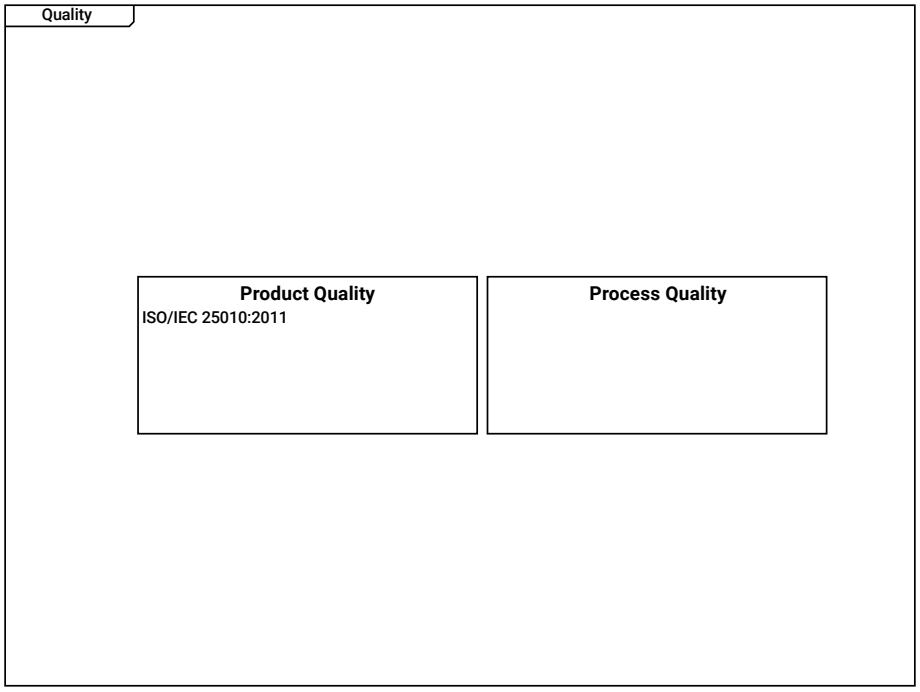
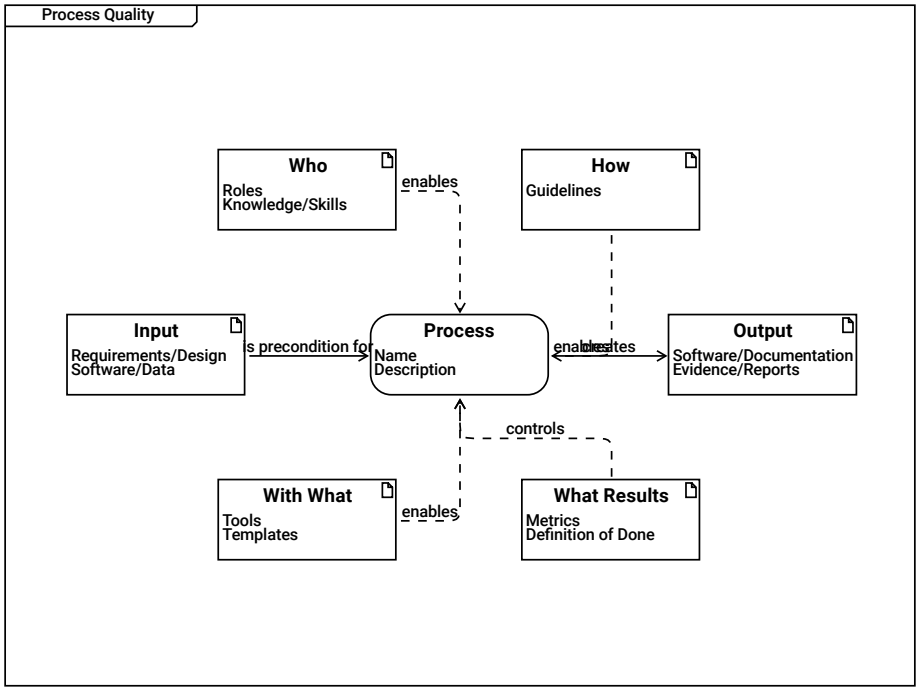
1 Quality Example




```
Quality    [0001D]


Product Quality    [0002C]
  ISO/IEC 25010:2011    [0004F]


Process Quality    [0001C]
```

