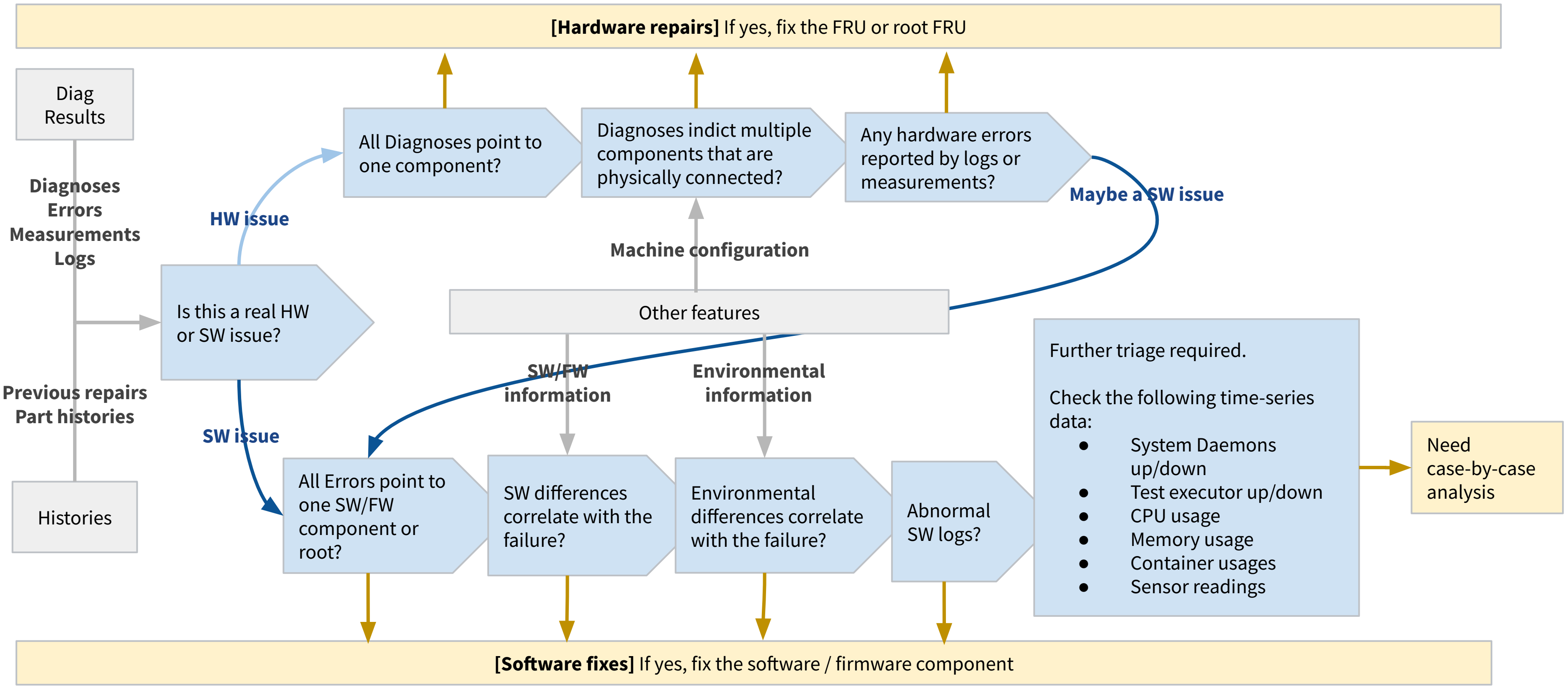
Learning from the 'Human Expert Approach'



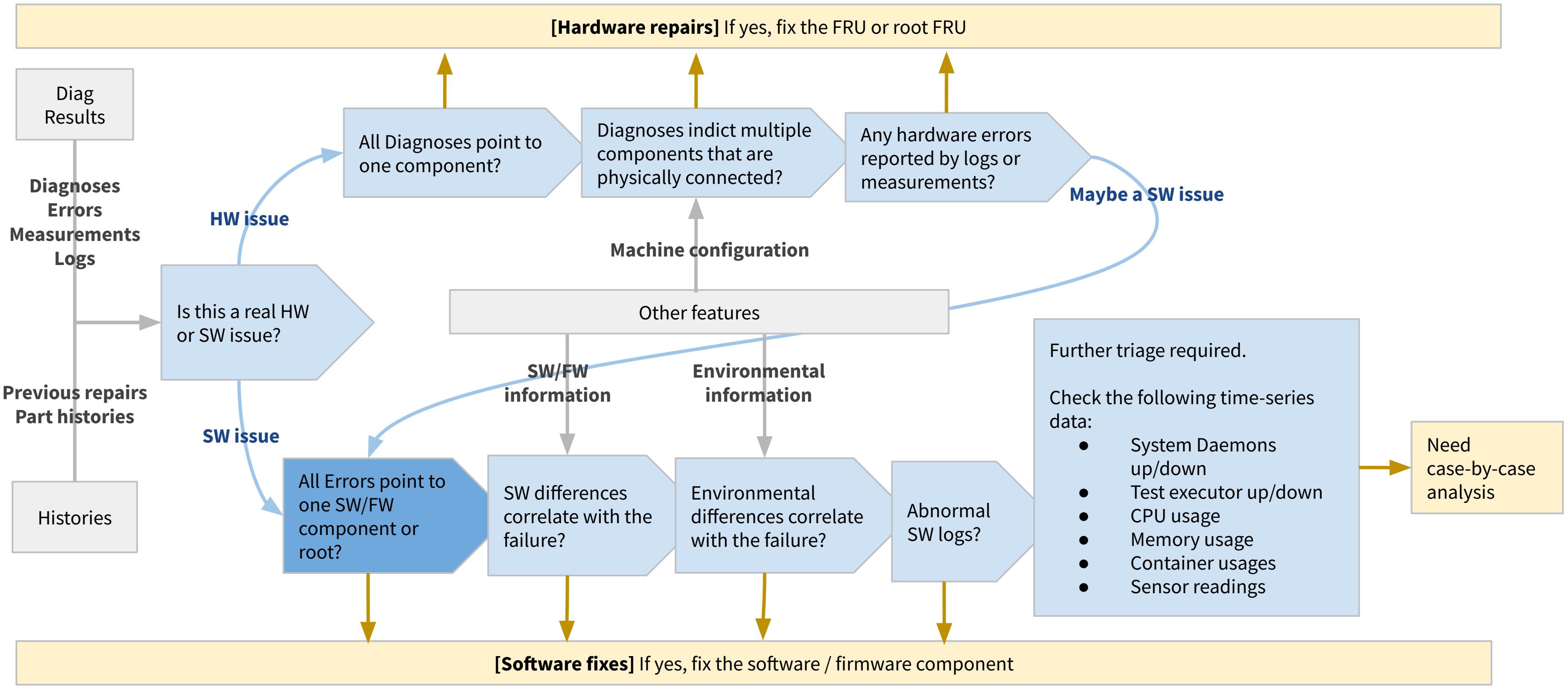
Learning from the 'Human Expert Approach'



