Erfarenheter från införande av MBSE i Skeldar och Neuron

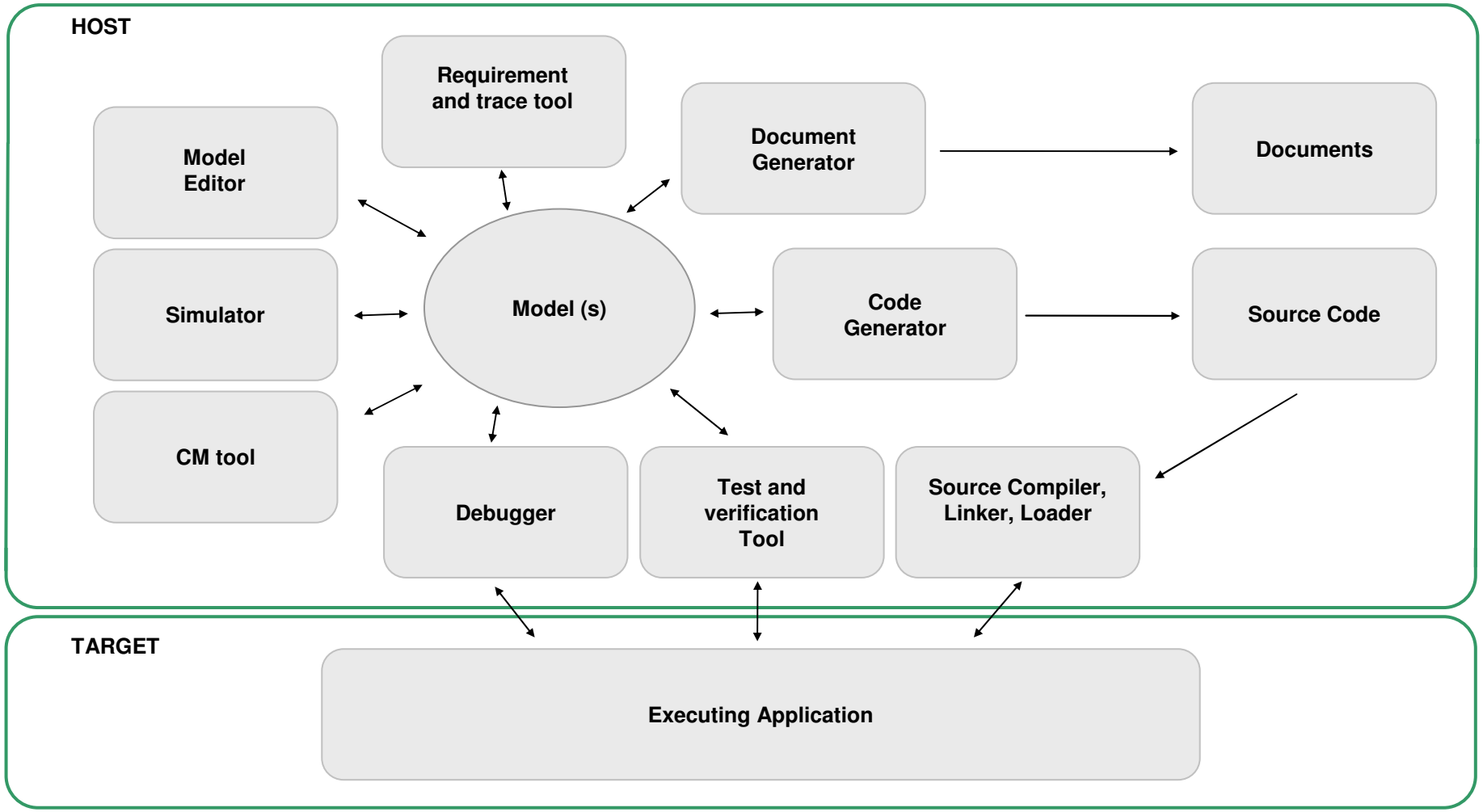
Magnus Skoog 2008-11-20

SAAB

Innehåll

- Introduktion till MBSE
- SKELDAR
- NEURON

MBSE Visionen



SAAB

MBSE i Skeldar

- Vad har åstadkommits?

ILLUSTRATION: SAAB

Bra saker som vi åstadkom (1)

