SESAM
Försvarssektors användargrupp för Sofware Engineering

Checklista programvaruriskkällor

FMV dokumentbeteckning
14910:12183/2006

I-L Bratteby-Ribbing, FMV

Datum
2006-09-15

Ver 1.4

Checklista allmänna programvaruriskkällor *(Checklist of General Software-related Hazards)*

1 Syfte

Att i ett initialt skede av systemutvecklingen underlätta identifiering av möjliga riskkällor, genom att tillhandahålla en förteckning över riskkällor, som kan ha sitt upphov i systemets programvara.

2 Skede

Vid PHL-analys, PHA, SHA, SSHA ([6], [7]) då programvarurealiseringar börjar ta form.
Förteckningen är även användbar vid konstruktionsgranskningar (*design reviews)*.

3 Begränsningar

En förteckning över allmänna riskkällor kan aldrig bli komplett. En mängd checklistor finns, som belyser skilda aspekter[1] ur olika synvinklar. Denna lista har begränsats till riskkällor förknippade med system realiserade i programvara[2]. Av speciellt intresse är programvarudelar, som reglerar och övervakar andra systemdelar, vilka i sin tur kan aktivera en riskkälla (se 4.1). Vid identifiering av möjliga riskkällor för ett programvarusystem kan det därför vara mer effektivt att – i stället för att inleda en riskkälleanalys utgående från det övergripande systemets grundläggande energikällor[3] – starta från därav avhängiga och kända riskkällor[4]. Exempel på detta kan vara Oavsiktlig avfyrning, Friendly-fire[5] och Orimliga flygdata. Ett annat angreppssätt är, att söka efter mer programvaruspecifika defekter, som visat sig aktivt ha bidragit till olika typer av olyckor, t ex Dolda förutsättningar och Precisionsförluster (se 4.9).

Vanligtvis är det inte en enstaka riskkälla utan en rad samverkande faktorer[6], som ligger till grund för att ett olycksscenario skall utvecklas. Olika typer av analysmetoder behöver tillgripas för att få fram dessa samband.

4 Riskkällor

En strukturering av programvarans riskkällor kan göras på många sätt och ur olika synvinklar. Samma riskkällekandidat kan därför dyka upp under flera rubriker. Nedanstående uppdelning utgår från den indelning som tillämpats i [8] med tillägg för mer programvarutekniska aspekter.

4.1 Energiformer – Kraft → Säkerhetskritiska funktioner

Allmänna riskkällor, t ex strålning samt mekaniska, termiska, kemiska energiformer, identifieras redan vid säkerhetsanalys på den övergripande systemnivån, [8].
På programvarunivå finns vissa funktioner, som kan påverka dessa allmänna riskkällor och därmed är att betrakta som säkerhetskritiska:

Styrning / Direkt påverkan:

- **Hel**automatisk styrning av säkerhetskritisk hårdvara/programvara/aktivitet **utan** möjlighet/tid att bemöta säkerhetshot genom operatörsingripande
- Hel-/halvautomatisk styrning av säkerhetskritisk hårdvara/programvara/aktivitet **med** möjlighet/tid att bemöta säkerhetshot genom aktivering av oberoende skyddssystem/operatör

Ex: Autopilot, Styr-/reglersystem för säkerhetskritisk aktivitet/produktionsprocess, Vapenstyrning, Eldledning (målidentifiering, -utpekning, avfyrning, målföljning, detonation).

Övervakning:

---

[1] Ex: Energiformer, materialegenskaper, omgivningsfaktorer, användningsområde, mänskliga faktorer, systemmod, jfr [8].
[2] I 'programvara' inbegripes alla dess representationsformer (kravspecar, arkitektur, gränsytespecar, detaljerad design, implementation/kod, test- o analysbeskrivningar/-resultat etc).
[3] Vilka dessa är torde framgå ur de riskkälleanalyser, som redan genomförts på övergripande systemnivå.
[4] Ur PHL/PHA går att spåra vilka delar, som berörs av det övergripande systemets grundläggande energi- och riskkällor.
[5] Friendly fire är en vådahändelse bestående av flera samtidigt aktiverade riskkällor och övriga bidragande faktorer.
[6] Se t ex [9] : avsn 7 (torpedbåten).