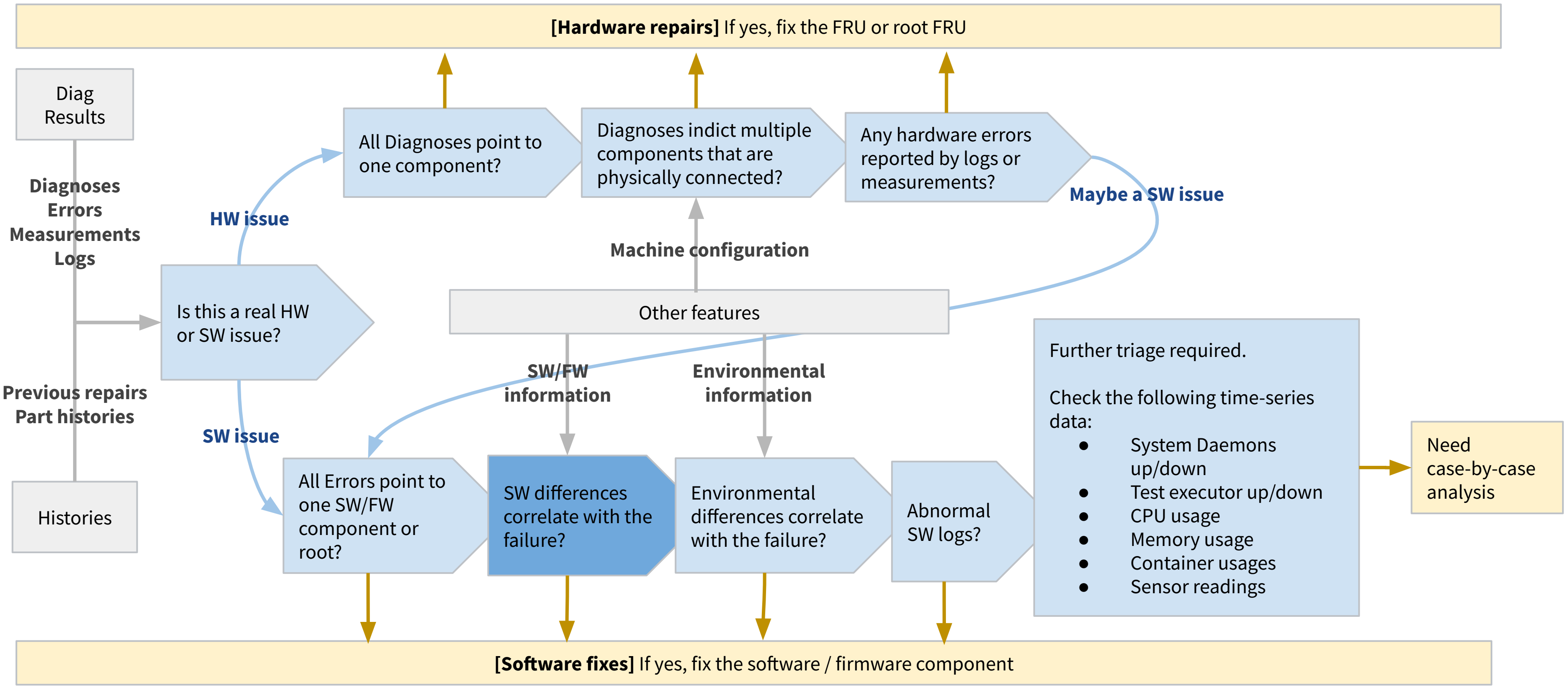
Learning from the 'Human Expert Approach'



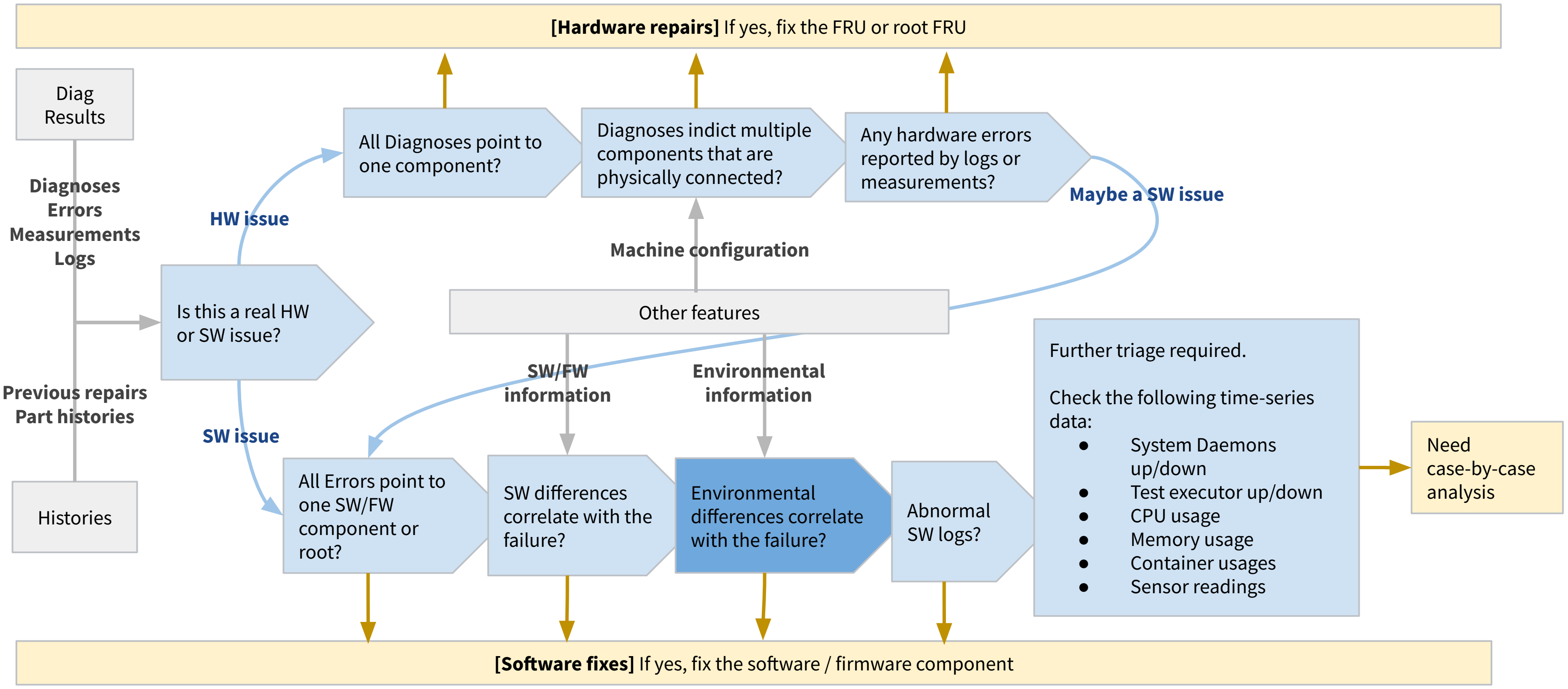
Learning from the 'Human Expert Approach'



