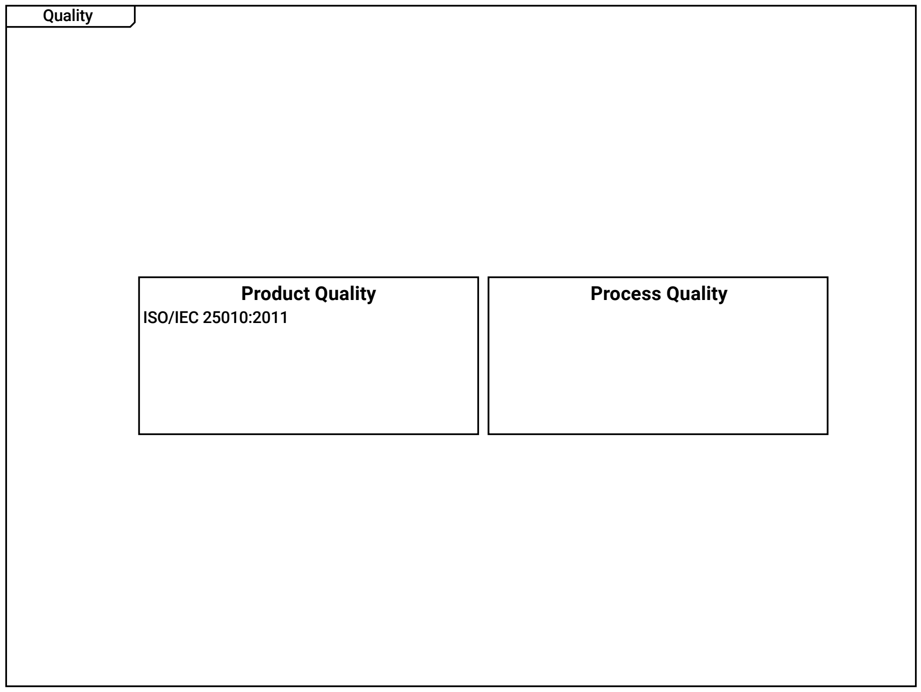
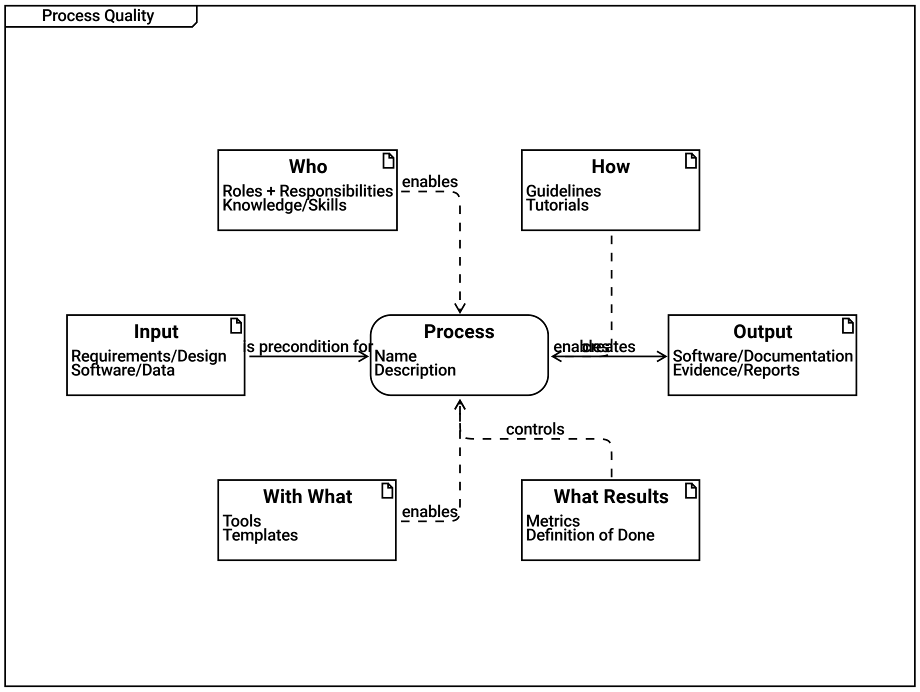
1 Quality Example



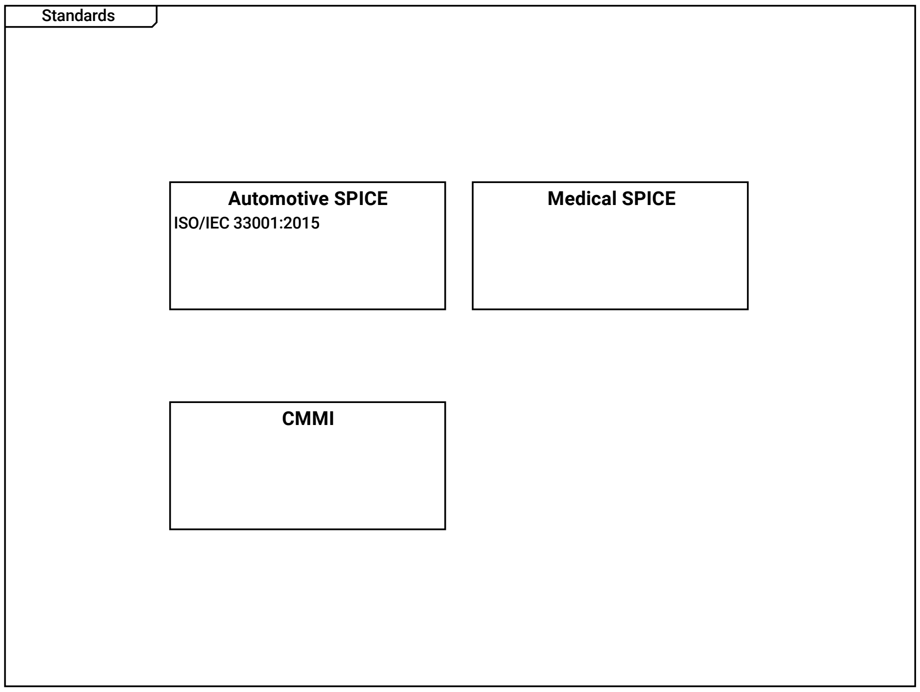
```
Quality    [D0001]
| Quality is a set of characteristics of a product.
| These characteritics can be measured.


Product Quality    [C0002]
| Product quality refers to characteristics that can be measured
| by analyzing the product that was created and that is in use.