2 Process Quality

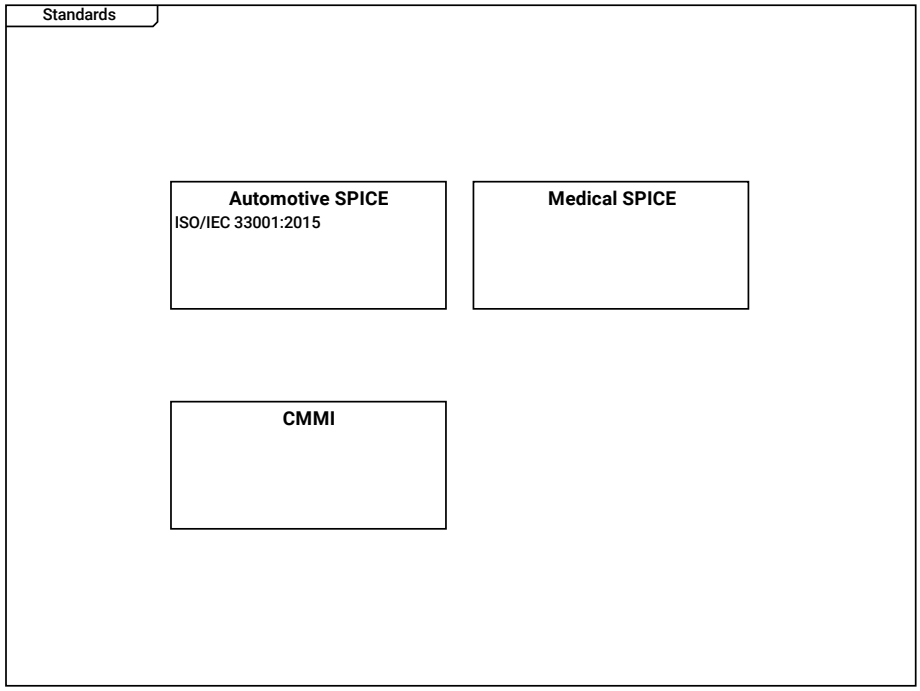



```
Process Quality    [0003D]
| The turtle diagram shows the elements of a process.
```

```
Who    [0010C]
| Roles,
| Skills, Knowledge,
| Trainings
  Roles + Responsibilities    [0048F]
  Knowledge/Skills    [0049F]
  enables --> Process    [0004R]
```

```
How    [0008C]
| Guidelines, Checklists,
| Templates
  Guidelines    [0052F]
  Tutorials    [0065F]
  enables --> Process    [0005R]
```

```
Input    [0011C]
  Requirements/Design    [0057F]
  Software/Data    [0058F]
  is precondition for --> Process    [0001R]


Process    [0005C]
  Name    [0011F]
  Description    [0012F]
  creates --> Output    [0002R]


Output    [0007C]
| Process output,
| Evidence on performed process
  Software/Documentation    [0055F]
  Evidence/Reports    [0056F]


With What    [0006C]
  Tools    [0050F]
  Templates    [0051F]
  enables --> Process    [0003R]


What Results    [0009C]
  Metrics    [0053F]
  Definition of Done    [0054F]
  controls --> Process    [0006R]
```

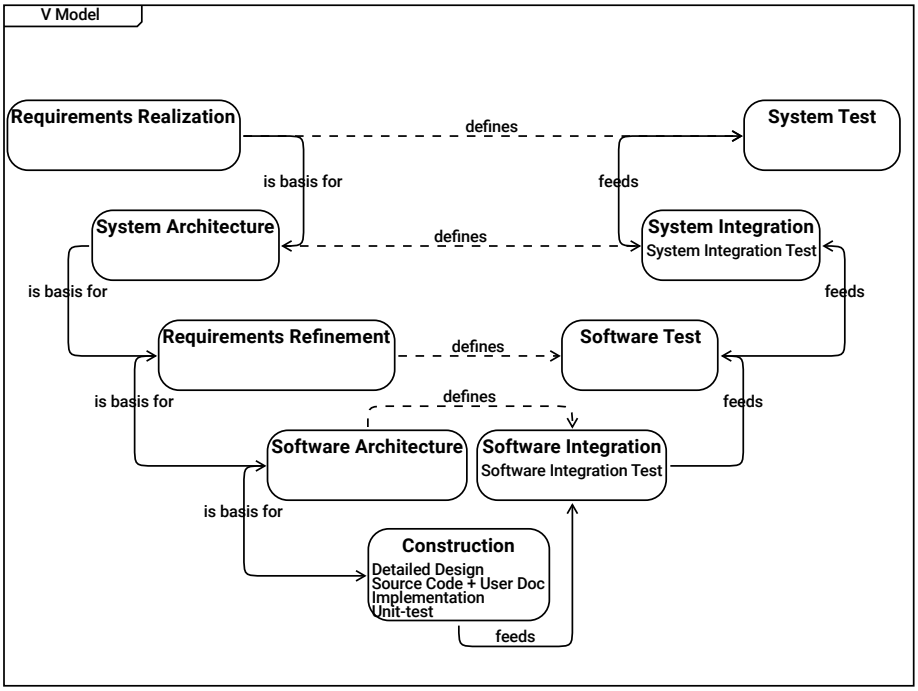


Standards [0006D]