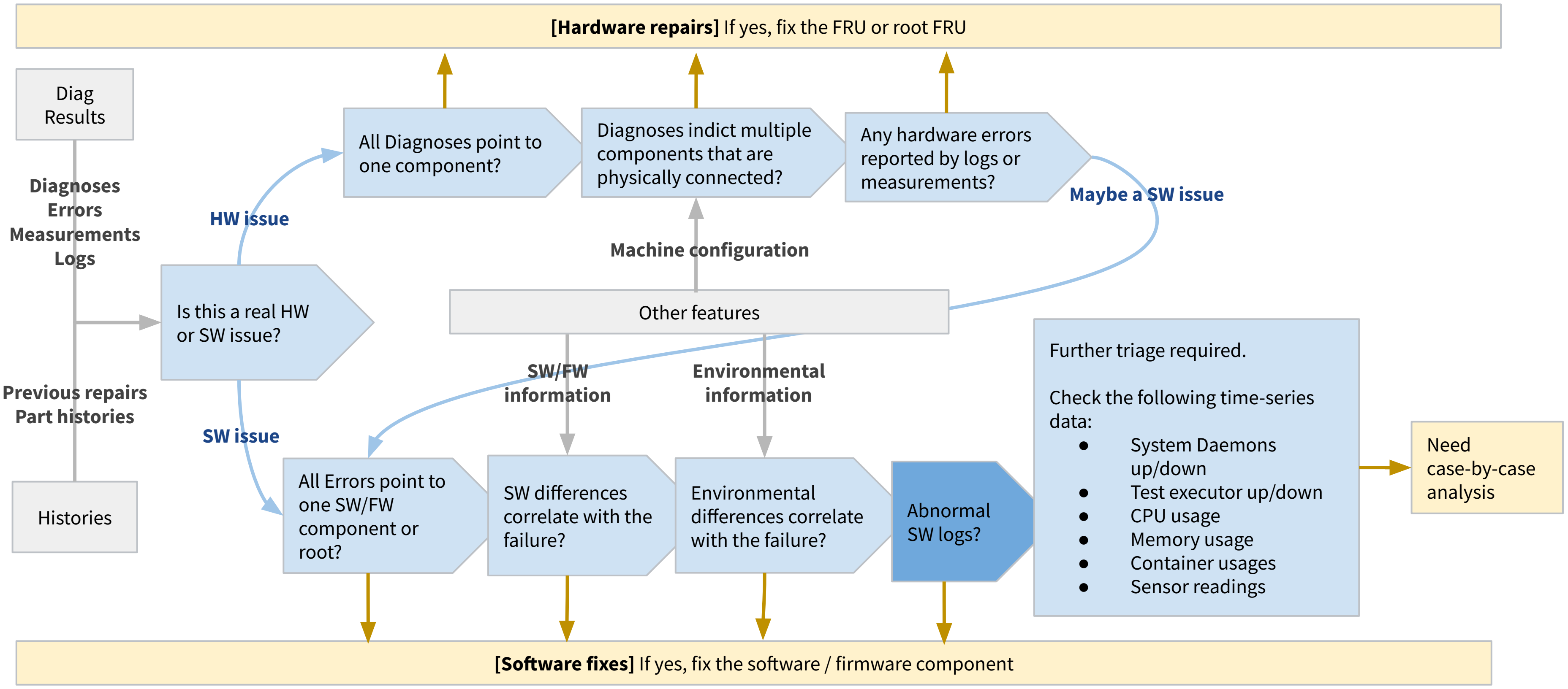
Learning from the 'Human Expert Approach'



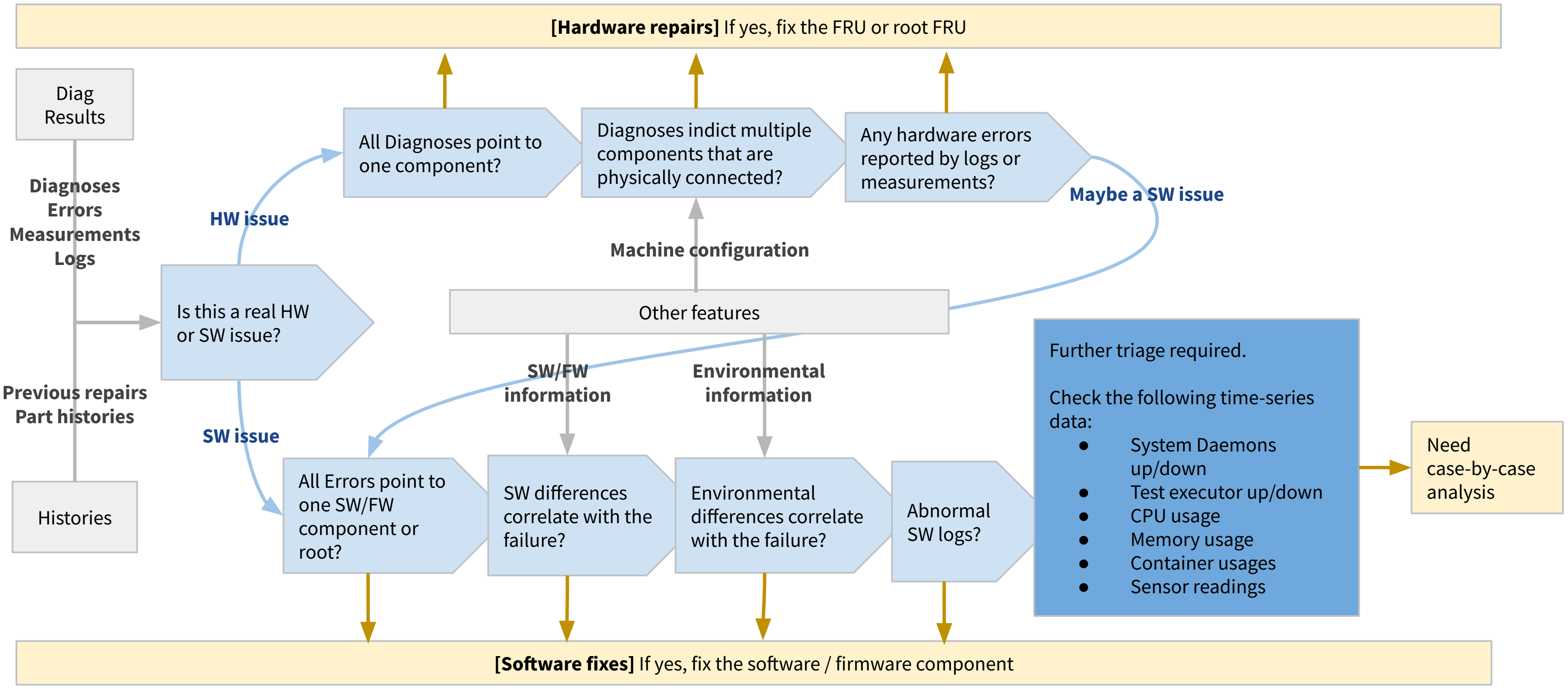
Learning from the 'Human Expert Approach'