  ISO/IEC 25010:2011    [F0004]


Process Quality    [C0001]
| Process quality can be measured by analyzing the reports and other workproducts
| that have  been created during different phases at engineering and during operation.
```

2 Process Quality

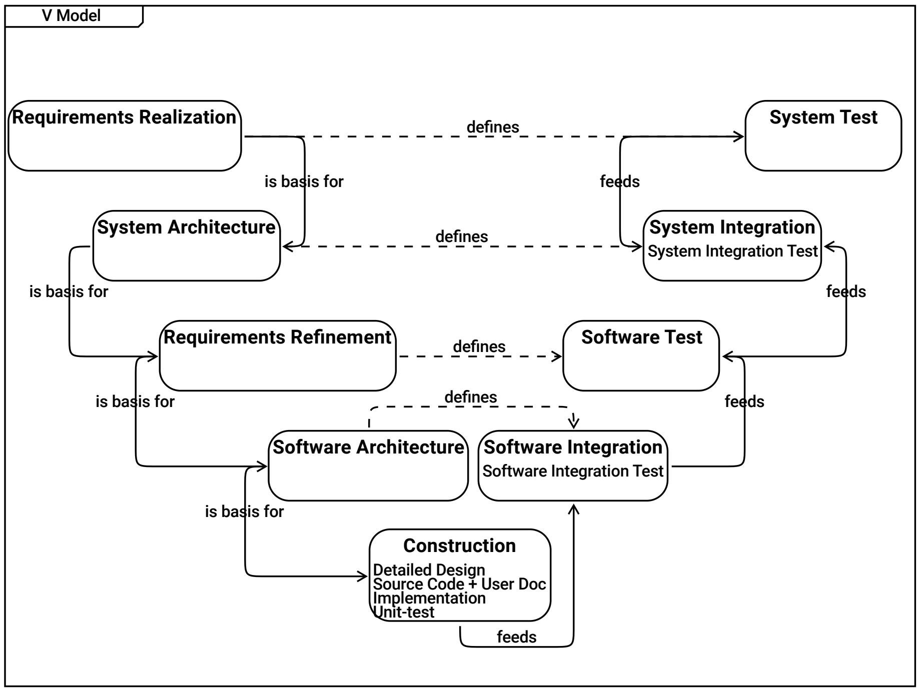


```
Process Quality    [D0003]
| The turtle diagram shows the elements of a process.


