Utfärdad av, tjänsteställe, telefon
Torbjörn Jungeby, RTKP2, 0586-829 27

Datum
2001-03-20

Dokumentbeteckning
KP001007

Mottagare
Håkan Edler, deltagarna i SESAMS mikroprojekt ”Drömverktyget”, se avsnitt 1, och kontaktpersoner hos utvalda vertygsleverantörer, se avsnitt 7

Informationsklass
ÖPPEN

SESAM Mikroprojekt ”Drömverktyget”. Delrapport 1

Sammanfattning av vad ett utvecklingsverktyg för programvara förväntas klara.

Sammanfattning

Dokumentet tas fram inom ramen för ett Mikroprojekt inom SESAMS arbetsgrupp "Metodik". Mikroprojektet har rubriken: "Krav på verktygsleverantörer" med arbetsnamnet "Drömverktyget".

SESAM har tillkommit för att organisera och stimulera sammarbete och samverkan inom programvaruområdet mellan försvarsindustrin, FMV och FOA. Se även: sesam.tranet.fmv.se

Detta dokument sammanfattar de krav och önskemål som ett utvecklingsverktyg för konstruktion och framtagning av programvara förväntas klara av enligt arbetsgruppen för Mikroprojektet "Drömverktyget".

Innehåll

1 Mikroprojektet: "Drömverktyget"

Gruppen för Mikroprojektet "Drömverktyget" består av följande medlemmar:

Torbjörn Jungeby,Saab Bofors Dynamics, Karlskoga, sammankallande.

Erik Marklund, Saab Bofors Dynamics, Karlskoga.

Anna-Stina Mattsson, Ericsson Saab Avionics, Kista.

Billy Johansson, Saabtech, Järfälla.

Mats Rundqvist, Saab, Linköping.

Gruppens mål är att under året, 2001, genomföra följande aktiviteter:

- Lista önskvärda krav, egenskaper för ett "Drömverktyg".
- Lista de verktygsleverantörer vi tror kan leverera "Drömverktyget".
- Genomföra ett "seminarium" med representanter från utvalda leverantörer.
- Sammanfatta uppgifter från "seminariet".
- Försöka svara på varför de s k CASE-verktygen inte riktigt slagit igenom, trots att de funnits och använts i många år.
- Kortfattat i rapportform och som anförande, sammanfatta resultat från samtliga målpunkter ovan. Klart till SESAM:s höstseminarium 2001.

2 Utförande

Som bas för att lista "Drömverktygets" önskvärda egenskaper har inkommna svar från ett 30-tal programvaruingenjörer använts. Svaren har baserats på följande frågeformulering:

> Vi vill att ni för varje rubrik respektive utvecklingsfas nedan listar de funktioner, önskemål, krav som ni tycker ett bra utvecklingsverktyg, -miljö, bör, skall innehålla.
>
> Klassa gärna önskemålen i bör och skall.
>
> Allmänt
> Kravanalys, analys
> Preliminär design
> Detaljerad design
> Kodning

Enhetstest
Förintegration
Integration
CSCI-test

Kan ni namn på leverantörer och deras utvecklingsprodukter ange dessa också.

Önskemålen har raffinerats av gruppen vid en gemensam träff. Det beslutades då att kategoriseringen bör och skall inte skall användas, dessutom komprimerades utvecklingsfaserna en del för att ge mer samlade utfall, enligt nedan:

Allmänt
Kravanalys, analys
Preliminär design, Detaljerad Design = Design
Kodning & enhetstest
Förintegration, Integration & CSCI-test

Dessutom konstaterades snabbt att vi ville ha en egen kategori för kravverktyg eftersom det ansågs så centralt och hade en rad egenskaper vi explicit ville uttrycka.

3 Utfall

Utfallen presenteras i form av två kompletta svar från den tillfrågade gruppen som vi ansåg var så fulltäckande var för sig att dessa återges ograverade.

Utöver dessa kommer sedan:

- Uppräkning av krav på "Drömverktyget" som funktion av utvecklingsfaserna samt kategorin krav på kravverktyg.
- Uppräkning av alla påkommna leverantörer.
- Uppräkning av de leverantörer som enhälligt av gruppen utvaldes att tillfrågas om medverkan i det seminarium för "Drömvertyget" som gruppen har att arrangera våren 2001.

4 Gallupsvar

4.1 Gallupsvar 1

Allmänt
De ska vara användbara så att det känns att dom ger oss stöd i det arbete som vi ska utföra.
De ska ha låg buggnivå.
Det ska finnas utbildning på verktyget.
Det ska finnas support för verktyget.
Verktygen och vår arbetsprocess ska stämma överens och det är viktigare att ha ett bra verktyg som stödjer hela processen än flera verktyg som inte hänger ihop. Då är det nog bättre att anpassa arbetsprocessen till effektiva verktyg.

