Vore leverandører

ARM

KEIL™ Tools by ARM

LAUTERBACH DEVELOPMENT TOOLS

Advanced Business Partner IBM

TASKING

Kontakt os

Om Nohau Danmark A/S

Nohau Danmark A/S er et datterselskab af det svenske Nohau Solutions AB.

Vi blev stiftet i 1991 og har dermed mere en 20 års erfaring med udviklingsværktøjer og processer til udvikling af embeddede løsninger.

Derfor er vi godt rustet til at hjælpe vores kunder med at finde løsninger, som sænker omkostningerne og øger produktiviteten i arbejdet med udvikling og test af embeddede løsninger. Det kan være ved opstart af et nyt projekt, hvor der er fokus på hele processen - lige fra specifikation til test. Mere almindeligt er det dog, at vi hjælper ved tilpasning og optimering af den nuværende proces – hvad enten det drejer sig om en optimering af økonomien, udviklingstiden eller kvaliteten.

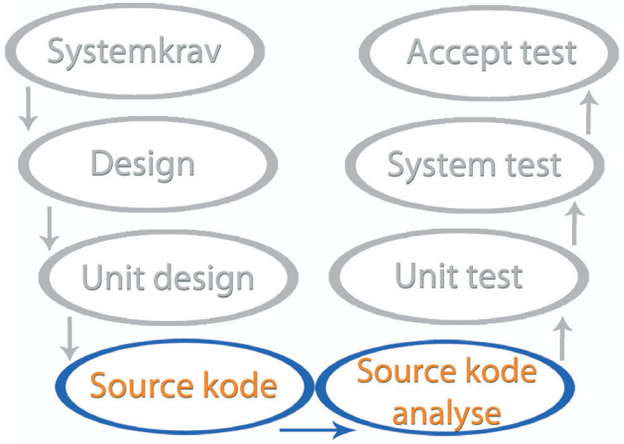
Nohau Danmark A/S
Hørkær 26, Plan 4
DK-2730 Herlev
Denmark

Phone: +45 44 52 16 50
Fax: +45 44 52 16 55

info@nohau.dk - www.nohau.dk

Nohau Danmark A/S

Implementering og Debugging af Embedded Software



nohau

Nohau tilbyder komplette løsninger

Vores ambitioner er, at hjælpe din organisation med at sænke omkostningerne og samtidig sikre den ønskede kvalitet til den fastsatte tid. Omkostningsoptimering er sjældent opnåelig samtidig med kvalitetsforbedringer og tidsmæssige optimeringer. Men ved et fornuftigt valg af værktøjer og proces, er det i de fleste situationer faktisk muligt, da automatisering kan øge effektiviteten ved samtidig sikring af kvaliteten.

Denne folder fokuserer på de af vores ydelser og produkter, som er koncentreret i den nederste del af V-modellen – nemlig værktøjer indenfor Implementering, debugging og statisk kodeanalyse.

Implementering og debugging

Med implementering mener vi kodning eller programmering af applikationen. Hvis man har lavet en præcis og komplet kravspecifikation, defineret en klar arkitektur og måske endda har valgt nogle veldefinerede kodningsregler, så burde det være trivielt – og ingen kunstart – at kode en applikation. Desværre er der ikke kun én rigtig løsning, men mange implementeringsmuligheder. Derfor introduceres der også fejl under implementeringen. Nogle fejl findes automatisk af syntakstjekkeren i compileren, mens de funktionelle fejl findes i den efterfølgende testfase. Årsagen til disse fejl kan så lokaliseres ved debugging, og fejlrettelserne skal igen implementeres. Der er flere måder at optimere disse iterationer i projektforløbet.

Moderne editorer og optimerende kompilere i et integreret udviklings- og byggemiljø sikrer hurtigt og let implementeret kvalitetskode med minimal størrelse og hurtig afvikling. Med avancerede debuggere med realtidstrace kan man hurtigere afdække årsagen til eventuelle funktionelle fejl, og de kan benyttes til realtidsoptimering og validering af performance. Statisk kodeanalyse kan i implementeringsfasen automatisk finde komplekse fejl og sårbarheder, ligesom de kan verificere at valgte kodningsregler overholdes.

Kode kan også implementeres automatisk af kodegenererende værktøjer, eller koden kan købes færdig i form af tredjeparts middleware. RTOS', kommunikationsprotokoller og fil-systemer er typiske eksempler på dette.