Who    [C0010]
| Roles,
| Skills, Knowledge,
| Trainings
  Roles + Responsibilities    [F0048]
  Knowledge/Skills    [F0049]
  enables --> Process    [R0004]


How    [C0008]
| Guidelines, Checklists,
| Templates
  Guidelines    [F0052]
  Tutorials    [F0065]
  enables --> Process    [R0005]
```

```
Input    [C0011]
  Requirements/Design    [F0057]
  Software/Data    [F0058]
  is precondition for --> Process    [R0001]


Process    [C0005]
  Name    [F0011]
  Description    [F0012]
  creates --> Output    [R0002]


Output    [C0007]
| Process output,
| Evidence on performed process
  Software/Documentation    [F0055]
  Evidence/Reports    [F0056]


With What    [C0006]
  Tools    [F0050]
  Templates    [F0051]
  enables --> Process    [R0003]


What Results    [C0009]
  Metrics    [F0053]
  Definition of Done    [F0054]
  controls --> Process    [R0006]
```

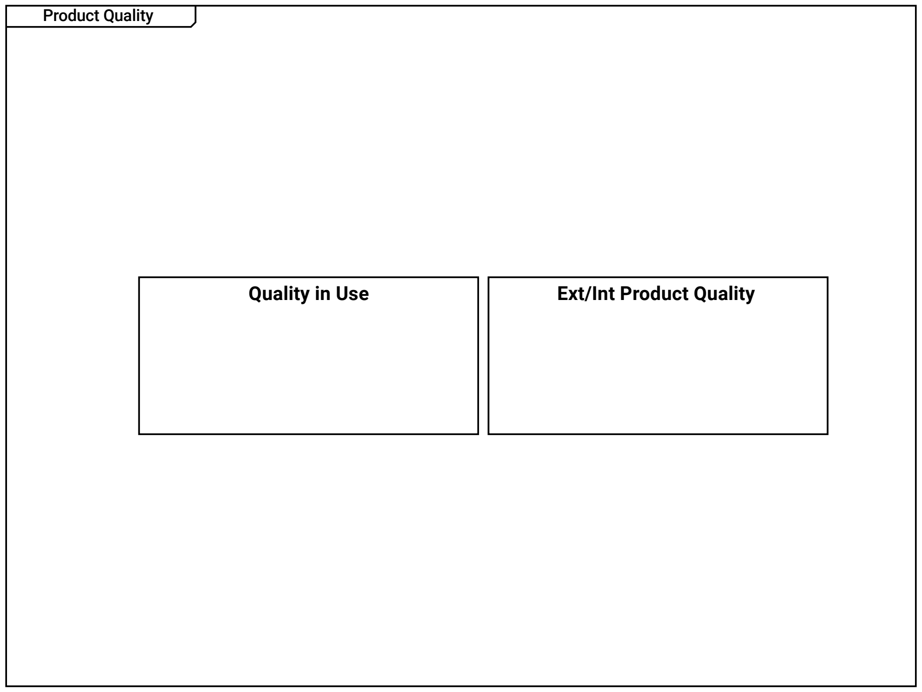


```
Standards    [D0006]


Automotive SPICE    [C0059]
  ISO/IEC 33001:2015    [F0003]


Medical SPICE    [C0060]


CMMI    [C0058]
```

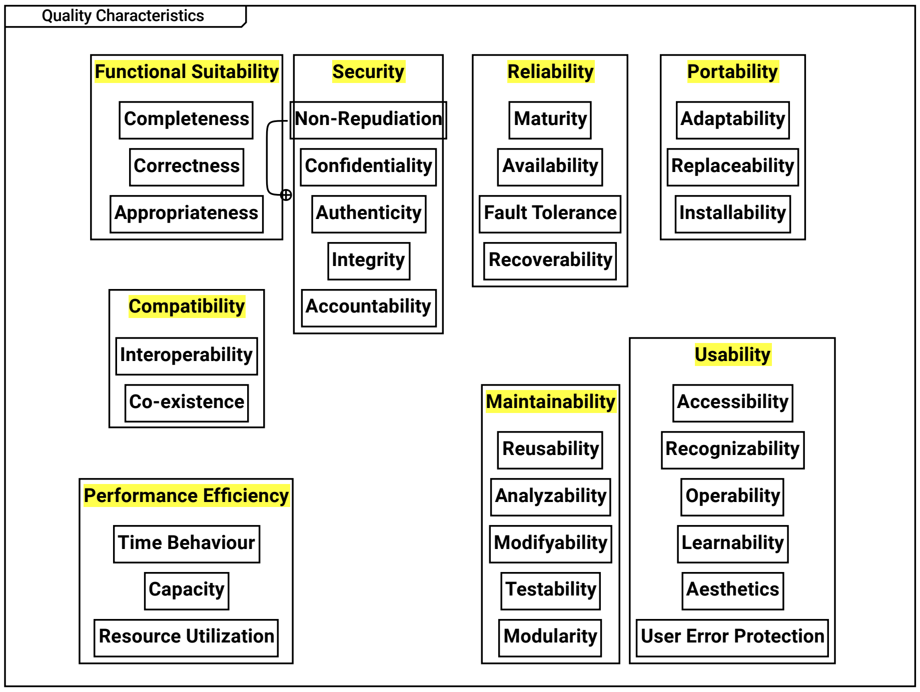


```
V Model    [D0009]


Requirements Realization    [C0064]
  is basis for --> System Architecture    [R0042]
  defines --> System Test    [R0050]


System Test    [C0072]


System Architecture    [C0065]
  is basis for --> Requirements Refinement    [R0043]
  defines --> System Integration    [R0051]


System Integration    [C0071]
  System Integration Test    [F0018]
  feeds --> System Test    [R0049]


Requirements Refinement    [C0066]
  is basis for --> Software Architecture    [R0044]
  defines --> Software Test    [R0052]


Software Test    [C0070]
  feeds --> System Integration    [R0048]


Software Architecture    [C0067]
  defines --> Software Integration    [R0053]
  | The Software Architecture defines the modules, interfaces and relations
  | needed to integrate and test the system.
