Standardin IEC 61508-3 testaustekniikoista

V-malli vai ketterämpi prosessi?

Mika Katara
mika.katara@tut.fi

Tampereen teknillinen yliopisto
Ohjelmistotekniikan laitos

Sisältö

Termien käännökset
Johdanto
Ohjelmiston turvallisuusvaatimusten määrittely
Ohjelmiston uudelleenkäyttö
Yksikkötestaus
Integrointitestaus
Järjestelmän turvallisuuden kelpuutus ohjelmiston osalta
Todentaminen
Liitteiden A ja B taulukot
Mallipohjainen testaus ja formaali verifiointi

Tässä esityksessä käytetyt termien käännökset

(Software) Safety Requirements Specification = (ohjelmiston) turvallisuusvaatimusten määrittely

Verification = todentaminen

Validation = kelpuutus

Safety manual = turvallisuusohjekirja

Software aspects of system safety validation = järjestelmän turvallisuuden kelpuutus ohjelmiston osalta

Normative = velvoittava

Informative = opastava

Tehdäänkö raudalla vai softalla?

Periaatteessa toiminnallisen turvallisuuden järjestelmät toteutetaan raudalla, mutta…

- PLC-pohjaiset ratkaisut ovat jo nyt yleisiä – nämä lasketaan ohjelmistoiksi
- Jos tarvittava laskenta on monimutkaista, tämä voi puoltaa ohjelmistoratkaisua
  - käytettävyys (availability) vs. turvallisuus
- Mikäli vaatimukset tarkentuvat vasta matkan varrella, voi softalla ”protoilla” helpommin
- Softan monistaminen on halvempaa kuin raudan
- Osataan tehdä paremmin softaa kuin rautaa
- …

Johdanto

"IEC 61508-3 requires a combination of fault avoidance (quality assurance) and fault tolerance approaches (software architecture), as there is no known way to prove the absence of faults in reasonably complex safety-related software, especially the absence of specification and design faults."

Järjestelmän ollessa toiminnassa pitkään luottamus laitteistoa kohtaan vähenee ja ohjelmistoa kohtaan kasvaa

Liitteet A-G, velvoittavia vain A ja D

A: Guide to the selection of techniques and measures

D: Safety manual for compliant items – additional requirements for software elements

Liite B "Detailed tables" ei ole enää velvoittava

Uusi liite C: Properties for software systematic capability

- Uusi versio sisältää paljon uutta verrattuna vanhaan
- Uudistusten vaikutuksia:
  - Vaihtoehtoja on enemmän
  - Standardia on hankalampi käyttää ”tarkistuslistana”
- Uuden version ymmärtäminen hankalaa ”vain” yhden alueen asiantuntijalle, saati sitten sen soveltaminen käytännössä
  - Soveltamisen vaikeus kasvaa jyrkästi SIL-tason kasvaessa
- Standardinmukaisuuden tarkistaminen hankalaa, “safety case” ei vieläkään pakollinen
- Menetelmät sangen erilaisia kuin laitteistolle
  - ”kaavoja” vain osan 61508-7 liitteessä D: ”A probabilistic approach to determining software safety integrity for pre-developed software”

Ohjelmiston turvallisuusvaatimusten määrittely (7.2)

Vaatimukset täytetään yleensä jonkinlaisella yhdistelmällä geneeristä sulautettua ohjelmistoa ja sovellusohjelmistoa

Oikeiden menetelmien valinnassa tavoitteena on, että

kaikki turvallisuustarpeet (safety needs) tulevat täytettyä

em. vaatimukset tulevat täytettyä **oikein**

määrittelyvirheistä ja monikäsitteisyyksistä päästään eroon

turvallisuusvaatimukset ovat ymmärrettäviä

ei-turvatoiminnot eivät pääse vaikuttamaan turvatoimintoihin ja

tarjotaan pohja verifioinnille ja validoinnille (**testattavuus)**

Standardi kannustaa laitteisto- ja ohjelmistosuunnittelijoiden **läheiseen yhteistyöhön** kun ohjelmiston osuus turvatoimintojen toteutuksessa ja ohjelmiston arkkitehtuuri tarkentuvat