Kodegenererende værktøjer

UML/SysML værktøjerne IBM Rational Rhapsody findes i flere varianter. Fra "low Cost" Architect versioner til den fulde "Developer" pakke med automatisk generering af den komplette applikationskode (ikke blot kodeskaller som i Architect-versionerne). Modellerne kan eksekveres og testes løbende under udviklingen, hvilket tidligt eliminerer arkitektur- og design-fejl. UML/SysML-komponenter lavet i IBM Rational Rhapsody kan testes på modelniveau i værktøjet. Testcases beskrives i form af UML-diagrammer (Sekvens, State og/eller Flow), som "afspilles". Resultatet kan ses som animerede state charts eller i form af autogenererede sekvensdiagrammer. Forskellene til tilsvarende diagrammer fra designfasen kan derefter fremhæves automatisk og grafisk. Fordelen er, at både design, implementering (ved automatisk kodegenerering) og test foretages i ét værktøj under et iterativt udviklingsforløb.

Med "Rhapsody Design Manager" integreres Rhapsody med de øvrige produkter på IBM's Jazz-platform, så sporbarhed vedligeholdes mellem f.eks. modeller, test cases, krav og kode. Dette er helt unikt, og giver fremragende muligheder for automatisk konsekvensanalyse ved ændringer et af stederne.

Integrerede udviklingsmiljøer og build tools

Keil MDK-ARM, der er ARM's eget integrerede udviklingsmiljø for microcontrollere, kan et langt stykke hen ad vejen generere kode automatisk ud fra wizards til konfiguration af projektet. Det er ikke kun opstartskode, men f.eks. også kode til de enkelte tasks i form af templates, hvor kun den egentlige funktionalitet skal skrives ind. I "MDK-ARM-Professional" medfølger middleware-biblioteker til USB-host og device, TCP/IP netværks protokoller, Filsystemer og GUI-software m.m. Alt sammen konfigureres ved hjælp af disse wizzards, hvorved en projektplatform er klar i løbet af få timer. Et arbejde der ellers tager uger. Der er også inkluderet debugger, som ved anvendelsen af proben, "ULINK-PRO", giver mulighed for ubegrænset realtids-trace streamet til PC-ens harddisk. Herved kan der laves code coverage analyse ved kørsel af systematiske tests.

ARM DS-5 er et tilsvarende Eclipsebaseret værktøj, hovedsageligt beregnet for udvikling til større applikationsprocessorer som Cortex-A m.fl. Den tilhørende debugger-probe DSTREAM med 4 Gbit trace memory er lynhurtig og prisbillig.

Både DS-5 og MDK-ARM fås med Qualification Kit og præcertificeret kompiler.

Til andre arkitekturer såsom PowerPC, Tricore, Aurix, C166 med flere har Tasking og Freescale et bredt udbud af IDE'er med kompiler, debugger og middleware.

Debuggere

Debuggere, der er integreret i udviklingsmiljøet, giver den fornødne funktionalitet til fornuftige penge. Har man behov for mere sofistikeret analyse af den opsamlede trace-data, så er Lauterbach Trace32 uovertruffen. Eksempelvis kan man med Lauterbach's "Context Tracking System" genskabe memory og registerindhold, som det var, da en given fejl opstod for mange minutter siden. Man kan så debugge i historikken - dette er helt unikt.

Lauterbach Trace32 er et modulopbygget koncept til stort set alle arkitekturer. Det kan udbygges med forelsempel logikanalysatorer eller opsamle data om effektforbrug som løbende korreleres med den opsamlede trace data. Dette giver en enestående mulighed for optimering af batterilevetid. Af andre funktioner kan nævnes Kernel Awareness for næsten alle operativsystemer, Multicore Debugging, Cash Analyse osv. - mulighederne er uendelige.

Statisk kodeanalyse

Hvor man ved unit test eller systemtest sikrer, at en applikation gør, hvad der forventes, så sikrer man sig med statisk kodeanalyse, at applikationen ikke gør noget uventet. Den bliver herved robust.

PRQA fra Programming Research er et sæt af værktøjer til statisk kodeanalyse. De fokuserer på tjek af kodningsregler som MISRA, HIC++, JSF m.fl. PRQA kan også lave såkaldt Program Flow Analysis for at finde egentlige fejl. Dette værktøj benyttes især ved applikationer, der skal certificeres.

Codesonar har større præcision, hvad angår egentlig "bug detection" og der er derfor meget få "false positives". Det er yderst effektivt, at værktøjet selv fortæller, hvilket vej applikationen skal gennemløbe, før en give fejl opstår. Dette kan spare timevis af analysearbejde. Samtidig kan der tjekkes for overholdelse af Misra-standarden og "security issues", ligesom der kan udføres forskellige former for arkitekturanalyse og beregning af flere metrikker.

Begge værktøjer bidrager både til effektiviteten og kvaliteten. PC-Lint fra Gimpel Software er et kosteffektivt værktøj med lavere præcision.

Værktøjerne bør anvendes i implementeringsfasen, så fejl kan findes så tidligt i processen som muligt og hermed bidrage til effektiviteten.