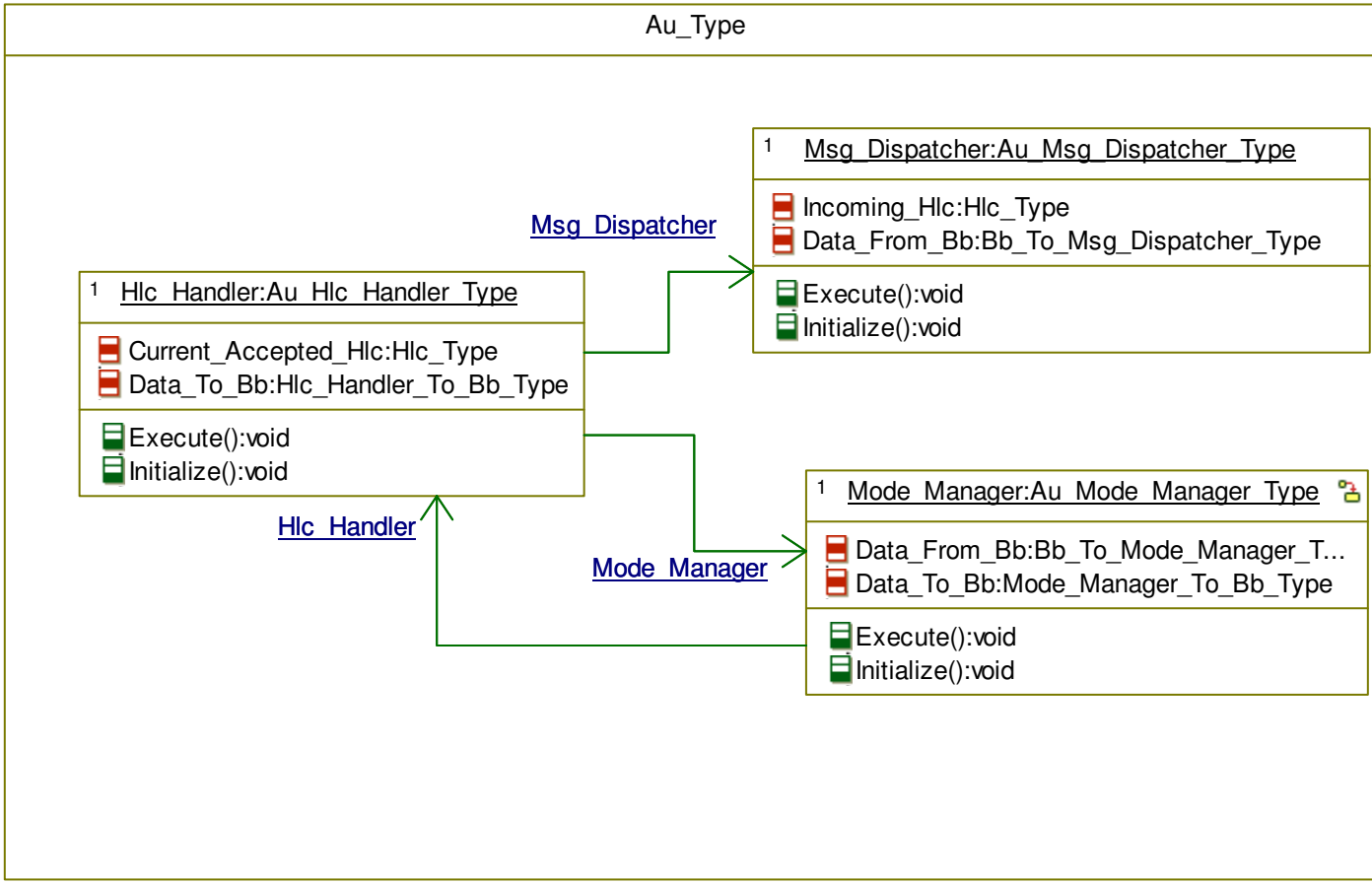
+ Vi hade ledningens stöd. Detta har inte alltid varit fallet i tidigare projekt.
+ Man satsade på utbildning och aktivt mentorskap
+ Common Core(CC) var utgångspunkten.
+ En integrerad modell-testmetod infördes
+ En väldefinierad mjukvaruarkitektur med väldefinierade komponenter infördes
+ Fokus på modellstruktur och version/konfigurationshantering
+ Vi visade att integrerad utveckling av UML komponenter tillsammans med MATRIXx eller Simulink möjlig
+ Storskalig systemmodellering
+ Mjukvarumodellering för de flesta komponenter (ca 20 st i Rhapsody och 5 i MATRIXx)
+ Kodgenerering av 92% av all kod (76% i Rhapsody, 16% i MATRIXx) av totalt ca 100 000 rader kod i Mfcu. I FscVSM dessutom ca 30 000 rader kod (Rhapsody)
+ Modelltest direkt i Rhapsody ersatte traditionell unit-test

Mindre lyckat

- Metodiken utvecklades under projektets gång
- Metodiken anpassades inte till systemutvecklingsprocessen fullt ut
- För litet fokus på spårning
- För litet fokus på verifiering
- Analysmodellen var otillräcklig