  is basis for --> Construction    [R0045]
  | The Software Architecture defines the modules, interfaces and relations
  | needed to create the system parts.


Software Integration    [C0069]
  Software Integration Test    [F0017]
  feeds --> Software Test    [R0047]


Construction    [C0068]
  Detailed Design    [F0015]
  Source Code + User Doc    [F0016]
  Implementation    [F0014]
  Unit-test    [F0013]
  feeds --> Software Integration    [R0046]
```

3 Product Quality



```
Product Quality    [D0002]


Quality in Use    [C0004]
| 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     [C0003]
| Product quality are internal and externally visible qualities,
| such as memory consumption or startup timings.
```

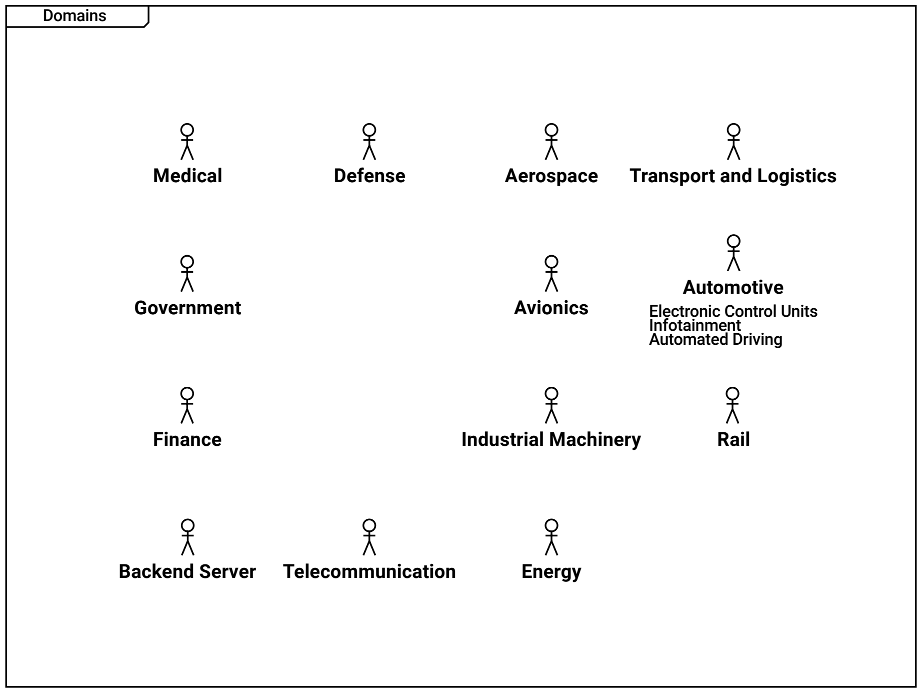


```
Quality Characteristics    [D0004]
| according to ISO 25010


Functional Suitability    [C0015]
  --> Completeness    [R0056]
  --> Correctness    [R0057]
  --> Appropriateness    [R0058]


Security    [C0018]
  --> Authenticity    [R0082]
  --> Non-Repudiation    [R0083]
  --> Accountability    [R0084]
  --> Integrity    [R0085]
  --> Confidentiality    [R0086]


Reliability    [C0021]
  --> Maturity    [R0062]
  --> Availability    [R0063]
  --> Fault Tolerance    [R0064]
  --> Recoverability    [R0065]
```

```
Portability    [C0020]
  --> Adaptability    [R0068]
  --> Installability    [R0069]
  --> Replaceability    [R0070]


Completeness    [C0016]


Non-Repudiation    [C0038]


Maturity    [C0035]


Adaptability    [C0048]


Correctness    [C0014]