- Övervakning av att det säkerhetskritiska systemet befinner sig i säkert tillstånd
- Funktion avsedd att detektera speciell typ av riskkälla

Ex: Kollisionsövervakning[7], Antikollisionssystem[8], Temperatur-/tryckövervakning.

Skydd:
- Överföring av säkerhetskritiskt system i osäkert tillstånd till säkert tillstånd (ev. i degraderad mod).
- Spärrar mot att säkerhetskritiskt system kan inta osäkert tillstånd.
- Skadelindring (aktuell då övergång från osäkert till säkert tillstånd ej möjlig)

Ex: Antikollisionssystem[8], Anfallsbekämpning, Överhettningsskydd/Termostat[8], Nödkylning, Nödstängning, Pilotutskjutning, Airbag, Spärr mot att kunna vrida kanon mot egen hytt.

Info-förmedling:
- Information med krav på omedelbar operatörsåtgärd för att bemöta säkerhetshot
- Kommando för styrning av säkerhetskritisk del genereras med krav på operatörsingripande för dess fullföljande
- Säkerhetsrelaterad information avsett som underlag för säkerhetskritiska beslut

Ex: Knapp som aktiveras vid osäkert tillstånd[9].

Indirekt påverkan:
Avser 'i princip' icke-kritisk programvara[10] som innehåller delar med:
- Indirekt påverkan på delar som klassats som säkerhetskritiska (dvs de med funktionalitet enl ovan)
- Informationsöverföring till säkerhetskritisk systemdel (dvs funktionalitet utöver 'Info-förmedling' ovan)
- Dynamiska språkmekanismer (ger opredikterbara systemegenskaper beträffande t ex prestanda, behov av processorkraft/minne, vilket kan leda till låsningar och exekveringsstopp)
- Samexistens med säkerhetskritisk programvarudel via delad plattform/minne/resurs (såvida plattformen/OS ej erbjuder säker partitionering mellan kritiska och icke-kritiska delar)
- Systemprogramvara, t ex
  - ◊ Operativ- och *runtime*-system (med/utan partitioneringsstöd), speciellt:
  - ◊ Systemets avbrottshanterare (vilken har att prioritera säkerhetskritisk process/*task*)
  - ◊ *Garbage collection* (minnesåtervinning, vilken belastar gemensamma resurser)
- Modellering/simulering/generering/verifiering av säkerhetskritisk programvara (givet *off-line*) [11]
- Analys under drift (vilken kan inverka på applikationens prestanda)

4.2 Materialegenskaper

Följande egenskaper hos programvara kan medföra ökad säkerhetsrisk och därmed uppfattas som riskkällor:

- Komplex design[12], t ex
  - _ Distribuerat system (ett antal oberoende, parallella/klustrade processorer med ett fixt/dynamiskt antal processer)[13]
  - _ Realtidssystem med tidskritiska element/funktioner (se 4.1)
  - _ Parallellexekverande processer (*concurrency*) (se 4.9)
  - _ Händelsestyrd/avbrottsstyrd realisering (dynamisk, prioritetsbaserad schemaläggning)[38] (se 4.9)
  - _ Cirkulära/vittförgrenade/dolda beroendekedjor
  - _ Stort antal tillstånd, förgreningspunkter (i program resp anropsflöde) eller gränssnitt/subsystem/-program