Automotive SPICE [0059C]
ISO/IEC 33001:2015 [0003F]

Medical SPICE [0060C]

CMMI [0058C]



V Model [0009D]

Requirements Realization [0064C]
  is basis for --> System Architecture [0042R]
  defines --> System Test [0050R]

System Test [0072C]

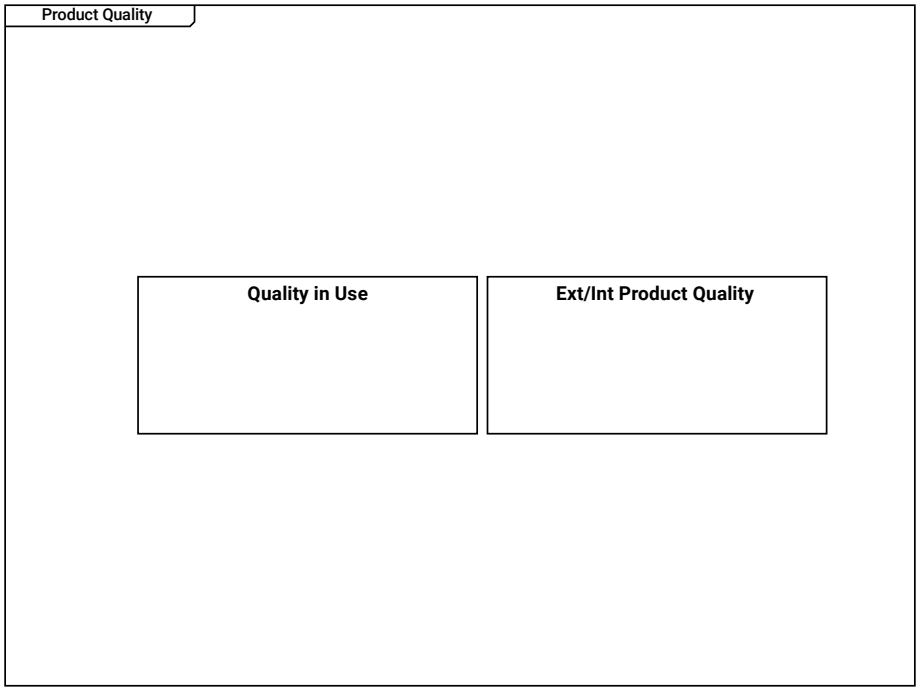
System Architecture [0065C]
  is basis for --> Requirements Refinement [0043R]
  defines --> System Integration [0051R]

System Integration [0071C]
  System Integration Test [0018F]
  feeds --> System Test [0049R]

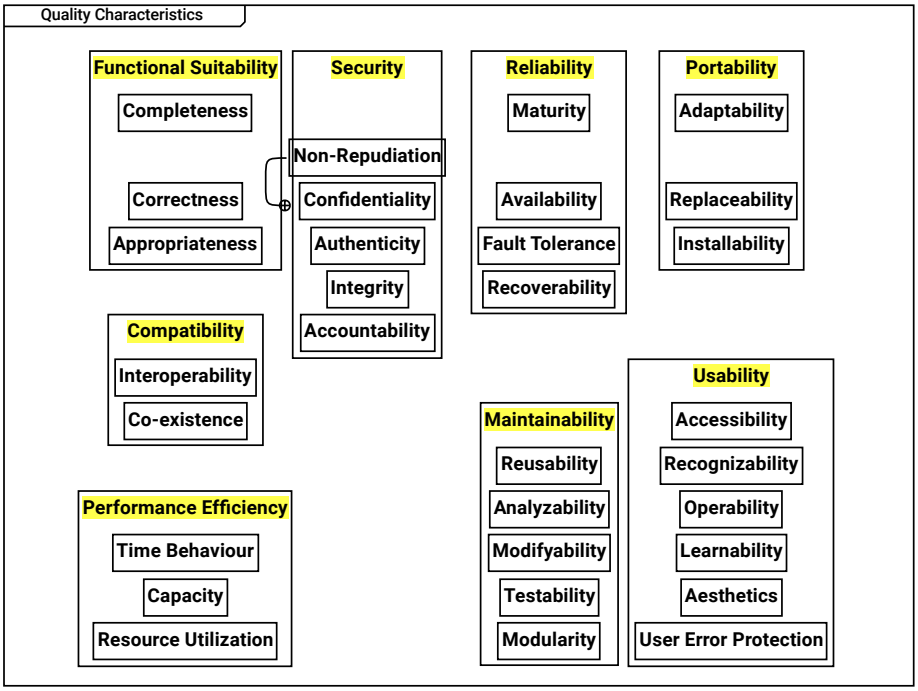
Requirements Refinement [0066C]
  is basis for --> Software Architecture [0044R]

```
  defines --> Software Test    [0052R]


Software Test    [0070C]
  feeds --> System Integration    [0048R]


Software Architecture    [0067C]
  defines --> Software Integration    [0053R]
  | The Software Architecture defines the modules, interfaces and relations needed to integr
  is basis for --> Construction    [0045R]
  | The Software Architecture defines the modules, interfaces and relations needed to create


Software Integration    [0069C]
  Software Integration Test    [0017F]
  feeds --> Software Test    [0047R]


Construction    [0068C]
  Detailed Design    [0015F]
  Source Code + User Doc    [0016F]
  Implementation    [0014F]
  Unit-test    [0013F]
  feeds --> Software Integration    [0046R]
```