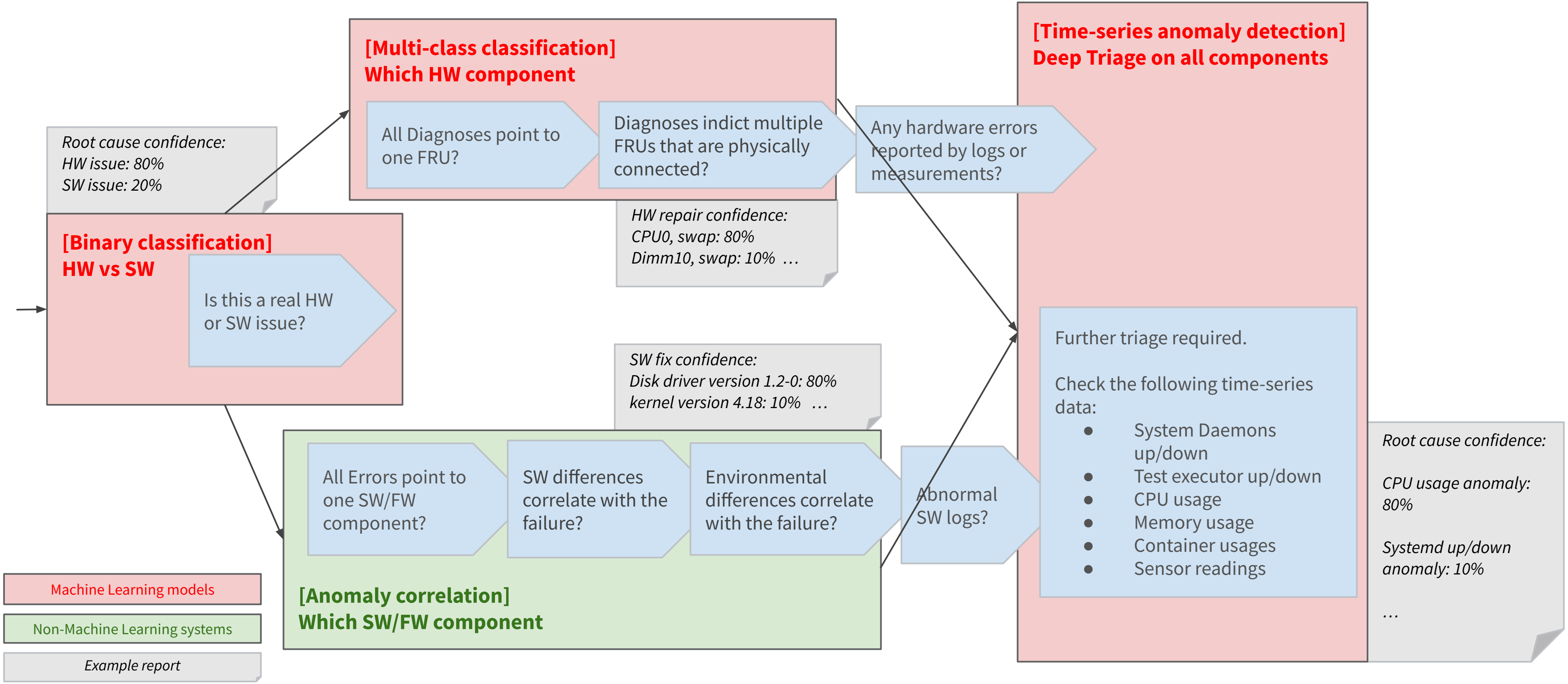
Learning from the 'Human Expert Approach'



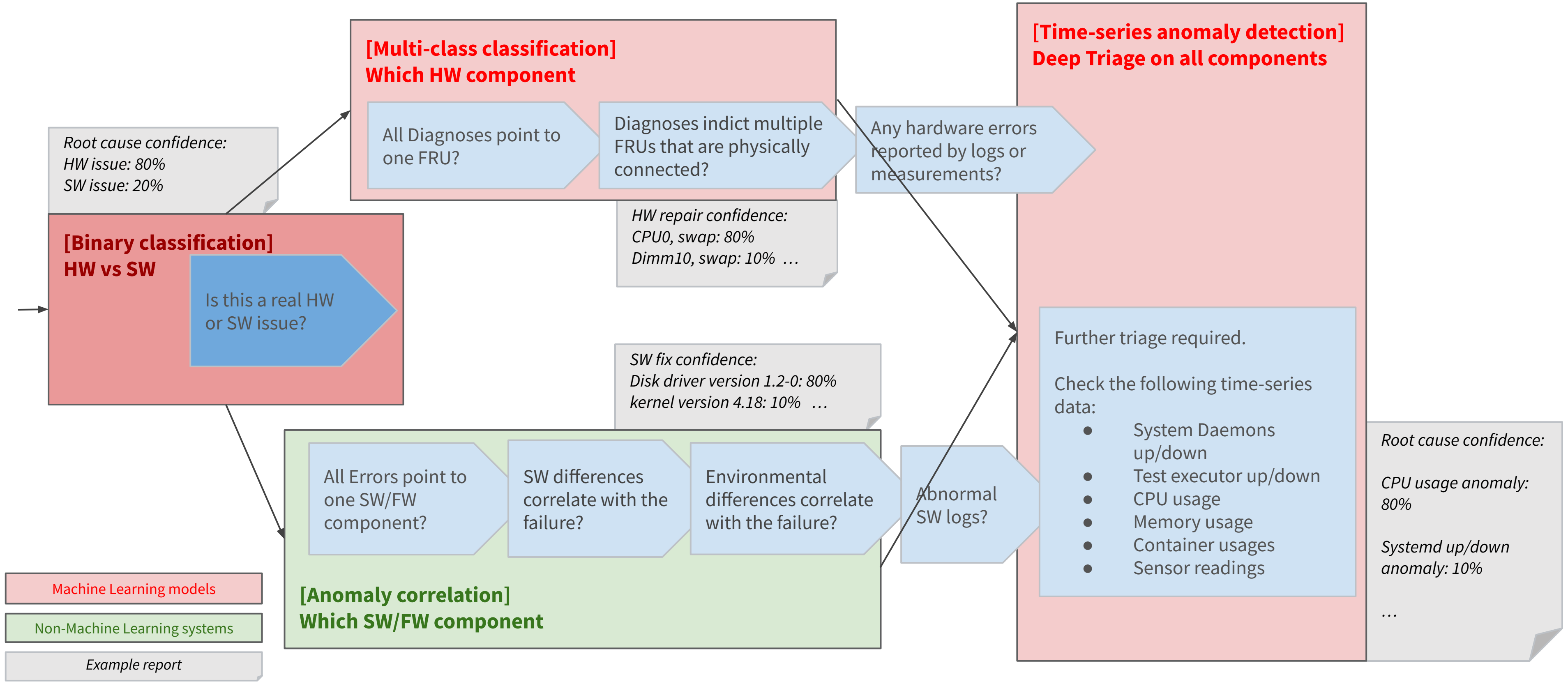
Learning from the 'Human Expert Approach'