Kravanalys, analys
Kravspårning, kravnedbrytning som medger att bakgrundsinformation läggs in som inte behöver redovisas i den färdiga SRS'en. Detta för att tankar och avväganden som gjordes då ska finnas kvar i ett senare skede när kraven ska omarbetas/utökas eller på annat sätt förändras.
Det vore bra om det var uppbyggt med länkar både "uppåt" för att nå grundkraven och "nedåt" för att nå designen.
Det borde finnas stöd för att detektera Grundkrav som inte implementeras.
Modelleringsmöjligheter för systembeteendet med koppling till design modelleringen.
Verifieringsmetod beskrivning.

Preliminär design
Designmodelleringen ska kunna kopplas till krav och verifieringsmetod som direkt ska kunna läsas.
Det ska finnas direkt koppling till Detaljdesignen av motsvarande del.
Stöd för de steg som ska ingå i preliminär design (helst styrbart per projekt).

Detaljerad design
Direkt koppling till Prel Designen, Verifieringsmetod och krav.
Direkt koppling till koden.
Stöd för de steg som ska ingå i detaljerad design (helst styrbart per projekt).

Kodning
Direkt koppling till Detaljerad design, Preliminär Design, Verifieringsmetod och krav.
Direkt koppling till Testfall för enhetstest.

Enhetstest
Stöd för regretionstest, automatiserat genomförande och utvärdering.
Stöd för analys av testtäckning (inte bara kodradstäckning utan även värdemängdstäckning).

Förintegration
Stöd för regressionstest, automatiserat genomförande och utvärdering.
Stöd för analys av testtäckning (inte bara kodradstäckning utan även värdemängdstäckning).

Integration
Stöd för regressionstest, automatiserat genomförande och utvärdering.

CSCI-test
Stöd för regressionstest, automatiserat genomförande och utvärdering.

4.2 Gallupsvar 2

Kravanalys,analys
Kravdatabas!
Databasen skall hålla både "systemkrav" och "mjukvarukrav" och kopplingen mellan dessa. Förändringar i t ex f-speckrav kan då direkt peka ut vilka SRS-krav som påverkas (man slipper då att "manuellt" hålla koll på förändringar av kravbilden).

Kravdatabasen bör dessutom vara integrerad med editor/ordbehandlare för kravdokumentskrivning, man kan då t ex skriva ett SRS-krav som tilldelas en unik identitet, denna identitet ska då kunna då "kopplas" mot t ex ett eller flera f-speckrav. Man bör även kunna nå associerade krav från t ex en SRS via nån form av "hyperlänkar".

Det vore även bra om krav kunde kopplas mot DFD-signaler/dataflöden, så att förändringar i DFD:n kan peka ut påverkade krav (samt tvärt om).

Preliminär design
Verktyg för att rita externa ASG:er bör generera kodskelett som kan användas i den fortsatta utvecklingen. Helst skall inte kodskelettet inte kunna modifieras utan att rita om ASG:n.

Kodning
Om man nu ska envisas med att rita intern ASG innan kodning påbörjas så bör även denna kunna användas för att generera kod. Vore dock bättre (tycker jag) om man hade ett verktyg som genererar intern ASG från den färdiga koden - jag har i alla fall aldrig lyckats rita en intern ASG som "hållit" även efter kodning, som det fungerat hittills så har jag fått rita om stora delar av ASG:n efter kodning vilket har tagit en hel del onödig tid!

Om man ska rita ASG innan kodning så SKA verktyget ha nån form av "autoroute", enbart detta skulle spara massor av tid!!

Enhetstest
Hjälp för att få full kodtäckning bör finnas.

Verktyg för att skriva testfall skulle gärna få vara integrerat med samma verktyg som används för att skriva krav. På detta sätt skulle varje testfall kunna kopplas till de krav som testas, och när ett test har körts så skulle man automatiskt kunna se vilka krav som verifierats (eller från ett SRS-krav se var och hur det ska testas, samt om det har testats och resultatet från testet).

Verktyget bör ha möjlighet att med någon "syntax" skriva testfall och förväntat utdata så att man kan få "automatisk felutpekning" om något gått fel vid ett test! Detta skulle göra det enklare att köra regressionstestning. Dessutom är det nog både effektivare, roligare och bättre att lägga mera tid på att skriva och granska testfall istället får att sitta och manuellt gå igenom en massa output-filer!