---

[7] Ofta halvautomatiskt: Varning för kollision åtföljd av instruktioner för hur systemet kan överföras till säkert tillstånd.
[8] Innehåller funktioner både för övervakning och styrning (därmed även skydd).
[9] Se [9]: avsn 3 samt 12.5.8: OSPREY.
[10] …vilken därigenom är att betrakta som säkerhetskritisk.
[11] Även om programvaran för modellering/simulering osv hålls separerad (*off-line)* från den systemversion som används i skarp drift, kan denna typ av programvara indirekt påverka hur det skarpa systemet utformas och är tänkt att användas. Görs ingen separering kommer i stället dessa funktioner att direkt kunna påverka säkerhetskritiska delar (se fotnot 21).
[12] Se t ex [3]: s 59, 71, 164-166, [9]: s 11-12, 27-28, [10]: s 35-37, 92-95.
[13] Processorerna kommunicerar via delat minne eller meddelanden över ett nätverk. Vid delat minne gäller bl a att förhindra samtidiga uppdateringar, att göra uppdaterad information tillgänglig även för övriga processer med så liten fördröjning som möjligt, att undvika felpropagering mellan processerna (t ex då en nod går ned). Vid meddelandebaserad kommunikation är ett av problemen att tackla den fördröjning, som överföring över nätet medför.

_ Indeterministiska språkelement/designlösningar (se 4.9)
_ Dolda ingångar/bakdörrar, onåbar kod (dvs död/deaktiverad kod) (se 4.9)
_ Ostrukturerad design, svag modularisering eller OO

- Extra funktionalitet (se 4.9)
- Numeriska representationer:
  _ Icke-kontinuerlig representation av matematiska funktioner[14]
  _ Approximativ representation av numeriska typer[15]
- Bristfällig dokumentation
- HMI (se 4.5)

4.3 Omgivningsfaktorer

Geodetiska extremlägen[16].
…

4.4 Användningsområde

_ Drift/användning[17] :
  _ Överskriden maximal drifttid
  _ Avsteg från avsedd/specad användningsprofil[18]
  _ I strid mot avsedd/specad omgivning
  _ Utanför tillåtna gränser[19]
_ Driftbeskrivning:
  _ Ofullständig dokumentation av avsedd:
    - användningsprofil,
    - tillämpningsområde,
    - systemomgivning,
    - systembeteende
    - operatörsåtgärd för fall där specade förhållanden ej föreligger/åtföljs (t ex checklistor)

4.5 Mänskliga faktorn

Mänskliga misstag inträffar både under uppbyggnad av ett system och vid dess användning/drift (vilket framgår av de olika avsnitten under kap 4). Under detta avsnitt behandlas främst riskkällor i programvarans gränssnitt mot operatör. Därmed utesluts t ex avsiktliga felgrepp samt organisatoriska aspekter och otillfredsställande förhållanden under systemet uppbyggnad (bristande säkerhetskultur, tidspress etc).

De programvarutekniska aspekterna på människa-maskin interaktionen inkluderar bl a hur systemet designats att nyttja olika utmedia[20], de typer av inmatningsmedia som valts för olika situationer, hur dessa avses användas för att stödja operatören i diagnostik, beslutsfattande och fortsatt agerande. En minst lika viktig ingrediens är en design utformad att bemöta oväntade operatörshandlingar, för att förhindra att dessa kan utlösa, bidraga till eller misslyckas förhindra ett olycksscenario.

_ Informationsförmedling (system → operatör)
  _ Förvirrande/motsägande/missledande/ofullständig/inaktuell/föråldrad info