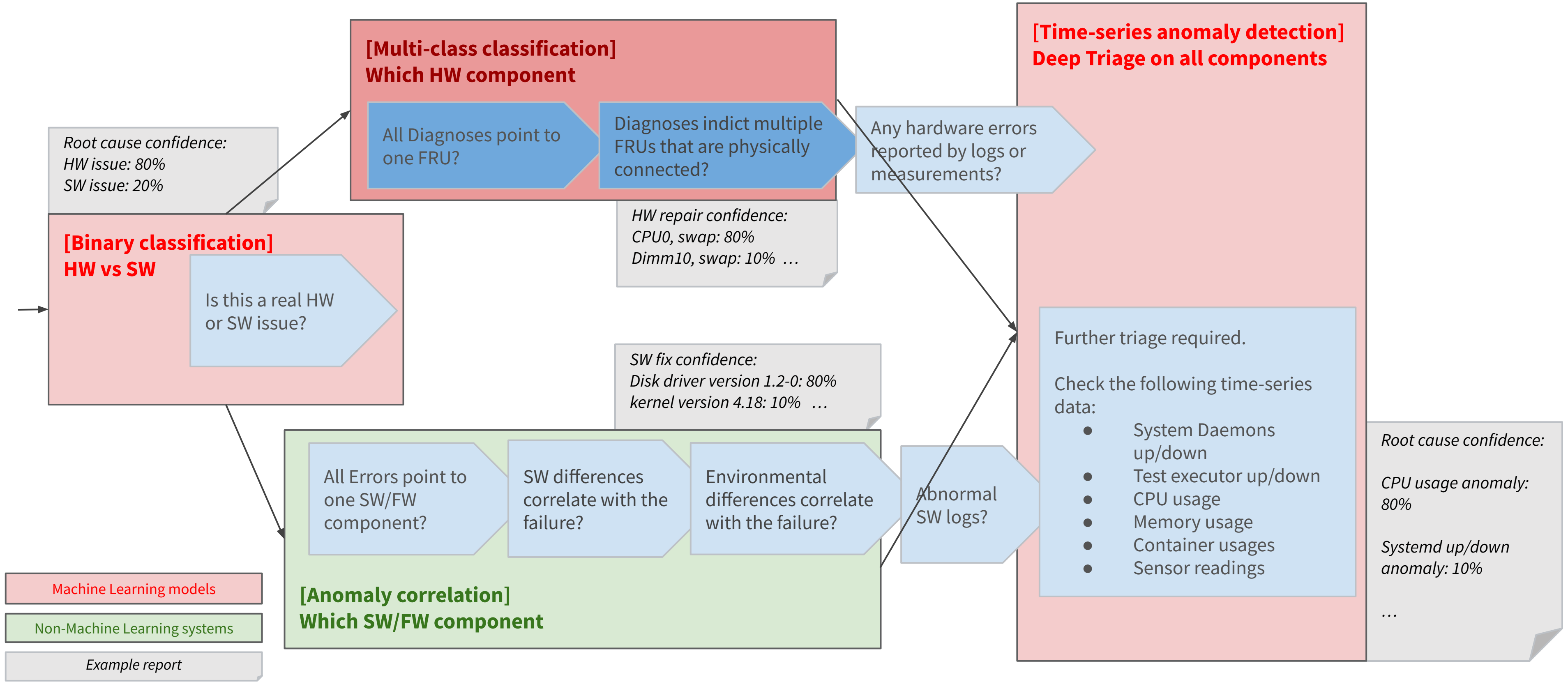
Mapping the 'Human Expert Approach' to the Inference Models for Hardware Diagnosis



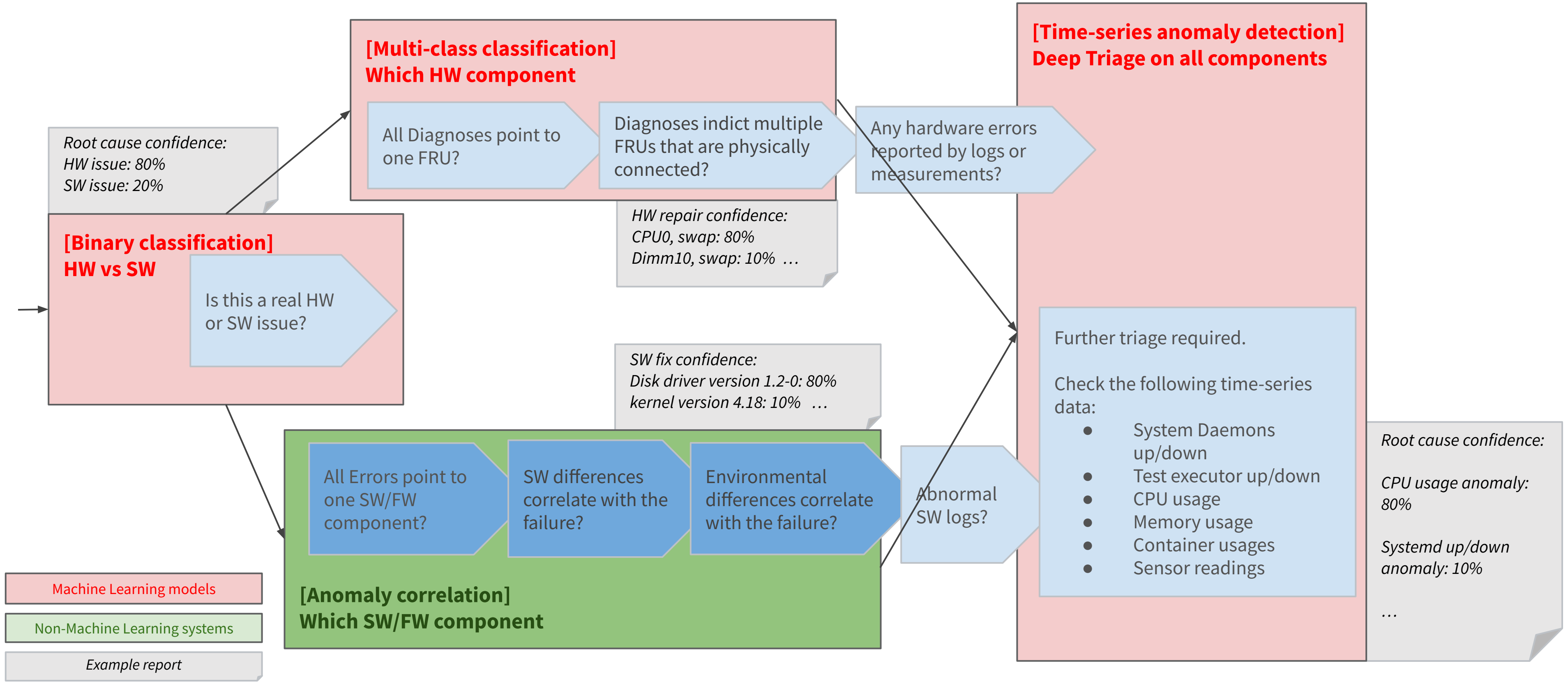
Mapping the 'Human Expert Approach' to the Inference Models for Hardware Diagnosis