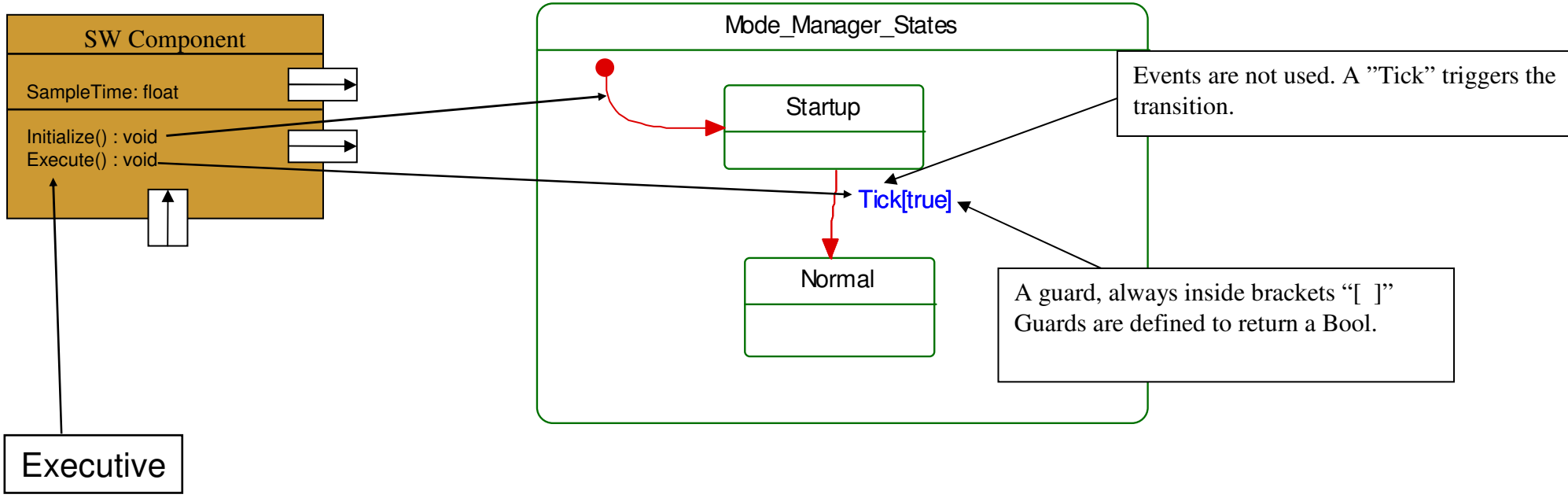
Kompositen

- Kapslar in komponentens klasser
- Bygger på statisk klassmodell
- Relationerna skapas ”automatiskt”
- Går att nästla



Beteende modelleras mha tillståndsmaskiner

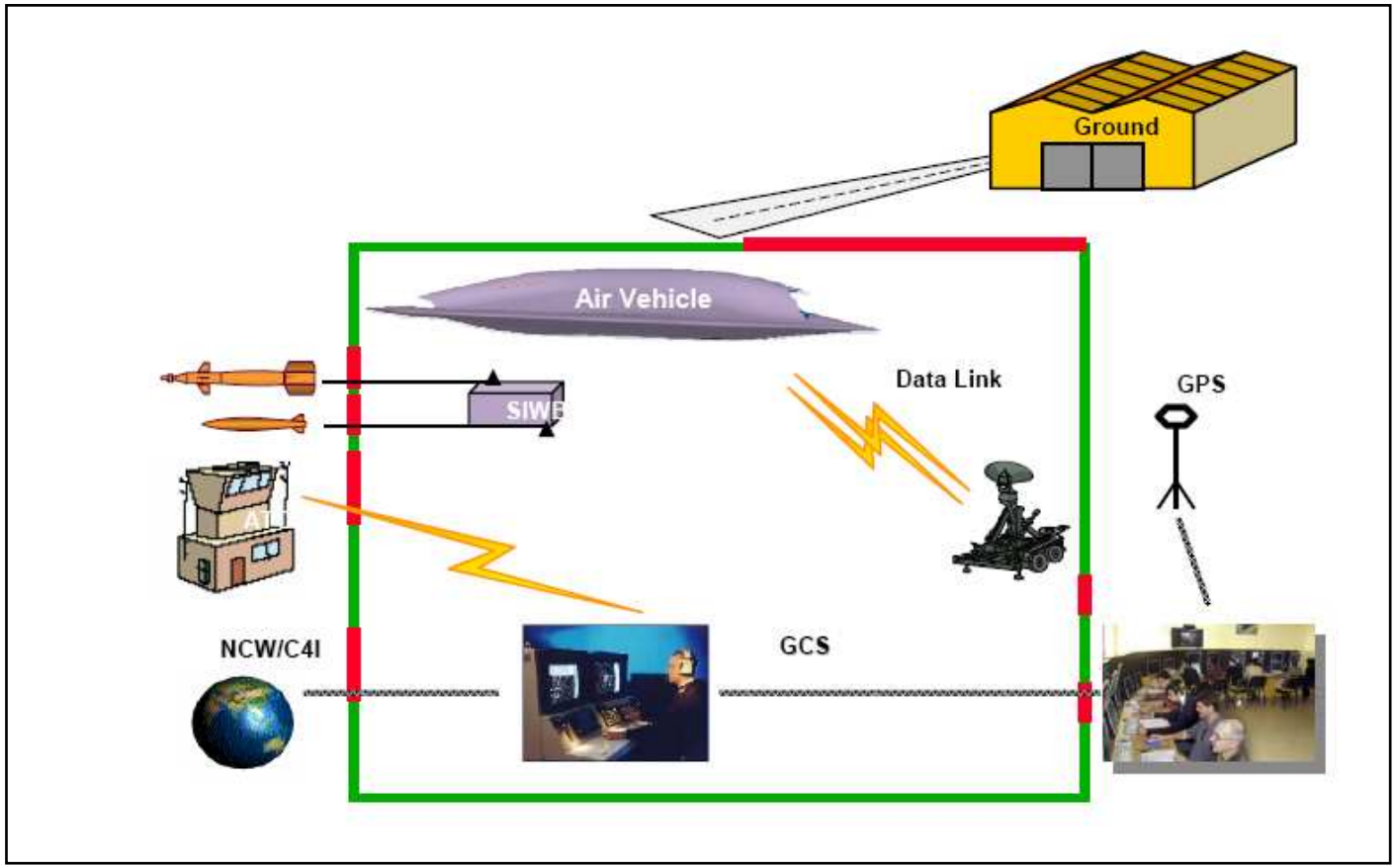
Tillståndsmaskiner används, men vissa begränsningar finns:
-Events/Messages/Köer används ej. Istället triggas maskinen av ”Tick” som styrs av ”Executiven”
-Ger fullständigt förutsägbar exekvering. Detta betyder tex. att max exekveringstid kan mätas.
-Nästlade tillstånd kan användas. Och de flesta andra konstruktioner.
-Tillståndsmaskiner kan inte kommunicera direkt med andra tillståndsmaskiner