[14] Kontinuerliga funktioner implementeras m h a funktioner med ett godtyckligt antal diskontinuiteter, vilket innebär att små förändringar i parametervärdena kan ge ett radikalt annorlunda beteende hos funktionen.
[15] Precisionsförluster p g a ackumulerade avrundningsfel kan uppstå (t ex vid lång drift utan omstart). Flytt av programvara till annan processor med annan flyttalsrepresentation, ger icke-identiska beräkningar o förändrat beteende ([9]:sist i 12.5.7.)
[16] Typiska fallgropar för navigeringssystem är trakterna kring Nordpolen (där förflyttning norrut kan innebära söderut) samt Döda havet (där systemet måste kunna hantera flygning under havsnivå).
[17] Jfr [3]: fotnot 317.
[18] Funktionalitet avsedd för viss objektyp använd mot andra/snarlika objektyper (med delvis annorlunda karakteristika). Ex: Patriotsystemet avsett för målfångst av stridsflyg (Mach 2) användes mot Scud-missiler (Mach 6), se [9]:12.5.5.
[19] Ex: Flygning utanför tillåtna gränser/flygenvelopen.
[20] Ex: Detaljer beträffande utforming av skärmbild/ljud/reglagerörelser, hur informationen är strukturerad på skärmen, informationsinnehållet (semantiska formuleringar, prioritering/angelägenhetsklassning, graden av granularitet/precision/ tillförlitlighet, tidsaspekter: aktualitet/ålder, angelägenhetsgrad), kommando/formulär/menynavigeringens upplägg.

_ Info presenteras/orienteras/struktureras i strid mot egenskaper hos det verkliga objekt, som avses
_ Ingen speciell markering för ålder hos info med begränsad gilitighetstid (t ex *fading*)
_ Otillräckligt beslutsunderlag hos lämnad info, där operatörsåtgärd förväntas
_ Okänd info-källa, oklart info-syfte, oklart info-status (ny/ändrad/borttagen)
_ Info om aktuellt systemmod eller förändrat systemmod saknas/svårtillgänglig
_ Ingen info lämnas vid systemövergång till degraderad mod
_ Info, som kräver omfattande sökningar/bearbetningar för att återfinnas/tolkas
_ Oklart i vissa lägen huruvida systemet eller operatören har prioritet
_ Info-innehåll ej anpassad till mänsklig kapacitet (överbelastning av operatörs minnes-/handlingsförmåga)
_ Säkerhetskritisk info utan kritikalitetsklassning eller utan avvikelse från runtinmässig info
_ Inga ångermekanismer/ varningar då systemet på gränsen lämna säkert tillstånd eller då variabel närmar sig säkerhetskritiskt intervall
_ Bristande info om förväntad operatörsåtgärd då systemet mottagit/lämnat motstridig info
_ Ingen återmatning av effekt av vidtagna operatörsåtgärder
_ Operatörsåtgärder eller presenterad info loggas ej

_ Operatörsåtgärder (operatör → system)
_ Icke avsedd eller felaktig inmatning förhindras ej
_ Inmatning kritisk för systemsäkerheten accepteras utan extra kontrollfrågor/-åtgärder
_ Farlig aktivitet kan inledas utan extra säkerhetsåtgärder (såsom redundanta inmatningsmedia, extra steg)
_ Inga handlingsalternativ presenteras i säkerhetskritisk situation
_ Snabbutgång från osäkert tillstånd / farlig aktivitet saknas (t ex operatörsåtgärd/-kommando/-knapp)
_ Genväg möjlig i operatörsprocedur, där hel sekvens väsentlig för systemsäkerheten
_ Säkerhetskritiskt larm kan nollställas innan korrigerande åtgärder vidtagits

4.6 Systemfaser/-operationer/-mod/-tillstånd

_ Simulering/Test/Utbildning[21]
_ Driftsförändring/Modövergång/Upp-nedkoppling/Aktivering
_ Nödstart/nödstängning
_ Extrem driftmod/Degraderad driftmod

4.7 Systemspecifikation

_ Odokumenterade krav (systembeteende vid normala/onormala förutsättningar/situationer)
_ Odokumenterade systemförutsättningar (användning/drift: se 4.4, systemomgivning)