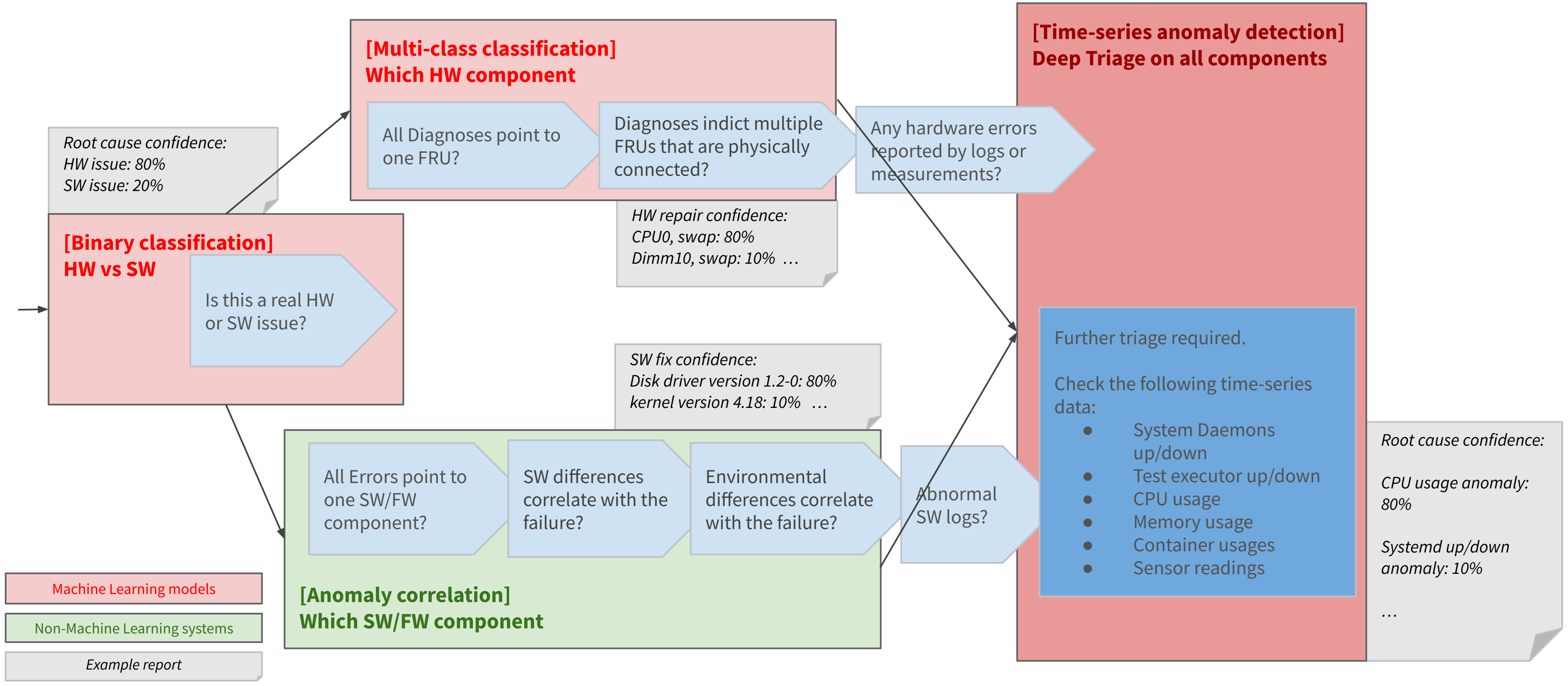
Mapping the 'Human Expert Approach' to the Inference Models for Hardware Diagnosis