Neuron

- Metodikkedja
- Erfarenheter

Neuron



SAAB

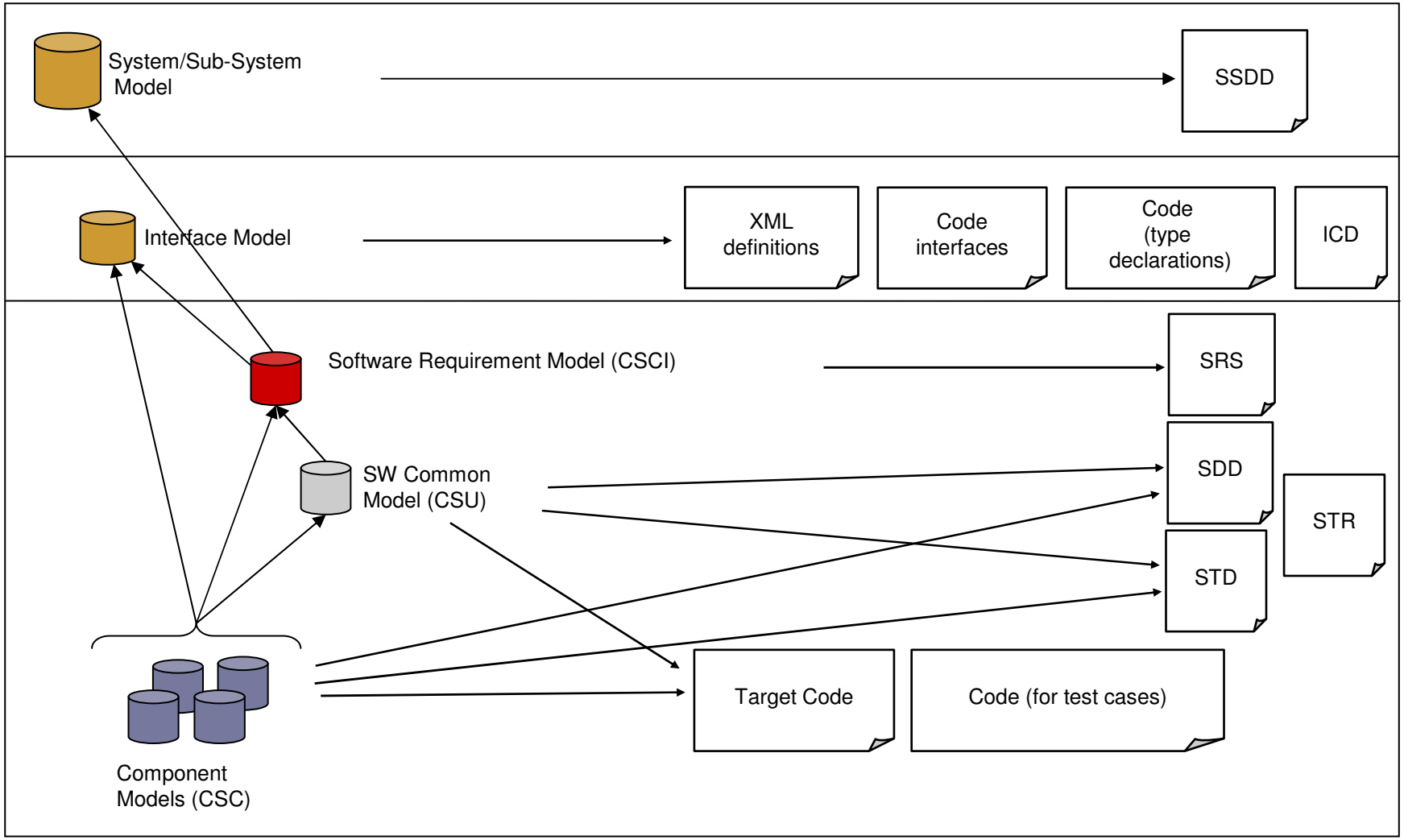
Bra saker som vi åstadkom (1)

+ Fokus på spårning
+ Fokus på verifiering
+ Generell informationsmodell
+ Verktygen hänger ihop