3 Product Quality




```
Product Quality    [0002D]


Quality in Use    [0004C]
| Quality in use can be measured when the product is already in use,
| e.g. the percentage of satisfied customers can be determined.


Ext/Int Product Quality     [0003C]
| Product quality are internal and externally visible qualities,
| such as memory consumption or startup timings.
```




```
Quality Characteristics    [0004D]
| according to ISO 25010


Functional Suitability    [0015C]
   --> Completeness    [0056R]
   --> Correctness    [0057R]
   --> Appropriateness    [0058R]


Security    [0018C]
   --> Authenticity    [0082R]
   --> Non-Repudiation    [0083R]
   --> Accountability    [0084R]
   --> Integrity    [0085R]
   --> Confidentiality    [0086R]


Reliability    [0021C]
   --> Maturity    [0062R]
   --> Availability    [0063R]
   --> Fault Tolerance    [0064R]
   --> Recoverability    [0065R]
```

```
Portability    [0020C]
   --> Adaptability    [0068R]
   --> Installability    [0069R]
   --> Replaceability    [0070R]


Completeness    [0016C]


Non-Repudiation    [0038C]


Maturity    [0035C]


Adaptability    [0048C]


Correctness    [0014C]


Confidentiality    [0039C]


Availability    [0034C]


Replaceability    [0050C]


Appropriateness    [0013C]


Authenticity    [0042C]


Fault Tolerance    [0036C]


Installability    [0049C]


Integrity    [0040C]
```

```
Recoverability    [0037C]


Compatibility    [0022C]
   --> Co-existence    [0066R]
   --> Interoperability    [0067R]


Accountability    [0041C]


Interoperability    [0028C]


Usability    [0017C]
   --> Recognizability    [0071R]
   --> Learnability    [0072R]
   --> Operability    [0073R]
   --> User Error Protection    [0074R]
   --> Aesthetics    [0075R]
   --> Accessibility    [0076R]


Co-existence    [0027C]


Maintainability    [0012C]
   --> Testability    [0077R]
   --> Modifyability    [0078R]
   --> Analyzability    [0079R]
   --> Reusability    [0080R]
   --> Modularity    [0081R]


Accessibility    [0029C]


Reusability    [0044C]


Recognizability    [0030C]


Performance Efficiency    [0023C]
   --> Time Behaviour    [0059R]
```

--> Resource Utilization [0060R]
--> Capacity [0061R]

Analyzability [0045C]

Operability [0024C]

Time Behaviour [0025C]

Modifyability [0046C]

Learnability [0032C]

Capacity [0026C]

Testability [0047C]