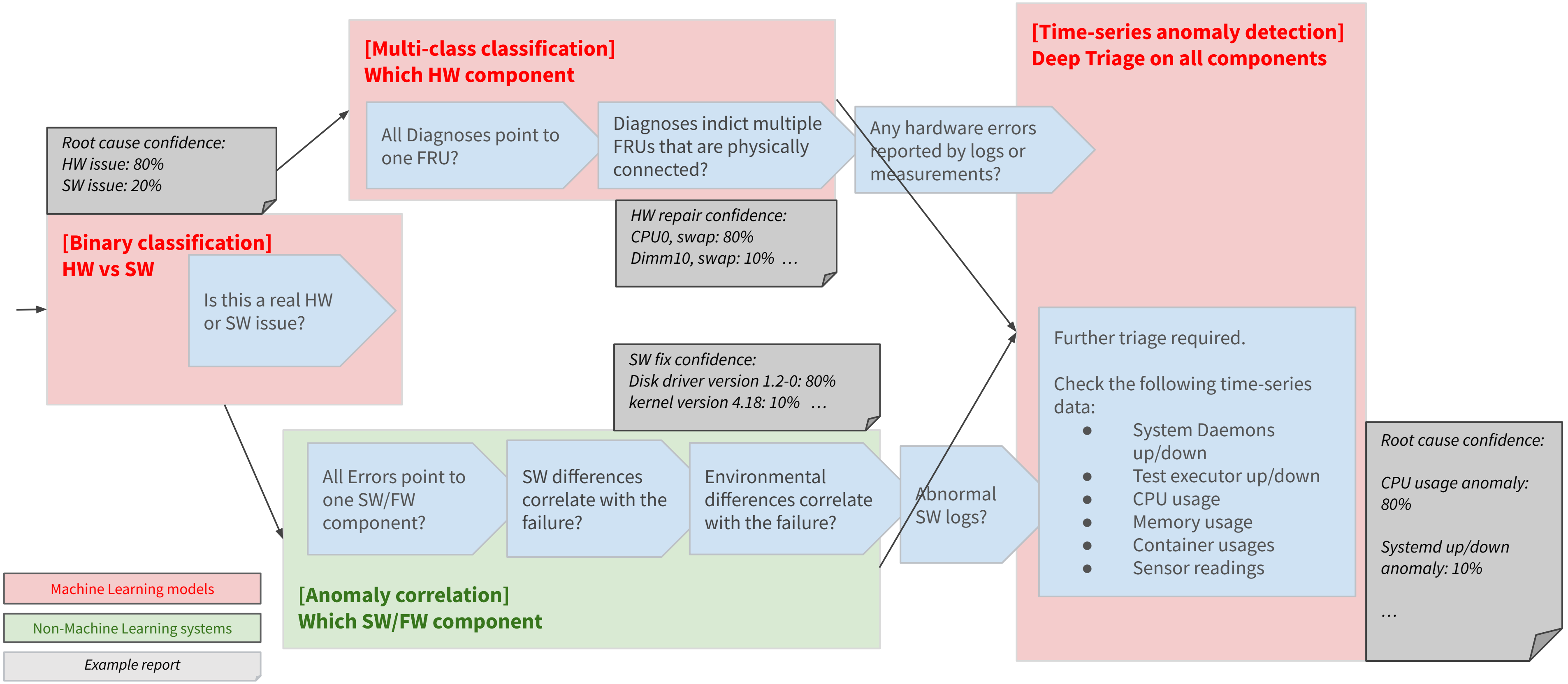
Mapping the 'Human Expert Approach' to the Inference Models for Hardware Diagnosis



Mapping the 'Human Expert Approach' to the Inference Models for Hardware Diagnosis



Mapping the 'Human Expert Approach' to the Inference Models for Hardware Diagnosis



What about the free-format logs?

Root cause confidence:
HW issue: 80%
SW issue: 20%

[Binary classification]
HW vs SW

Is this a real HW or SW issue?

[Multi-class classification]
Which HW component

All Diagnoses point to one FRU?

Diagnoses indict multiple FRUs that are physically connected?

Any hardware errors reported by logs or measurements?

HW repair confidence:
CPU0, swap: 80%
Dimm10, swap: 10% …

SW fix confidence:
Disk driver version 1.2-0: 80%
kernel version 4.18: 10% …

All Errors point to one SW/FW component?

SW differences correlate with the failure?

Environmental differences correlate with the failure?

Abnormal SW logs?

[Anomaly correlation]
Which SW/FW component

[Time-series anomaly detection]
Deep Triage on all components

Further triage required.

Check the following time-series data:

- System Daemons up/down
- Test executor up/down
- CPU usage
- Memory usage
- Container usages
- Sensor readings

Root cause confidence:

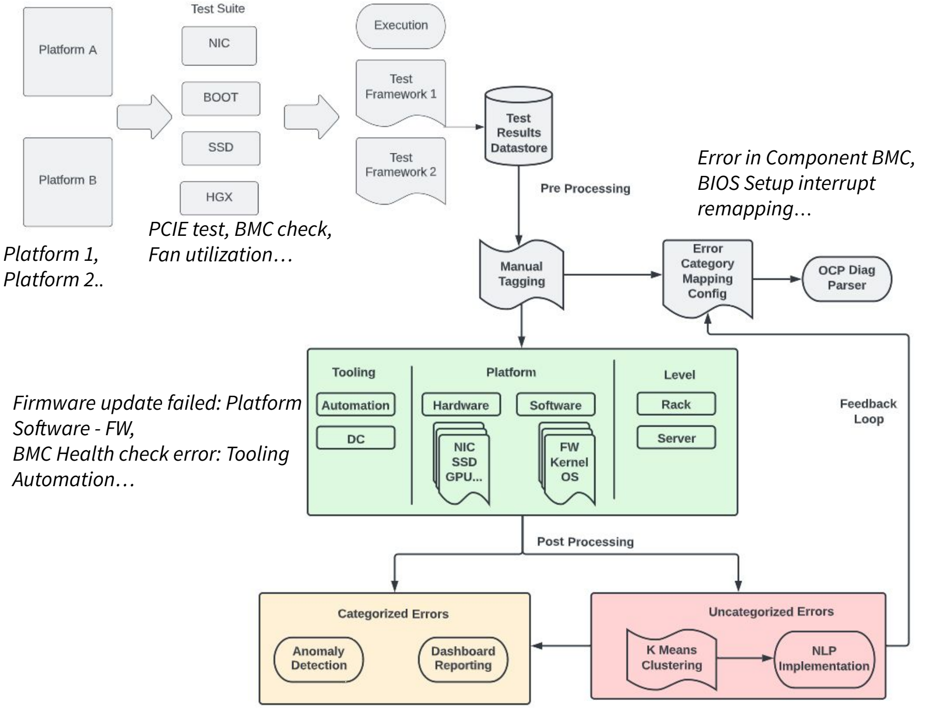
CPU usage anomaly: 80%

Systemd up/down anomaly: 10%

…

End to End Flow to categorize the error logs

- The pilot has been across Compute and Storage platforms, further expanding for GPUs
- Data preparation across different phases helps in building an enriched datasource for categorization
- Merging the Post and Pre processing steps are the essential piece to build the workflow



60% coverage

40% coverage with 92% accuracy

Machine Learning models

Data Mining models

Predictions

NLP Implementation

"UEFI_boot_testUefiBoot Checks if UEFI boot is enabled on a host. Then provision server through PXE boot. Provision using Cyborg tool. Refer to Cyborg class for usage….."

Tokenization

'UEFI_boot_testUefiBoot', 'Checks', 'if', 'UEFI', 'boot', 'is', 'enabled', 'on', 'a', 'host', '.', 'Then', 'provision', 'server', 'through', 'PXE', 'boot', '.', 'Provision', 'using', 'Cyborg', 'tool', '.',..

Stemming

```
['uefi_boot_testuefiboot',
 'check',
 'uefi',
 'boot',
 'enabl',
 'host.',
 'then',
```

Stop Word Removal

'boot', 'enabl', 'host.', 'then', 'provi', 'server'

'boot', 'enabl', 'host.', 'provi', 'server'

Lemmatization

Lemma for 'parameters' is 'parameter'
Lemma for 'arguments' is 'argument'

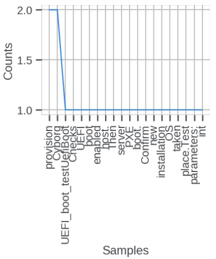
POS Tagging & Filtering

```
[('UEFI_boot_testUefiBoot', 'NN'),
 ('Checks', 'NNS'),
 ('if', 'IN'),
 ('UEFI', 'NNP'),
 ('boot', 'NN'),
 ('is', 'VBZ'),
 ('enabled', 'VBN'),
 ('on', 'IN'),
 ('a', 'DT'),
 ('host.', 'NN'),
```

'UEFI_boot_testUefiBoot', 'Checks', 'UEFI', 'boot', 'enabled', 'host.', 'Then', 'provision', 'server', 'PXE', 'boot.'