Mindre lyckat

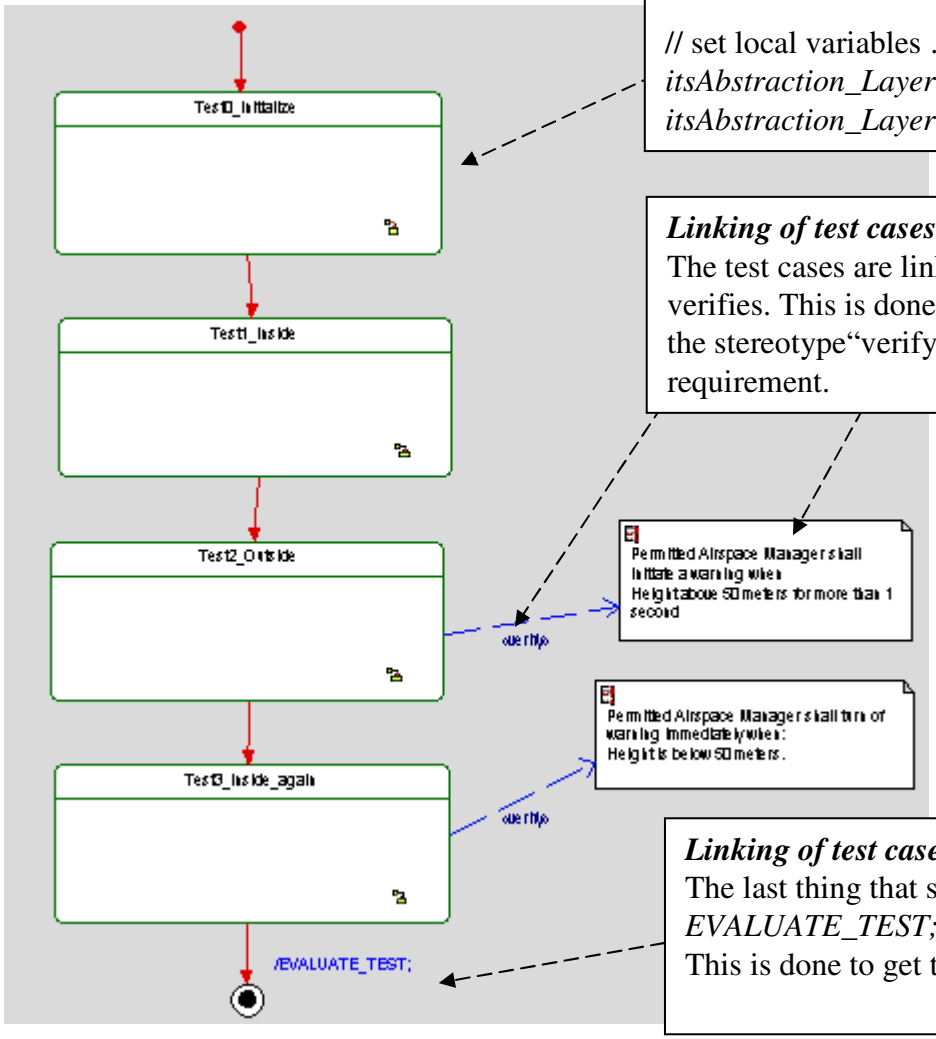
- Metodiken utvecklades under projektets gång
- Metodiken anpassades inte till systemutvecklingsprocessen fullt ut
- Systemmodellen är otillräcklig

Användning av Modeller



SAAB

Exempel – Komponenttest i Rhapsody



INIT_TESTS*(“Checking PAM function “);*

// set local variables …
itsAbstraction_Layer->getReadPort(name, inhandle);
itsAbstraction_Layer->getWritePort(name, outhandle);

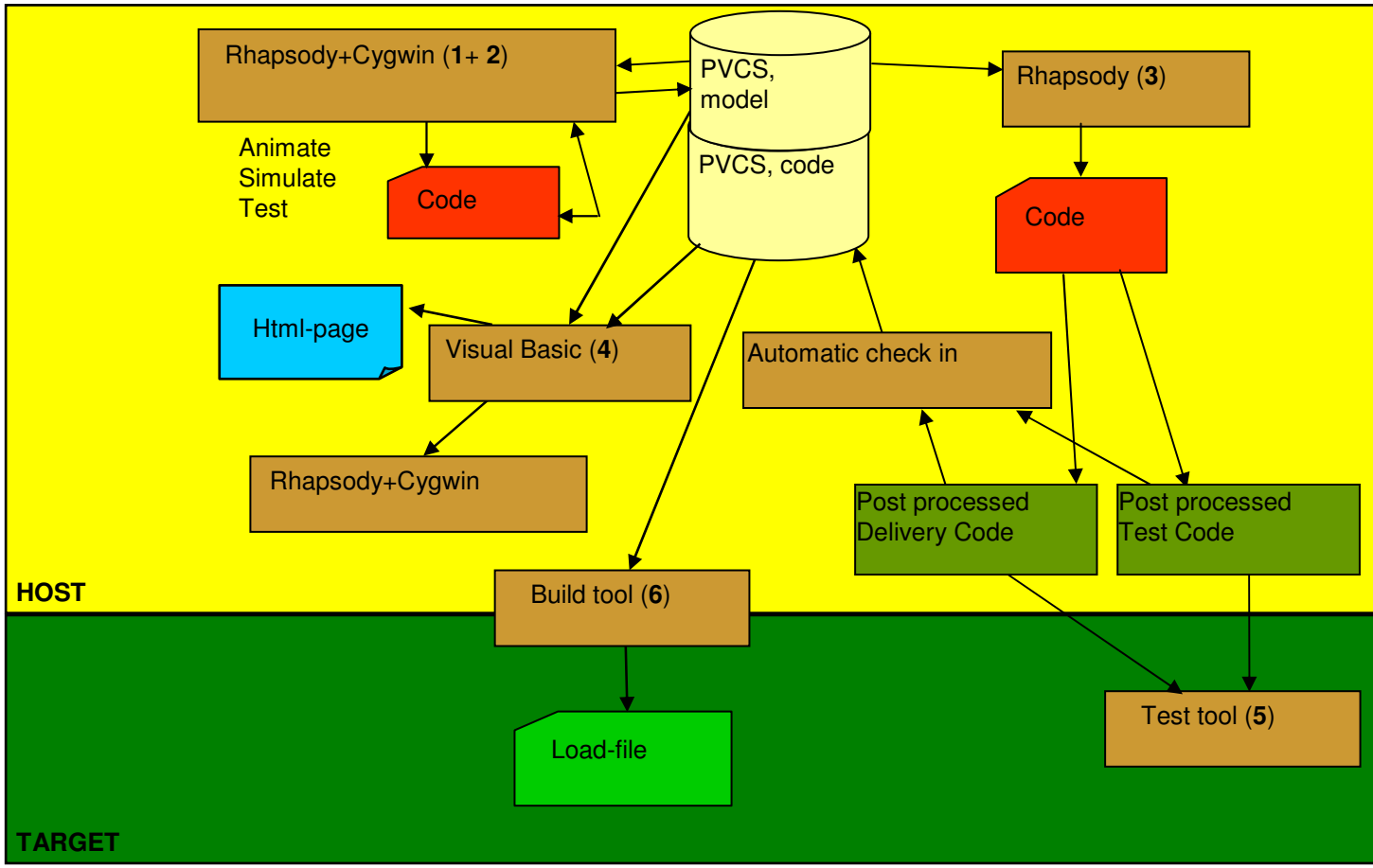
Linking of test cases
The test cases are linked to the requirements that it verifies. This is done by drawing a dependincy with the stereotype“verify” from the test case to the requirement.