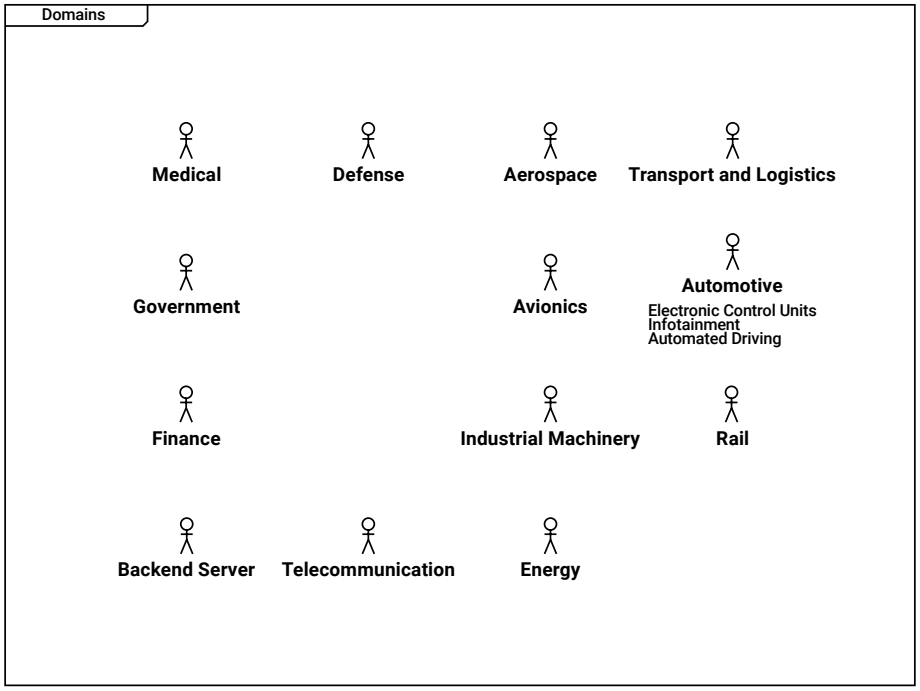
Aesthetics [0031C]

Resource Utilization [0019C]

Modularity [0043C]

User Error Protection [0033C]

3.1 Product Quality Measures



Domains [0005D]

Medical [0056C]

Defense [0053C]

Aerospace [0051C]

Transport and Logistics [0111C]

Government [0112C]

Avionics [0054C]

Automotive [0052C]

Electronic Control Units [0001F]
Infotainment [0002F]
Automated Driving [0047F]

Finance [0105C]

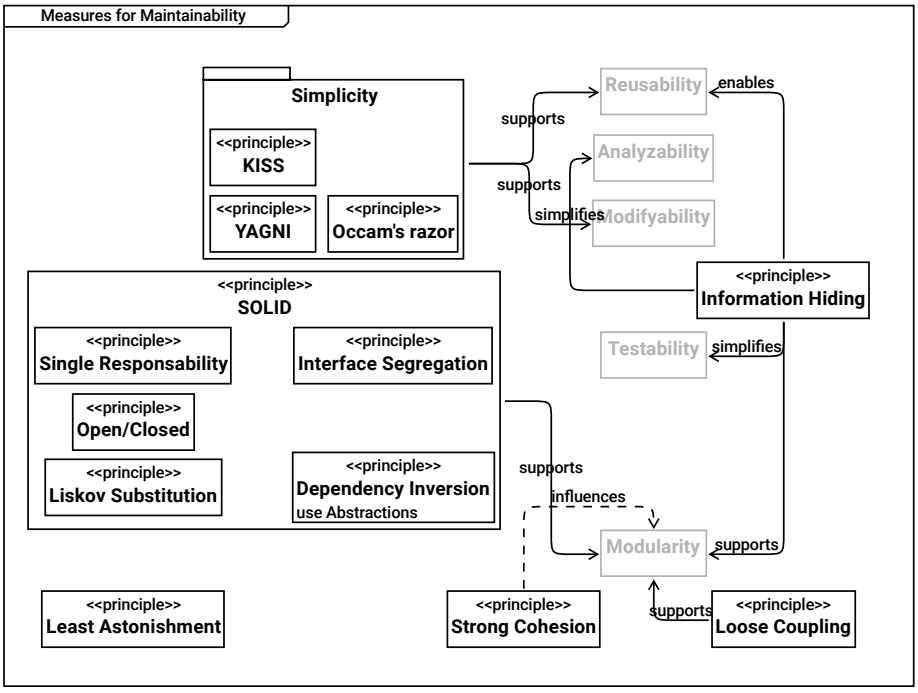
Industrial Machinery [0055C]

Rail [0107C]

Backend Server [0057C]

Telecommunication [0106C]