Confidentiality    [C0039]


Availability    [C0034]


Replaceability    [C0050]


Appropriateness    [C0013]


Authenticity    [C0042]


Fault Tolerance    [C0036]


Installability    [C0049]


Integrity    [C0040]
```

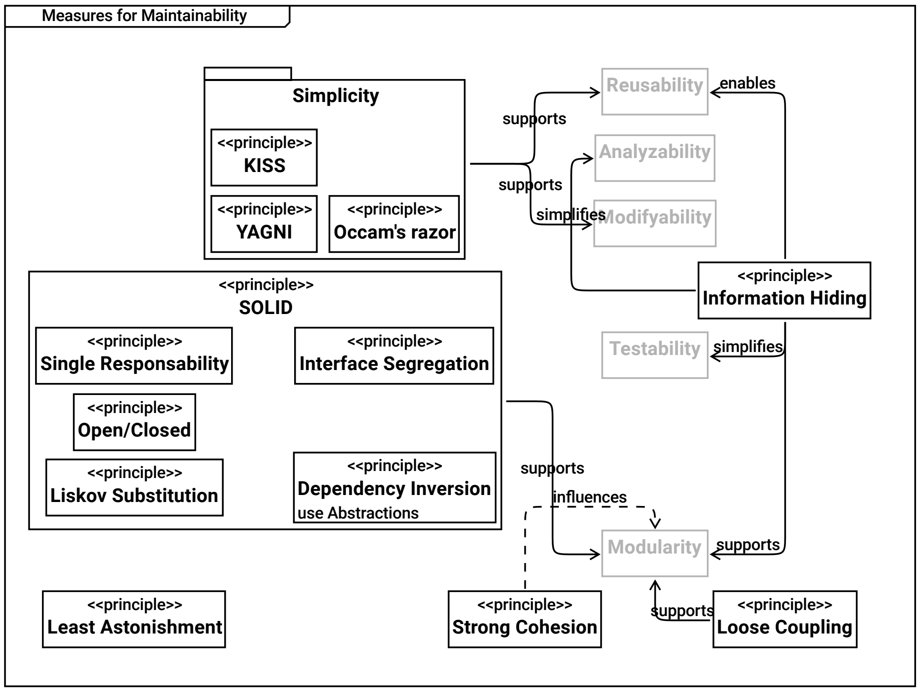
```
Recoverability    [C0037]


Compatibility    [C0022]
  --> Co-existence    [R0066]
  --> Interoperability    [R0067]


Accountability    [C0041]


Interoperability    [C0028]


Usability    [C0017]
  --> Recognizability    [R0071]
  --> Learnability    [R0072]
  --> Operability    [R0073]
  --> User Error Protection    [R0074]
  --> Aesthetics    [R0075]
  --> Accessibility    [R0076]


Co-existence    [C0027]


Maintainability    [C0012]
  --> Testability    [R0077]
  --> Modifyability    [R0078]
  --> Analyzability    [R0079]
  --> Reusability    [R0080]
  --> Modularity    [R0081]


Accessibility    [C0029]


Reusability    [C0044]


Recognizability    [C0030]


Performance Efficiency    [C0023]
  --> Time Behaviour    [R0059]
  --> Resource Utilization    [R0060]
  --> Capacity    [R0061]


Analyzability    [C0045]


Operability    [C0024]


Time Behaviour    [C0025]


Modifyability    [C0046]


Learnability    [C0032]


Capacity    [C0026]


Testability    [C0047]


Aesthetics    [C0031]


Resource Utilization    [C0019]


Modularity    [C0043]


User Error Protection    [C0033]
```

3.1 Product Quality Measures



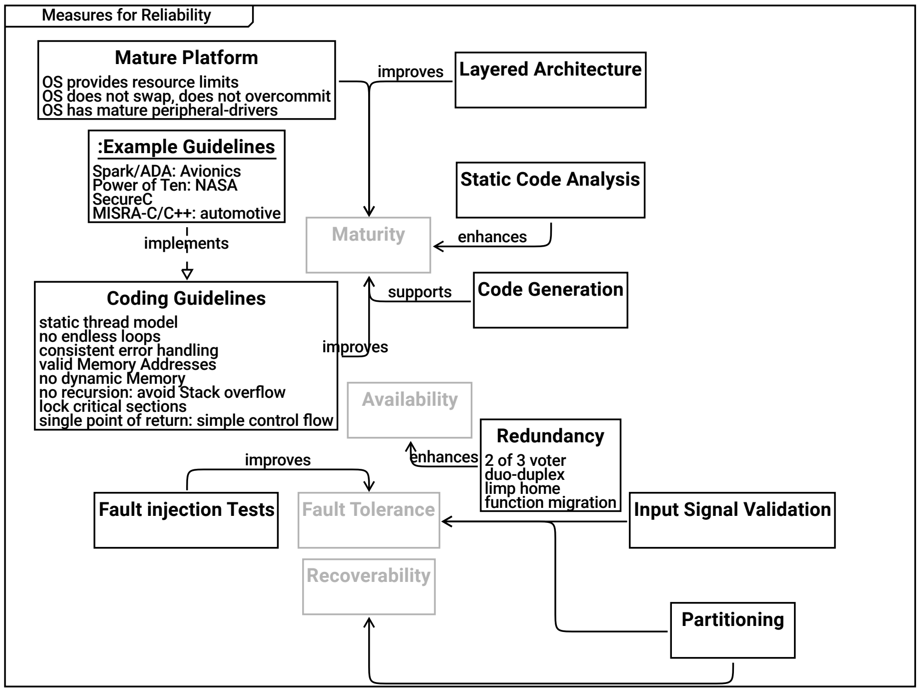
```
Domains    [D0005]
| Measures to improve software quality are sometimes domain-specific.