Linking of test casesEvaluation of test suite
The last thing that shall be done is:
EVALUATE_TEST;
This is done to get the test summary.

Simulering, testning, automattestning och leverans av kod

Development flows:

1. Develop and test in Rhapsody
2. Postprocess and test on host including measuring of code coverage
3. Delivery of code (check in)
4. Automatic regression test, daily build/daily test on host
5. Formal tests on target
6. Build the code for target

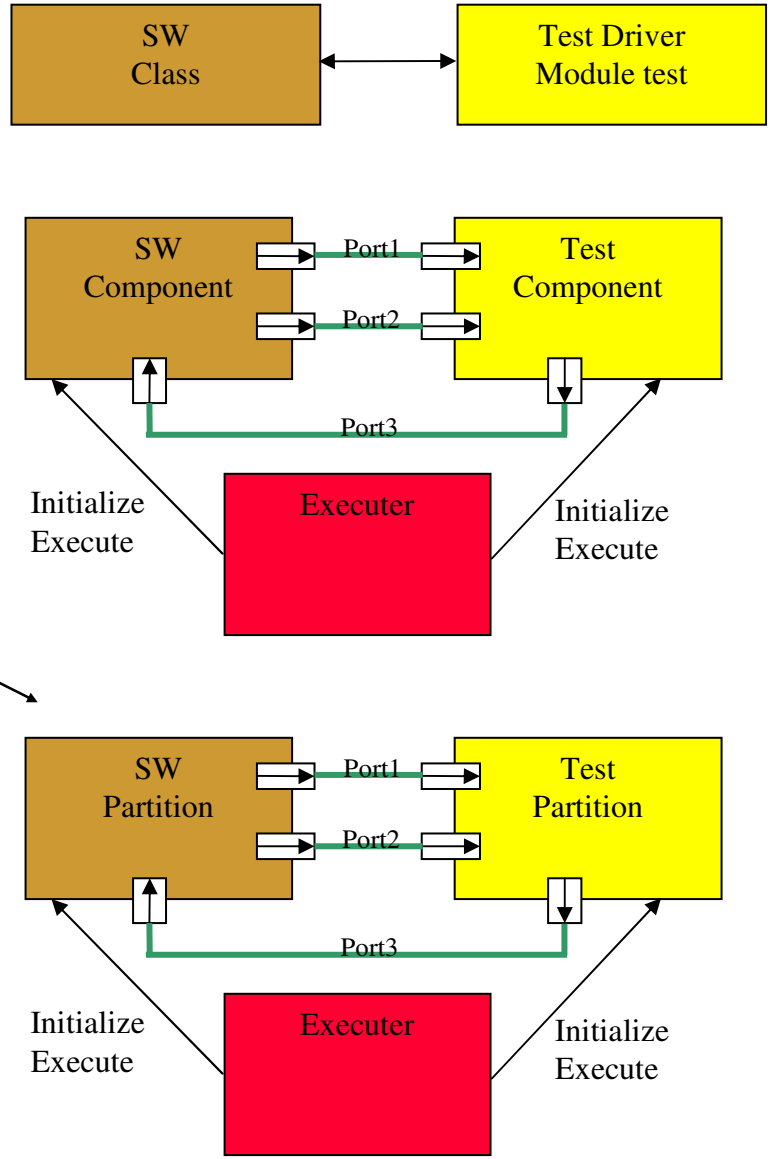


SAAB

Test principle

Testnivåer

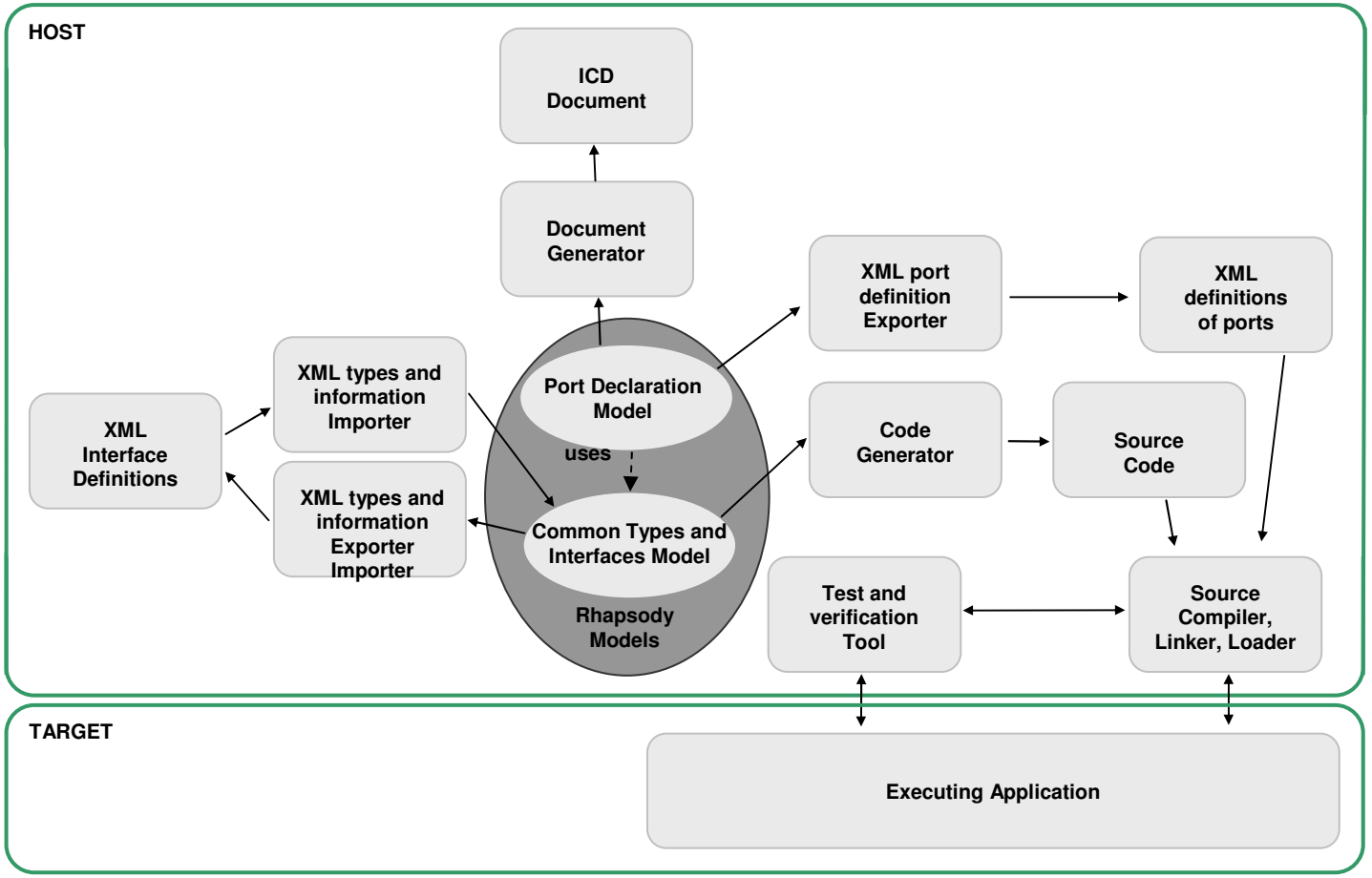
1. Module test
2. Component test
3. Partition test



- Alla tester på host först
- Alla komponenter är passiva. De kan ha beteende definierat i tillståndsmaskiner
- Alla gränssnitt är väldefinierade i port-modell
- Testkomponenter på alla nivåer. Implementerade som tillståndsmaskiner

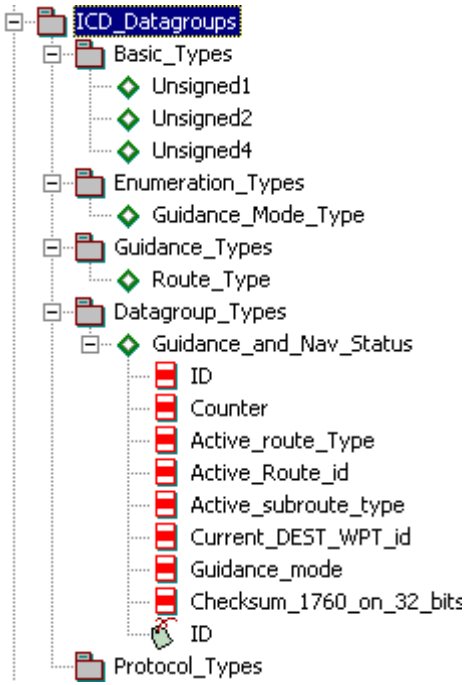
SAAB

Informationsmodell



SAAB

Informationsmodellering(1)