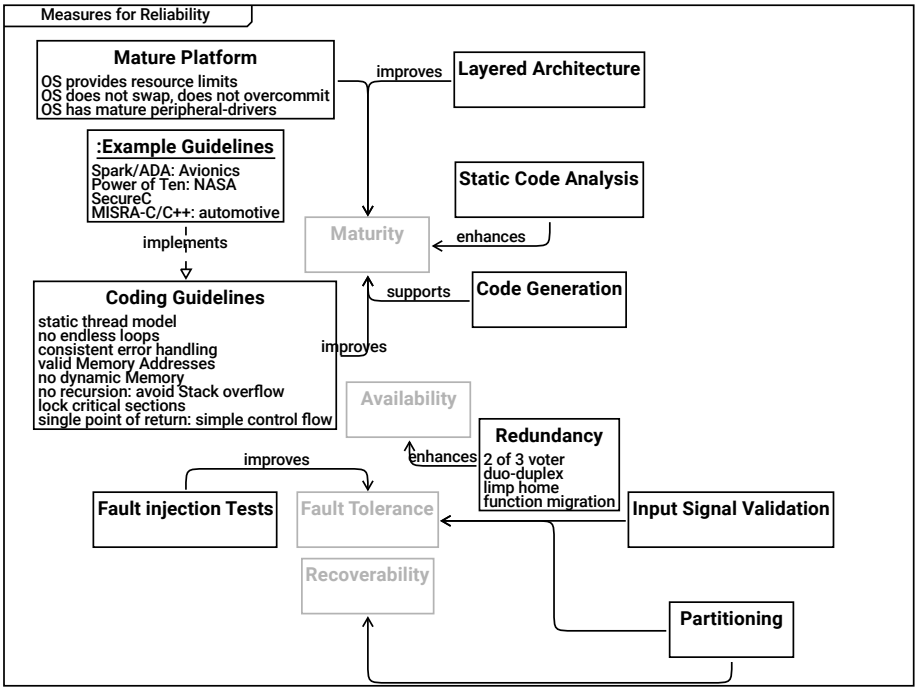
Energy [0110C]



Measures for Maintainability [0007D]

```
Simplicity    [0098C]
   --> KISS    [0106R]
   --> YAGNI    [0107R]
   --> Occam's razor    [0108R]
  supports --> Modifyability    [0109R]
  supports --> Reusability    [0110R]


Reusability    [0044C]


KISS    [0094C]
| Keep it simple and stupid


Analyzability    [0045C]


YAGNI    [0095C]
| You aren't gonna need it


Occam's razor    [0097C]
| Among competing hypotheses, the one with the fewest assumptions should be selected


Modifyability    [0046C]


Information Hiding    [0102C]
| A sofware component shall hide its implementation details and make information accessible
  enables --> Reusability    [0115R]
  supports --> Modularity    [0116R]
  simplifies --> Testability    [0117R]
  simplifies --> Analyzability    [0118R]


Single Responsability    [0089C]
| A software component shall be responsible for one topic only


SOLID    [0096C]
   --> Interface Segregation    [0101R]
   --> Liskov Substitution    [0102R]
   --> Dependency Inversion    [0103R]
```