Count Word Frequency

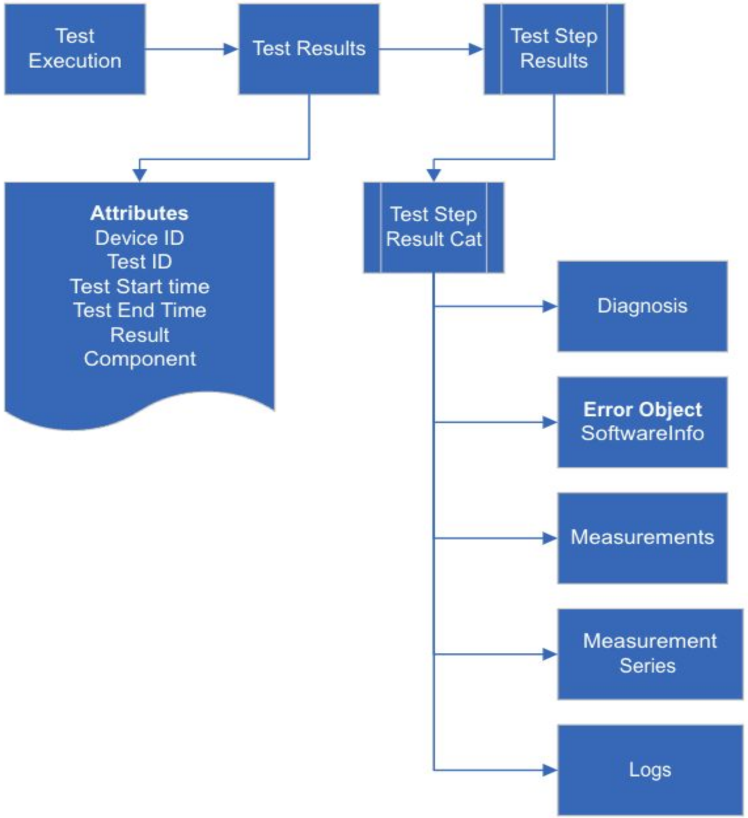
```
UEFI_boot_testUefiBoot:1
Checks:1
UEFI:1
boot:1
enabled:1
host.:1
Then:1
provision:2
server:1
PXE:1
```



Frequency Distribution

OCP Diag Parser

After we cluster the logs to a specific root cause, we want to structure the raw text to the standard OCPDiag structure so they can be captured by other automatic models.



OCP Diagnostics and Inference Repair Models Impact

- Provides continuous improvement over heuristic approaches, augments and amplifies the investment made in the repair heuristics.
- Once the model is developed, operational efficiency improvements "come for free" and are easily A/B tested for effectiveness.
- Model performance data provides a "closed loop" for test quality. How are diagnostic changes affecting our ability to root-cause hardware failures?

1.25
Average Debug Attempt Count

27 % reduction
Average Number of Retests over repair rules

15.39 min saved
Average Cycle Time

2070 test hour saved
per build per month

Call to Action

- If you develop diagnostics that you distribute with your hardware, adopt the OCP Diagnostic output specification. It can help make your hardware 'plug and play' for hyperscalers and very large enterprises.
- Hyperscalers have very large fleets to generate enormous quantities of data to help improve your hardware tests. OCP format makes that easy.
- If you are passionate about tackling this problem with us? Join the OCP Test and Validation subgroup for ML driven repair strategies.
- Do you use Open Firmware? Leverage Redfish models for component connectivity in machine management telemetry.

Join us on Slack: http://ocptestandvalidation.slack.com
Get the Bits: https://github.com/opencomputeproject/ocp-diag-core

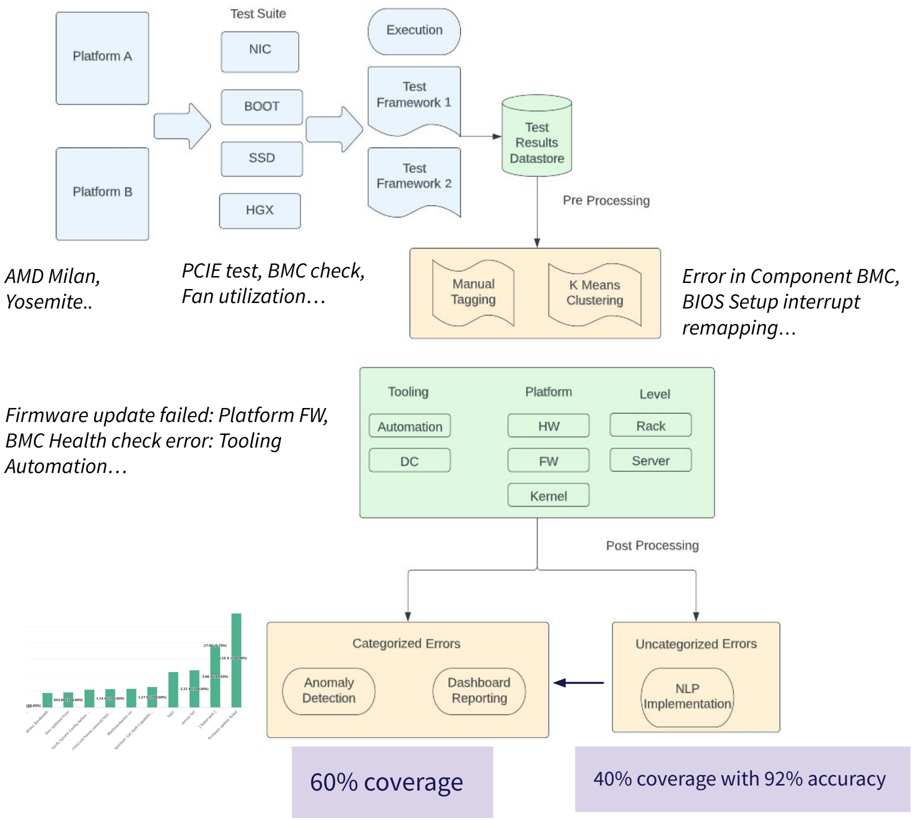
Open Discussion

EMPOWERING OPEN.

OCP GLOBAL SUMMIT

OCTOBER 18-20, 2022
SAN JOSE, CA

End to End Flow to categorize the error logs



Raw logs

Machine Learning models

Data Mining models

Predictions