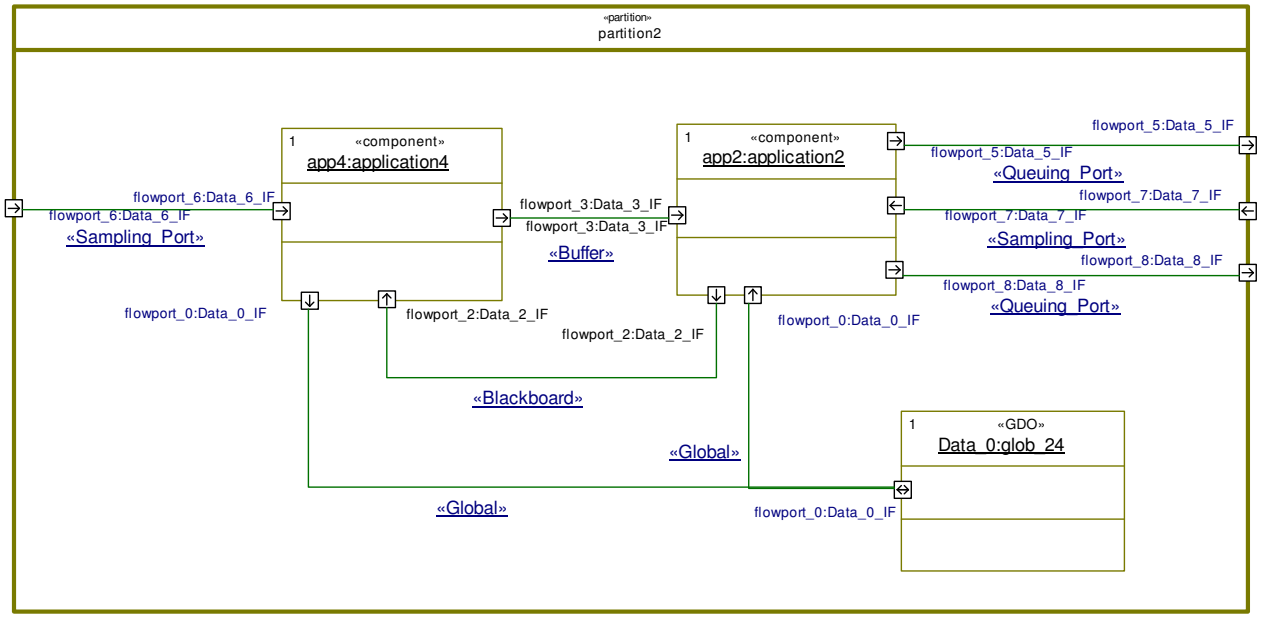


Gen av kod →

Import/ Export
XML / Rhapsody ↔

SAAB

Informationsmodellering(2)



Gen av XML till plattform

Gen av port
Och modelldata

System Development Using Rhapsody

Metodikinnehåll:

Dokument

- System Development Using Rhapsody (SyDRhap)
- Rhapsody Software Modeling Guide for Component Based Models
- Rhapsody Software Testing Guide for Component Based Models
- SysML/UML System Modeling Guide
- Information modeling and exchange in XML and Rhapsody
- Automatic Software Testing based on Rhapsody
- Document Generation Guideline from Rhapsody
- Using Rhapsody Generated Code and applying to C++ coding Standard
- Configuration Management using Rhapsody and Dimensions
- Useful Scripts developed for Rhapsody

Skript/Verktyg/Mallar mm

Dokumentgenereringsmallar

- SSDD (Gripen NG)
- SRS, SDD, STD, IRS (Neuron)

Sammanfattning

- Införande tar tid. Många är berörda.
- Ständig förbättring krävs hela tiden
- Utan visioner kommer man ingenstans!
- Förändringar blir ofta mer övergripande än man först tror

Nivåer på test/verifiering

Metodik	**Tillför**
ü Testmetodik	ü Komponenttester
ü Kodtäckningsmetodik	ü Täckningsgrad synliggörs
Komponent-testmetodik	Heltäckande Komponenttester
Automattestning	Helheten synliggörs
Daily build/Daily test	Design och test vävs ihop. Kvaliteten ökar
Integrationstestmetodik	Integrationen av komponenter förenklas
Test på target	Hitta kompilatorberoende fel. Hitta realtidsrelaterade problem. Kodtäckning på objektnivå

SAAB

MMM = Model Maturity Model

Detta är en analogi som kan vara bra att jämföra för de som känner till CMM/CMMI. Enligt CMM finns 5 mognadsnivåer:

Nivå 1: Hjältenivån. Det finns några få entusiaster som använder modeller. De är som små öar i ett väldigt hav.

Nivå 2: Någon/Några börjar beskriva metodkedjor för arbetet och användningen av modellerna.

Nivå 3: Metodkedjorna ensas ihop med varandra och de processer som företaget använder för systemutveckling.

Nivå 4: Modellanvändningen är integrerad med projektstyrningen.

Nivå 5: Optimerad användning av modeller. Kombinationer av verktyg och metodkedjor används på ett optimalt sätt.

Lärdom: Försök inte gå för snabbt fram. Det tar flera projekt för att optimera modellanvändningen. Men det krävs framförallt ett långsiktigt mål.