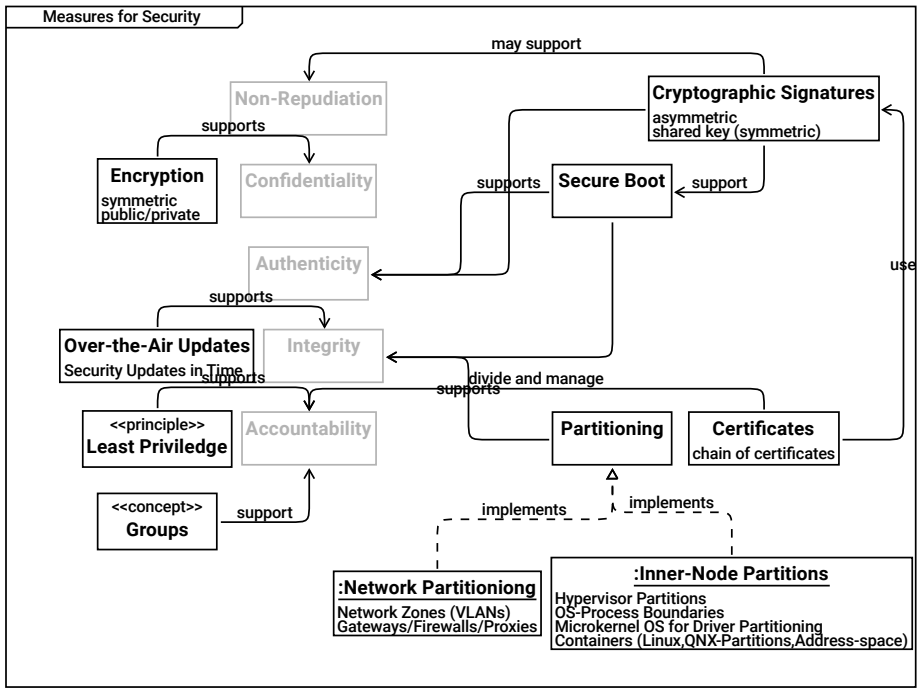
```
   --> Open/Closed    [0104R]
   --> Single Responsability    [0105R]
  supports --> Modularity    [0111R]


Interface Segregation    [0092C]
| Avoid general purpose interfaces, design multiple interfaces specific to the needs of dif


Testability    [0047C]


Open/Closed    [0090C]
| Open for extension, closed for modification


Liskov Substitution    [0091C]
| An implementation of an interface can be replaced by another implementation of the same i


Dependency Inversion    [0093C]
| A software component shall depend on abstractions, not on concrete implementations
  use Abstractions    [0046F]


Modularity    [0043C]


Least Astonishment    [0103C]
| If a reader is astonished when looking at the design, a redesign shall be considered.
| Measure: Conformity of style and concepts


Strong Cohesion    [0104C]
  influences --> Modularity    [0119R]


Loose Coupling    [0101C]
| split an entity that consists of multiple loosely coupled parts
  supports --> Modularity    [0114R]
```




```
Measures for Reliability    [0008D]


Mature Platform    [0109C]
  OS provides resource limits    [0061F]
  OS does not swap, does not overcommit    [0062F]
  OS has mature peripheral-drivers    [0063F]
   --> Maturity    [0124R]


Layered Architecture    [0061C]
  improves --> Maturity    [0039R]


Example Guidelines    [0073C]
  Spark/ADA: Avionics    [0022F]
  Power of Ten: NASA    [0019F]
  SecureC    [0021F]
  MISRA-C/C++: automotive    [0020F]
  implements --> Coding Guidelines    [0054R]


Static Code Analysis    [0086C]
```

```
  enhances --> Maturity    [0099R]


Maturity    [0035C]


Code Generation    [0087C]
| An understandable model and a small code generator
| allow to generate mature software.
  supports --> Maturity    [0100R]


Coding Guidelines    [0062C]
  static thread model    [0010F]
  | Execution threads shall not be started/stopped dynamically
  no endless loops    [0008F]
  | Every loop shall have a counter to ensures that
  | after a predefined maximum value the loop is definitely quit
  consistent error handling    [0009F]
  | Inconsistencies in error handling make
  | bugs in error handling more likely
  valid Memory Addresses    [0007F]
  | Only valid memory addresses may be read/written.
  | E.g. Java solves this by prohibiting pointers,
  | In C/C++, check pointers and array indices before usage
  no dynamic Memory    [0006F]
  | When the program is running,
  | - it must not fail due to
  |   - memory fragmentation (virtual addresses/physical pages)
  |   - out of memory situations
  | - it shall have a defined timing (which new/malloc cannot provide)
  no recursion: avoid Stack overflow    [0005F]
  lock critical sections    [0024F]
  | Always lock critical sections.
  | Exceptions to locking are a nightmare.
  single point of return: simple control flow    [0023F]
  | Simple control flow is key to understandable code
  improves --> Maturity    [0040R]


Availability    [0034C]


Redundancy    [0074C]
  2 of 3 voter    [0025F]
  duo-duplex    [0026F]
```