| Some domains are focusing on tests, some on formal proves,
| some on reaction times till deploying software updates.


Medical    [C0056]


Defense    [C0053]


Aerospace    [C0051]


Transport and Logistics    [C0111]


Government    [C0112]


Avionics    [C0054]
```

```
Automotive    [C0052]
  Electronic Control Units    [F0001]
  Infotainment    [F0002]
  Automated Driving    [F0047]


Finance    [C0105]


Industrial Machinery    [C0055]


Rail    [C0107]


Backend Server    [C0057]


Telecommunication    [C0106]


Energy    [C0110]
```

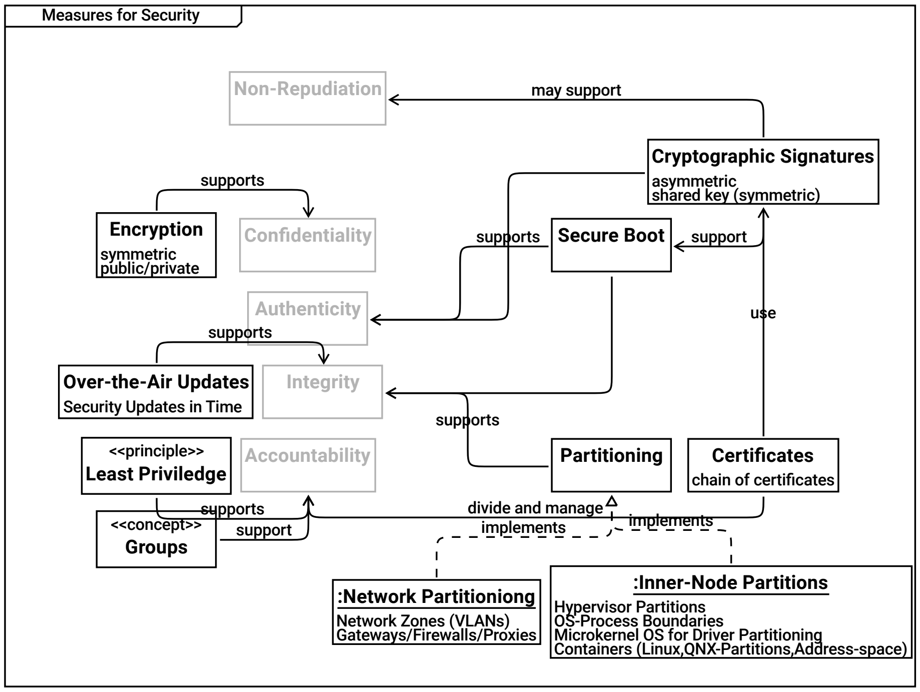

Measures for Maintainability
Simplicity
<<principle>> KISS
<<principle>> YAGNI
<<principle>> Occam's razor
Reusability
Analyzability
Modifyability
enables
supports
supports
simplifies
<<principle>> Information Hiding
<<principle>> SOLID
<<principle>> Single Responsability
<<principle>> Interface Segregation
<<principle>> Open/Closed
<<principle>> Liskov Substitution
<<principle>> Dependency Inversion
use Abstractions
Testability
simplifies
supports
influences
Modularity
supports
supports
<<principle>> Least Astonishment
<<principle>> Strong Cohesion
<<principle>> Loose Coupling

```
Measures for Maintainability    [D0007]


Simplicity    [C0098]
  --> KISS    [R0106]
  --> YAGNI    [R0107]
  --> Occam’s razor    [R0108]
  supports --> Modifyability    [R0109]
  supports --> Reusability    [R0110]


Reusability    [C0044]


KISS    [C0094]
| Keep it simple and stupid


Analyzability    [C0045]


YAGNI    [C0095]
| You aren’t gonna need it
```

```
Occam's razor    [C0097]
| Among competing hypotheses, the one with the fewest assumptions should be selected


Modifyability    [C0046]


Information Hiding    [C0102]
| A sofware component shall hide its implementation details and
| make information accessible only via defined interfaces
  enables --> Reusability    [R0115]
  supports --> Modularity    [R0116]
  simplifies --> Testability    [R0117]
  simplifies --> Analyzability    [R0118]


Single Responsability    [C0089]
| A software component shall be responsible for one topic only


SOLID    [C0096]
  --> Interface Segregation    [R0101]
  --> Liskov Substitution    [R0102]
  --> Dependency Inversion    [R0103]
  --> Open/Closed    [R0104]
  --> Single Responsability    [R0105]
  supports --> Modularity    [R0111]


Interface Segregation    [C0092]
| Avoid general purpose interfaces,
| design multiple interfaces specific to the needs of different users/clients


Testability    [C0047]


Open/Closed    [C0090]
| Open for extension, closed for modification


Liskov Substitution    [C0091]
| An implementation of an interface can be replaced
| by another implementation of the same interface.