5 Krav på "Drömverktyget"

De uppräknade kraven på "Drömverktyget" är kortfattade och lämnar i vissa stycken medvetet utrymme för tolkning. Det avser typiskt krav som beskriver egenskaper som är komplexa och intuitivt lättare att förstå än att bena upp.

5.1 Allmänt

1) Bakåtkompatibelt
2) Import/Export med standard-format på filerna
3) Väldefinierad koppling mellan olika verktyg för de olika faserna
4) Reverse engineering från kod till detaljerad design, t ex buhrgrafer
5) Generering av kodskelett från detaljerad design, t ex ritade buhrgrafer
6) UML, verktyg som stöder grafisk modellering bör följa den standarden
7) Konfigurationsstyrning, olika versioner av modeller på alla nivåer bör vara konfigurationsstyrda
8) Bra editor som en integrerad del av verktygen, alternativt bör verktygen ha väldefinierade textfilsgränsyta, jmf punkt 2)
9) Support i form av buggfixning ska vara god så att man inte tvingas till onödiga ”workarounds” kring kända fel

10) Utbildning, sjösättningshjälp, när verktyget först tas i bruk, och kontinuerlig utbildning under dess livstid

11) Livslängd, lagring, produktstrategier ska vara definierade så att ändringar av modeller kan göras även efter lång tid (verktygens livslängd bör överträffa de produkter de används för att utveckla!)

12) Certifiering av kompilator, operativsystem och testverktyg

5.2 Kravanalys, analys

13) Testfallsgenerering utgående från analysmodell

14) Koppling krav/testfall

15) Kravallokering från funktionskrav på CSCI-nivå till komponentnivå

16) Enhetsdefinitioner - burkifiering, en analysmodell bör kunna uttrycka väsentliga hårdvaruegenskaper

17) Systemanalys med både SW och HW, såväl elektronik som mekanik, där samarbetet dem emellan kan beskrivas

18) Konsistenschecker inom modeller

19) Analyskompilator för gränsytor mot andra enheter, alltså kontroll av att gränsytekrav omhändertagits i en modell

5.3 Design

20) Koppling mellan Preliminär Designen, Verifieringsmetod och designkrav

21) Direkt koppling från design till koden, hyperlänkar.

5.4 Kodning & Enhetstest

22) Kodgenerering från analysmodell, minst gränsytedeklarationer från datakatalog

23) Testfallsgenerering från analysmodell

24) Kodtäckning, stöd för att mäta vid utförda tester

25) Kravtäckning, stöd för att utvärdera beskrivna testfall

5.5 Förintegration, Integration & CSCI-test

26) Testtäckning
27) Inspelning för regressiontester av GUI
28) Debugger för målmiljö
29) Grafisk presentation av inre tillstånd i programvaran i målmiljö
30) Daily build

5.6 Kravdatabas

31) Kravstruktur
32) Spårbarhet
33) Hyperlänkar
34) Rapportgenerering
35) Beroendediagram
36) Ägare till krav
37) Kopplingar till analysmodell
38) Kopplingar till tester
39) Aktiva/passiva krav
40) Koppling till CM-verktyg
41) Kalkylstöd

6 Leverantörer

Leverantörslistan totalt var:

- NRT (Bridgepoint).
- Mango DSP.
- Nohau (Rapsody, utvecklingsmodell ROPE).
- Rational (bl a Requiste Pro Soda, Rose, utvecklingsmodell RUP).
- Windriver (Tornado).
- Telelogic.
- ATTOL.
- Vector Software.
- Adaptive Systems (bl a Teamwork).
- SDRC (Slate).
- McCabeUnderstand for Ada (AdaDL).
- Software research (CAPBAK).
- Mercure Interactive.
- XRunner.
- IPPS CRADLE/REQ.
- Aonix 10X.
- Centerline Development QC Replay.
- Adatest IPL.
- Panorama.
- Marconi.
- RTM.

7 Utvalda leverantörer

Leverantörerna i kortlistan, med angivna kontaktpersoner, var:

- NRT, Christer Andersson, `christer@nrt.se`.
- Rational, Tommy Lenhamn, `tole@rational.com`.
- Nohau, Stefan Ryberg, `sr@nohau.se`.
- Windriver, Ulf Dahlblom, `ulf.dahlblom@windriver.com`.
- Telelogic, Magnus Seifert, `magnus.seifert@telelogic.com`.
- Adaptive Systems, Martin Johem, `Martin.Johem@adaptive.se`.
- SDRC, Tony Bergström, `tony.bergstrom@sdrc.com`.