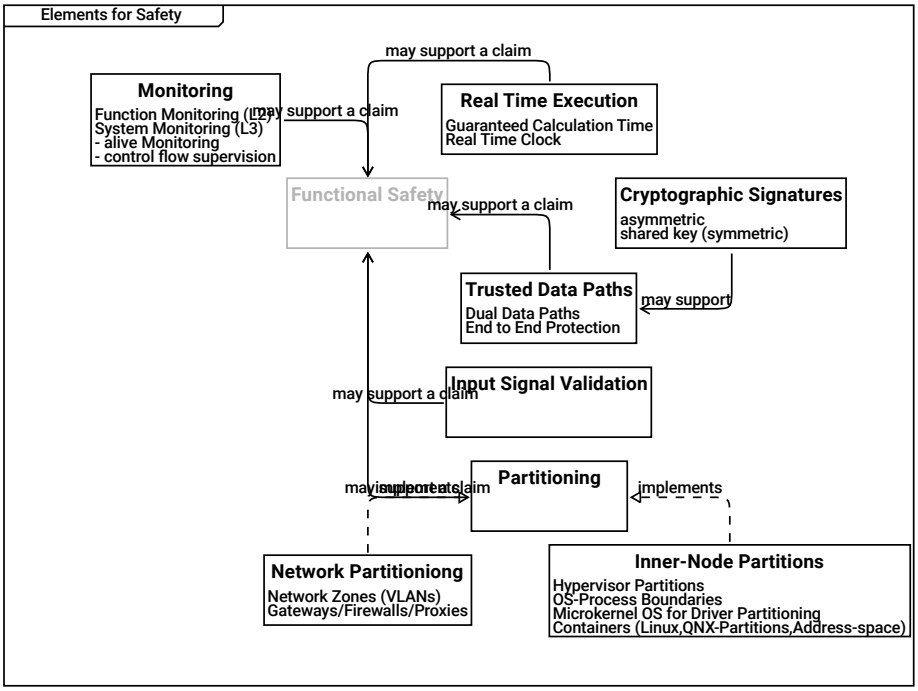
```
  limp home    [0027F]
  function migration    [0028F]
  enhances --> Availability    [0055R]


Fault injection Tests    [0063C]
  improves --> Fault Tolerance    [0041R]


Fault Tolerance    [0036C]


Input Signal Validation    [0083C]
   --> Fault Tolerance    [0128R]


Recoverability    [0037C]


Partitioning    [0075C]
   --> Fault Tolerance    [0129R]
   --> Recoverability    [0130R]
```




```
Measures for Security    [0010D]
```

```
| Functional safety and security are different goals
| but have common mechanisms to support these.
|
| The diagram is not meant to be complete,
| it just shows that technical mechanisms support quality goals.


Non-Repudiation    [0038C]


Cryptographic Signatures    [0079C]
  asymmetric    [0038F]
  shared key (symmetric)    [0039F]
  supports --> Authenticity    [0091R]
  may support --> Non-Repudiation    [0120R]
  support --> Secure Boot    [0123R]


Encryption    [0080C]
  symmetric    [0036F]
  public/private    [0037F]
  supports --> Confidentiality    [0092R]


Confidentiality    [0039C]


Secure Boot    [0108C]
   --> Integrity    [0121R]
   --> Authenticity    [0122R]


Authenticity    [0042C]


Over-the-Air Updates    [0078C]
  Security Updates in Time    [0035F]
  supports --> Integrity    [0090R]


Integrity    [0040C]


Least Priviledge    [0099C]
| Entities shall have only the access rights they need for their purpose
  supports --> Accountability    [0112R]
```

```
Accountability    [0041C]


Partitioning    [0075C]
  supports --> Integrity    [0089R]


Certificates    [0114C]
  chain of certificates    [0064F]
  use --> Cryptographic Signatures    [0126R]
  divide and manage --> Accountability    [0127R]


Groups    [0113C]
| Grouping Clients/Actors helps
| Grouping Services
| helps in administration of access rights
  support --> Accountability    [0125R]


Network Partitioniong    [0076C]
  Network Zones (VLANs)    [0029F]
  Gateways/Firewalls/Proxies    [0030F]
  implements --> Partitioning    [0087R]


Inner-Node Partitions    [0077C]
  Hypervisor Partitions    [0031F]
  OS-Process Boundaries    [0032F]
  Microkernel OS for Driver Partitioning    [0033F]
  Containers (Linux,QNX-Partitions,Address-space)    [0034F]
  implements --> Partitioning    [0088R]
```




```
Elements for Safety    [0011D]


Monitoring    [0084C]
  Function Monitoring (L2)    [0040F]
  System Monitoring (L3)    [0041F]
  - alive Monitoring    [0059F]
  - control flow supervision    [0060F]
  may support a claim --> Functional Safety    [0093R]


Real Time Execution    [0085C]
  Guaranteed Calculation Time    [0044F]
  Real Time Clock    [0045F]
  may support a claim --> Functional Safety    [0098R]


Functional Safety    [0081C]


Cryptographic Signatures    [0079C]
  asymmetric    [0038F]
  shared key (symmetric)    [0039F]
```

```
  may support --> Trusted Data Paths    [0097R]


Trusted Data Paths    [0082C]
  Dual Data Paths    [0042F]
  End to End Protection    [0043F]
  may support a claim --> Functional Safety    [0094R]


Input Signal Validation    [0083C]
  may support a claim --> Functional Safety    [0095R]


Partitioning    [0075C]
  may support a claim --> Functional Safety    [0096R]


Network Partitioniong    [0076C]
  Network Zones (VLANs)    [0029F]
  Gateways/Firewalls/Proxies    [0030F]
  implements --> Partitioning    [0087R]


Inner-Node Partitions    [0077C]
  Hypervisor Partitions    [0031F]
  OS-Process Boundaries    [0032F]
  Microkernel OS for Driver Partitioning    [0033F]
  Containers (Linux,QNX-Partitions,Address-space)    [0034F]
  implements --> Partitioning    [0088R]
```