4.8 Gränssnitt – Integration

- Ofullständiga/inaktuella beskrivningar av interagerande systemkomponenter
- Missvisande tidshantering:
  – Oklar distinktion mellan t ex verklig, predikterad, återuppspelad resp simulerad tid
  – Felsynkroniseringar/störningar, t ex över nätverket[22]
- Oavsiktlig samverkan mellan integrerade delar[23]:
  – Förbisedda faktorer som kan aktivera viss händelse
  – Inhibering av styrfunktion/-krets som delas av integrerade komponenter

[21] En viss påverkan på systemet i skarp drift kan resultera från dessa aktiviteter (se fotnot 11), i synnerhet där utbildnings-/simulerings-/testfunktioner inte rensats bort från det skarpa systemet. Om utbildning, simulering eller test tillåts utföras på den systemversion, som också används vid skarp drift, kan (vid ofullständig återställning till verkligt systemstatus) rest-tillstånd från tidigare systemdrift ligga kvar latent och felaktigt påverka systemets statusbild i skarp drift. Därmed kan en verkligt kritisk situation uppstå, se t ex [3]: fotnot 60 (NORAD), 142 (bombluckan). Ytterligare exempel är det simulerade el-bortfallet i Tjernobyl, se [9]:12.5.1.

[22] Exempel: Fördröjningar över nätverket av bl a den gemensamma systemtid (räknad från den tidpunkt då första noden startas) eller av den absolut tid, som används vid beräkningar. Missad växling sommar-/vintertid vid politisk tid/lokal tid (den kalendertid som används vid skärmpresentation). Störningar vid hämtning av GPStid, UTCtid etc.

[23] Designbrister som uppstår vid integerering av komponenter, vilka taget separat inte är säkerhetshotande.

Ex: Självdetonation av torped fastnad i tub[24], Synkroniserade broms- och takljusblink[25]

4.9 Konstruktion/Implementation[26]

* Nyutveckling:
  - Ingen separering mellan delar/funktioner/data av olika kritikalitet[27]
  - Otillräcklig/Felaktig (tid/värde/resultatordning) /Utebliven/Överflödig funktionalitet
  - Data-/kommandofel:
    - Datasampling för långsam/snabb
    - Datakollision/minnesöverskrivningar p g a alltför många processorer på samma LAN, modfiering av felaktig/likartad datapost, samtidig modifiering från flera källor
    - Data i felaktig form eller ordning
    - Kommando ej avsänt/mottaget
    - Kommando ogiltigt (t ex för aktuellt systemmod p g a transmissionsfel/operatörsmisstag etc)
  - Onödig funktion/knapp/meny aktiverad/valbar, död kod
  - Otillräcklig precision för att kunna uppfylla specad funktionalitet[28]
  - Otillräcklig prestanda för att kunna uppfylla specad funktionalitet
  - Funktion avsedd för visst system-/exekveringsmod ej deaktiverad i annat mod
  - Ej deaktiverade/kvarglömda testinstruktioner
  - Oidentifierade/ohanterade *exceptions* (dvs exekveringsalternativ vid exceptionella/osäkra tillstånd)
  - Ingen redundans för att motverka (kunna lämna) osäkert tillstånd.
  - Beroende mellan redundanta realiseringar (t ex identiska realiseringar i stället för diversifierade)
  - Logikfel[29]
  - Osäkra/tvetydiga/indeterministiska språkkonstruktioner:
    - Funktioner med sidoeffekter
    - Konstruktioner vars evalueringsordning beror av vald kompilator[30]
    - Subprogram vars parameteröverföringsmekanism (call-by-ref, by-value) beror av valt språk
    - Dynamiska konstruktioner som ej kan blockeras från språket
    - Dynamisk konstruktion, som kan leda till minnesbehov utöver tillgängligt minne
    - Tidskritisk konstruktion, där processorkapaciteten inte räcker och deadlines missas
    - Oinitialiserade/odefinierade variabler/pekare
    - Globala variabler (i stället för '*information hiding*'-teknik), delat minne[13]
    - Rekursion, *while*-loopar, loopar med multipla utgångar, avsnitt utan utgångar[31]