Ohjelmistojen suunnittelu ja kehitys (7.4)

Testattavuus ja **ylläpito** on otettava huomioon ohjelmiston suunnittelussa

Modulaarisuus, tiedon kätkentä ja kapselointi ovat tärkeitä menetelmiä ylläpidettävyyden kannalta

Eräänä tärkeänä tavoitteena on turvatoimintoja toteuttavan ohjelmiston **yksinkertaisuus**

Riippumattomuus turvakriittisen ja ei-turvakriittisen koodin välillä, sekä eri turvatasoja edustavien koodikomponenttien väleillä **sekä datan että suoritusajan suhteen**

Kuten laitteistonkin osalta, voidaan **riippumattomia** ohjelmistokomponentteja käyttää yhdessä siten, että

SC N + SC N => SC N+1

Mikäli halutaan uudelleenkäyttää aiemmin tehtyä ohjelmistoa, tarvitaan turvallisuusohjekirja sekä yksi kolmesta alla esitetystä tavasta varmistaa uudelleenkäytettävän ohjelmiston toiminnallinen turvallisuus

Reitti 1s: ohjelmisto on tehty standardin vaatimalla tavalla

Reitti 2s: ”Proven in Use”

Reitti 3s: toteutuksen arviointi jälkikäteen (ks. 7.4.2.13)

Yksikkötestaus (7.4.7)

- Yksikön pitää toteuttaa spesifikaationsa
  - Tämä todetaan koodikatselmoinneilla ja yksikkötestauksella
- Käytettävien menetelmien pitäisi taata sekä testauksen kattavuus että sen oikeellisuus, toistettavuus sekä tarkasti määritelty testikonfiguraatio
- Yksikkötestaus tehdään järjestelmäsuunnittelun aikana tehdyn testaussuunnitelman mukaisesti
- Tavoitteena on näyttää, että yksikkö toteuttaa aiotun toiminnallisuuden – eikä mitään muuta
- Koska täydellinen testaaminen siten, että yksikön kaikki syöte- ja tulosarvot saataisiin katettua, on yleensä mahdoton tehtävä, käytettävien testausmenetelmien pitäisi pyrkiä pitämään testitapausten määrä järkevänä
- Yksikkötestauksen tulokset on dokumentoitava
- Jos testit eivät mene läpi, on korjaavat toimenpiteet määriteltävä

Integrointitestaus (7.4.8)

Ohjelmiston integrointitestaus tulee määritellä suunnittelu- ja kehitysvaiheen aikana

Integrointitestaussuunnitelman tulee ottaa kantaa seuraaviin:

- Ohjelmiston jakaminen hallittaviin integrointikokonaisuuksiin
- Testitapaukset ja –data
- Minkätyyppistä testausta tullaan suorittamaan
- Testiympäristö, -työkalut, -konfiguraatiot ja -ohjelmat
- Päätösehto (hyväksymiskriteerit)
- Korjaavat toimenpiteet, jos testit eivät mene läpi

Integrointitestaussuunnitelma pitää sisällään integrointitestauksessa käytettävät testitapaukset, jotka näyttävät, että kaikki yksiköt (modules, elements, subsystems) toimivat yhdessä oikein suorittaessaan aiotut toiminnallisuudet – eikä mitään muuta

Tulokset dokumentoidaan peilaten hyväksymiskriteereihin, ja myös syyt epäonnistuneille testeille dokumentoidaan

"During software integration, any modification to the software shall be subject to an impact analysis which shall determine all software modules impacted, and the necessary re-verification and re-design activities"

Tähän teemaan palataan kun pohditaan ketterien ohjelmistokehitysprosessien soveltamista standardin yhteydessä

Järjestelmän turvallisuuden kelpuutus ohjelmiston osalta (7.7)

Menetelmien valintaan vaikuttavat ominaisuudet:

- Testauksen kattavuus ja oikeellisuus suhteessa ohjelmiston määrittelyyn
- Toistettavuus
- Tarkasti määritelty konfiguraatio kelpuutukselle

Tulokset dokumentoidaan ja erityisesti jokaisen turvatoiminnan kohdalla tarvitaan tietoa seuraavista:

- Kronologisesti järjestetty historiatieto siitä mitä on tehty, jotta voidaan jäljittää myöhemmin mitä on tapahtunut
- Mikä kelpuutussuunnitelman versio on käytössä
- Mitä turvatoimintoa kelpuutetaan, viite kelpuutussuunnitelmaan
- Käytettävät työkalut ja laitteisto yhdessä kalibrointidatan kanssa

- Kelpuutustoimien tulokset
- Erot havaittujen ja odotettujen tulosten välillä (jos eroa, kirjataan päätös siitä, jatketaanko vai palataanko muutospyynnön kautta aiempaan elinkaaren vaiheeseen)

Testaus on pääasiallinen keino kelpuuttamiseen, mutta analyysiä (analysis), animointia ja mallinnusta voidaan käyttää apuna

Tulosten pitää osoittaa, että kaikki vaatimuksen turvatoiminnoille on oikein täytetty – ja vain ne

Mikäli kelpuutus epäonnistuu, syyt pitää dokumentoida

Todentaminen (7.9)

Tavoitteena elinkaaren vaihetuotteiden todentaminen siten, että vaiheen syötteistä tuotetaan oikeita ja yhdenmukaisia tuloksia

Menetelmien valintaan vaikuttavat ominaisuudet samoja kuin kelpuuttamisessa, mutta nyt suhteessa edellisen vaiheen tuotokseen

Todentamissuunnitelma viittaa kriteereihin, tekniikoihin ja työkaluihin, joita todentamisessa tullaan käyttämään, ja ottaa kantaa seuraaviin asioihin:

- Turvallisuuden eheyden (integrity) vaatimusten arviointi
- Todentamisstrategian, -toimien ja –tekniikoiden valinta ja dokumentointi
- Todentamistyökalujen valinta ja käyttö
- Todentamistulosten arviointi
- Tarvittavat korjaavat toimenpiteet

Mitä kaikkea todentaminen pitää sisällään:

- Ohjelmiston turvallisuusvaatimusten todentaminen
- Ohjelmiston arkkitehtuurin todentaminen
- Järjestelmäsuunnittelun todentaminen
- Moduulisuunnittelun todentaminen
- Koodin todentaminen (staattisesti)
- Datan todentaminen
- Suorituskyvyn (time performance) todentaminen
- Yksikkötestaus
- Integrointitestaus
- HW/SW-integraatio
- Järjestelmän turvallisuuden kelpuutus ohjelmiston osalta

Liitteiden A ja B taulukot

Taulukko A.5 – Ohjelmiston suunnittelu ja kehitys – yksikkötestaus ja integrointi

Taulukko A.6 – HW/SW-integraatio

Taulukko A.7 – Järjestelmän turvallisuuden kelpuutus ohjelmiston osalta

Taulukko A.8 – Muutokset

Taulukko A.9 – Ohjelmiston todentaminen

Taulukko B.2 – Dynaaminen analyysi ja testaus

Taulukko B.3 – Toiminnallinen- ja mustalaatikkotestaus

Taulukko B.6 – Suorituskykytestaus

Taulukko B.8 – Staattinen analyysi

Mallipohjainen testaus ja formaali verifiointi

- Mallipohjainen testaus on verrattain uusi teknologia, joka on otettu mukaa standardin uuteen versioon
  - Menetelmän taustalla ovat formaalit menetelmät ja formaali verifiointi
- Formaali verifiointi (todistaminen) on toinen tärkeä todentamismenetelmä, joka oli mukana jo standardin aiemmassa versiossa

KIITOS

Yhteystiedot:
Mika Katara
Tampereen teknillinen yliopisto
Ohjelmistotekniikan laitos
040 849 0743
mika.katara@tut.fi