| In object oriented design, types can be replaced by subtypes.


Dependency Inversion    [C0093]
| A software component shall depend on abstractions, not on concrete implementations
  use Abstractions    [F0046]


Modularity    [C0043]


Least Astonishment    [C0103]
| A reader shall not be surprised when looking at the design.
  Conformity of style and concepts    [F0066]


Strong Cohesion    [C0104]
  influences --> Modularity    [R0119]


Loose Coupling    [C0101]
| split an entity that consists of multiple loosely coupled parts
  supports --> Modularity    [R0114]
```

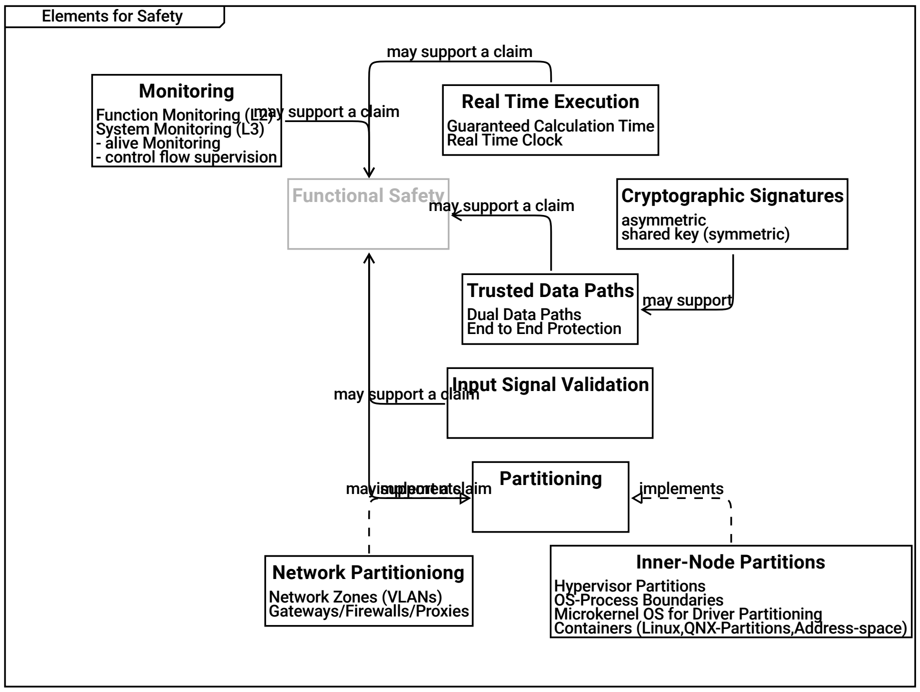


```
Measures for Reliability    [D0008]


Mature Platform    [C0109]
  OS provides resource limits    [F0061]
  OS does not swap, does not overcommit    [F0062]
  OS has mature peripheral-drivers    [F0063]
  --> Maturity    [R0124]


Layered Architecture    [C0061]
  improves --> Maturity    [R0039]


Example Guidelines    [C0073]
  Spark/ADA: Avionics    [F0022]
  Power of Ten: NASA    [F0019]
  SecureC    [F0021]
  MISRA-C/C++: automotive    [F0020]
  implements --> Coding Guidelines    [R0054]


Static Code Analysis    [C0086]
  enhances --> Maturity    [R0099]


Maturity    [C0035]


Code Generation    [C0087]
| An understandable model and a small code generator
| allow to generate mature software.
  supports --> Maturity    [R0100]


Coding Guidelines    [C0062]
| Coding guidelines define how to get reproducible behavior of software.