* Återanvändning (ÅA):
  - Ofullständig/otillgänglig dokumentation över programvarans olika representationsformer[32]
  - Konstruktionen ej baserad på 'Design för ÅA'-principer
  - Icke-identisk drifts-/användningsprofil (se 4.4)
  - Dold förutsättning: Odokumenterade antaganden (härledbar endast via detaljstudium av koden)[33]
  - Dolt beroende till andra komponenter (♣)

---

[24] Sprängningen triggad då torpedbana 180° mot utskjutningsriktningen, vilket även råkar inträffa då avfyrad torped fastnat i tuben och fartyget ändrar kurs 180°.
[25] Se t ex [9] : avsn 7 (polisbilen).
[26] Listan över möjliga riskkällor kan göras längre. Många feltyper, som i princip kan utgöra riskkällor, fångas dock mer effektivt via traditionella programutvecklingsverktyg (t ex för statisk och dynamisk analys).
[27] Interaktioner/beroenden mellan icke-kritisk och kritisk del kan bl a innebära att störningar sprids till kritisk del, att kritiska data är åtkomliga/manipulerbara från icke-kritisk del, att icke-kritisk del kan stjäla processortid från kritisk del.
[28] Ex:Patriot, där förlängd drifttid (med ackumulerade avrundningsfel som följd) omöjliggjorde målfångst, se [9]:12.5.5.
[29] Ex: Osprey-logikens felaktiga aktivering av en mjukvaruknapp i ett säkert (men degraderat) systemtillstånd, vilket ledde till osäkert tillstånd och haveri. Se [9]:12.5.8.
[30] Ex: Tillämpad evaluerings- resp intialiseringsordning (för operander resp globala/statiska variabler) kan påverka resultatet.
[31] Se t ex [3]: fotnot 46 och 125.
[32] Krav, gränsytespecar, design, implementation, test-/analysresultat, problemrapporter, ändringsspecar, utvecklings- resp användarhistorik/-teknik.
[33] Ex: Kod baserad på Ariane 4:s trajektorybanor återanvändes i Ariane 5, som försetts med starkare motorer (se [9]:12.5.4).

– Dold ingång/bakdörr/testgenväg
– Osäker/indeterministisk konstruktion som programmeringsmässigt ej kan isoleras vid ÅA
– Onödig/oanvänd funktionalitet som programmeringsmässigt ej kan göras oåtkomlig för applikationen[41]
– Onödig/oanvänd funktionalitet som kan avskärmas, men som ej deaktiverats vid ÅA i applikationen
– Implementation beroende av en *default*-mekanism och flyttad till annan processor utan denna[34]
– Implementation, som återanvänds på annan processor med annan flyttalsrepresentation[34]
– Realtidsrealisering, som återanvänds på en snabbare/långsammare processor[35]
– OS[36]:

- Garbage collection som enbart styrs av OS (ej av applikationsprogrammeraren)
- Dynamisk minnesallokering[37]
- Otillräckligt antal prioriteringsnivåer tillgängliga för applikationsprogrammet
- Skeduleringsmodell (fördelning av delade resurser på parallellexekverande processer)[38]
- Osäker partitionering av minne/tid _ ingen garanti att kritisk applikationsdel skyddas från påverkan från del av lägre/ingen kritikalitet[39]:
  a) Exekvering av kritisk process kan avbrytas för exekvering av process med lägre kritikalitet,
  b) Minne avsedd för kritisk information påverkbar av del med lägre kritikalitet.
- Otillräckligt integritetsskydd:
  a) Uppdateringar/infoavtappning av kod/data möjlig (t ex via nätet).
  b) Ingen garanti mot att en applikation kan påverkas av annan applikation på samma OS
- Otillräcklig feltolerans: fel[40] i en applikationsdel kan spridas till andra delar/applikationer
- Otillräcklig prestanda (responstid vid avbrott/interrupt)
- Överflödig funktionalitet svår att blockera (bl a vid ostrukturerade/oklara beroendekedjor, se ♣ ovan)[41]

4.10 Produktionsprocess

- Ofullständiga/inaktuella systemsäkerhetsanalyser[42]
- Otillräcklig/Missad analys av krav- och systemändringars effekt.
- Otillräckliga verifieringsprocedurer (granskning/test ej utförd i paritet med systemdelens kritikalitet)
- Inaktuella testfall/Missade ändringstest
- Säkerhetsanalyser o verifieringar ej utförda med run-timesystemet som del av applikationen

4.11 Produktionsverktyg

- Inkorrekt implementation av språket semantik
- Olika optioner satta vid validering av kompilator resp OS/run-time-system
- Icke-korrekt kodgenerering (kompilatorbuggar, kodoptimeringsmissar)
- Inaktuella/obefintliga listor över identifierade problem i hårdvara samt kompilator
- Olika kompilatorversioner vid kodgenerering för test resp drift

5 Referenser

[1] Uppdragsspecifkation SäkAnalysMetoder, FMVdokument 14910:88774/2005.
[2] Försvarsmaktens handbok för Systemsäkerhet, M7740-784851 H SystSäk, s 92 ff.
[3] Försvarsmaktens handbok för programvara i säkerhetskritiska tillämpningar, M7762-000531 H ProgSäk
[4] Hazard Analysis Techniques for System Safety, C.A. Ericsson, ISBN 0-471-72019-4, s 62 ff + App C.
[5] System Safety Analysis Handbook, www.system-safety.org/ProductsSale.php, s 3-209 ff.
[6] PHL-analys, Faktablad för Preliminär riskkällelista, FMVdokument 14910:2662/2006.

[34] En av flera möjliga problem med USS Yorktown. Se [9]: 12.5.7.
[35] Se t ex [3]: fotnot 208.
[36] Se t ex [9] s 18-19, 29 (*Smart Ship*-konceptet), http://sesam.smart-lab.se:Arbetsgrupper:Programvarusäkerhet:Produkter.
[37] Problem, bl a om minne frisläpps flera ggr (*exception* triggas), inte frigörs alls (*memory leakage*) eller resulterar i överfyllda buffertar (där andra minnesareor för kod/data skrivs över).
[38] Olämplig modell kan bl a leda till prioritetsinversion, *deadlocks* m m . Se t ex *Mars Pathfinder*, [9].
[39] Se t ex [3] fotnot 133.
[40] Fel: felkälla, feltillstånd eller felyttring.
[41] Problemet diskuteras bl a i [9]: avsn 12.4.
[42] Utred vilka osäkra tillstånd som kan uppnås vid normal såväl som vid degraderad driftmod.

[7] PHA, Faktablad för Preliminär riskkälleanalys, FMVdokument 14910:2795/2006.
[8] Generell riskkällechecklista, FMVdokument 14910:6057/2006.
[9] Programvarusäkerhet –en introduktion, FMVdokument KC Ledstöd 14910:38346/02.
[10] NASA Software Safety Guidebook, NASA-GB-8719.13.

6 Dokumenthistorik

Version	Datum	Beskrivning
1.0	06-02-03	Faktablad enl [1]: uppgift 4. Utkast inför SESAM-gruppens första PHL-analys.
1.1	06-04-04	Kompletteringar baserat på [3], [9], [10] samt FMV:s Programvarusäkerhetskurser.
1.2	06-05-10	Kompletteringar efter SESAM-gruppens möte 06-04-07.
1.3	06-08-18	Fortsatta kompletteringar baserat på [3], [9], [10] mfl.
1.4	06-09-15	Fortsatta kompletteringar under 4.6, 4.7. Tillägg fotnot 11, 18, 28, 29, 33, 34.