| Managing system resources is a key factor.
  static thread model    [F0010]
  | Execution threads shall not be started/stopped dynamically
  no endless loops    [F0008]
  | Every loop shall have a counter to ensures that
  | after a predefined maximum value the loop is definitely quit
  consistent error handling    [F0009]
  | Inconsistencies in error handling make
  | bugs in error handling more likely
  valid Memory Addresses    [F0007]
  | 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    [F0006]
  | 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    [F0005]
  lock critical sections    [F0024]
  | Always lock critical sections.
  | Exceptions to locking are a nightmare.
  single point of return: simple control flow    [F0023]
  | Simple control flow is key to understandable code
  improves --> Maturity    [R0040]


Availability    [C0034]


Redundancy    [C0074]
  2 of 3 voter    [F0025]
  duo-duplex    [F0026]
  limp home    [F0027]
  function migration    [F0028]
  enhances --> Availability    [R0055]


Fault injection Tests    [C0063]
  improves --> Fault Tolerance    [R0041]


Fault Tolerance    [C0036]


Input Signal Validation    [C0083]
  --> Fault Tolerance    [R0128]


Recoverability    [C0037]


Partitioning    [C0075]
  --> Fault Tolerance    [R0129]
  --> Recoverability    [R0130]
```



```
Measures for Security    [D0010]
| 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    [C0038]


Cryptographic Signatures    [C0079]
  asymmetric    [F0038]
  shared key (symmetric)    [F0039]
  supports --> Authenticity    [R0091]
  may support --> Non-Repudiation    [R0120]
  support --> Secure Boot    [R0123]


Encryption    [C0080]
  symmetric    [F0036]
  public/private    [F0037]
  supports --> Confidentiality    [R0092]


Confidentiality    [C0039]


Secure Boot    [C0108]
  --> Integrity    [R0121]
  --> Authenticity    [R0122]


Authenticity    [C0042]


Over-the-Air Updates    [C0078]
  Security Updates in Time    [F0035]
  supports --> Integrity    [R0090]


Integrity    [C0040]


Least Priviledge    [C0099]
| Entities shall have only the access rights they need for their purpose
  supports --> Accountability    [R0112]


Accountability    [C0041]


Partitioning    [C0075]
  supports --> Integrity    [R0089]


Certificates    [C0114]
  chain of certificates    [F0064]
  use --> Cryptographic Signatures    [R0126]
  divide and manage --> Accountability    [R0127]


Groups    [C0113]
| Grouping Clients/Actors helps
| Grouping Services
| helps in administration of access rights
  support --> Accountability    [R0125]
```

```
Network Partitioniong    [C0076]
  Network Zones (VLANs)    [F0029]
  Gateways/Firewalls/Proxies    [F0030]
  implements --> Partitioning    [R0087]


Inner-Node Partitions    [C0077]
  Hypervisor Partitions    [F0031]
  OS-Process Boundaries    [F0032]
  Microkernel OS for Driver Partitioning    [F0033]
  Containers (Linux,QNX-Partitions,Address-space)    [F0034]
  implements --> Partitioning    [R0088]
```



```
Elements for Safety    [D0011]


Monitoring    [C0084]
  Function Monitoring (L2)    [F0040]
  System Monitoring (L3)    [F0041]
  - alive Monitoring    [F0059]
  - control flow supervision    [F0060]
  may support a claim --> Functional Safety    [R0093]
```

```
Real Time Execution    [C0085]
  Guaranteed Calculation Time    [F0044]
  Real Time Clock    [F0045]
  may support a claim --> Functional Safety    [R0098]


Functional Safety    [C0081]


Cryptographic Signatures    [C0079]
  asymmetric    [F0038]
  shared key (symmetric)    [F0039]
  may support --> Trusted Data Paths    [R0097]


Trusted Data Paths    [C0082]
  Dual Data Paths    [F0042]
  End to End Protection    [F0043]
  may support a claim --> Functional Safety    [R0094]


Input Signal Validation    [C0083]
  may support a claim --> Functional Safety    [R0095]


Partitioning    [C0075]
  may support a claim --> Functional Safety    [R0096]


Network Partitioniong    [C0076]
  Network Zones (VLANs)    [F0029]
  Gateways/Firewalls/Proxies    [F0030]
  implements --> Partitioning    [R0087]


Inner-Node Partitions    [C0077]
  Hypervisor Partitions    [F0031]
  OS-Process Boundaries    [F0032]
  Microkernel OS for Driver Partitioning    [F0033]
  Containers (Linux,QNX-Partitions,Address-space)    [F0034]
  implements --> Partitioning    [R0088]
```