# Realtidsindustrins syn på verktyg för exekveringstidsanalys

Andreas Ermedahl
Institutionen för Datorteknik
Uppsala Universitet, Sverige
email: ebbe@docs.uu.se

Jan Gustafsson
Institutionen för Datateknik
Mälardalens Högskola, Sverige
email: jgn@mdh.se

## 1 Introduktion

*Advanced Software Technology* (ASTEC) är ett kompetenscentrum vid Uppsala universitet som gemensamt finansieras av NUTEK, ett flertal industripartners och Uppsala universitet. Inom ASTEC drivs ett antal projekt med inriktning mot programvaruutveckling. I ett av dessa projekt, ART, bedrivs i samarbete med IAR Systems AB i Uppsala och Mecel AB i Göteborg forskning om metoder av säkerhetskritisk realtidsprogramvara. En central komponent i detta arbete är utveckling av metoder för exekveringstidsanalys. Sådana metoder är en förutsättning för att kunna analysera och förutsäga ett programs realtidsbeteende.

Som ett led i vår strävan att fokusera forskningen på industriellt intressanta frågeställningar skickade vi under våren 1997 ut en enkät till svenska företag som utvecklar realtidsprogramvara. Enkäten syftade till att inventera industrins behov av och intresse för exekveringstidsanalys.

Med i enkäten fanns en kortare beskrivning över vad exekveringstidsanalys innebär, state-of-the-art inom forskningen av exekveringstidsanalys, ett utkast till ett tänkt verktyg samt ett tjugotal frågor. Svaren på dessa frågor samt de svarande personernas egna idéer och funderingar utgör underlaget för denna enkätsammanfattning.

## 2 Sammanfattning av enkätsvar

### Låg svarsfrekvens

Av ca. 3000 utsända formulär blev endast 55 besvarade. Detta innebär att det finns ett stort mörkertal, och att vi inte kan uttala oss om vad de som ej svarade anser. Vi vet inte heller om vi nått alla företag som sysslar med realtidsprogrammering. Vad vi kan säga är att det finns 55 företag med tillräckligt stort intresse för att svara och att de tycker på det sätt som vi redovisar. Vad vi möjligen kan anta är att de som svarat är representativa för de företag som arbetar aktivt med realtidsprogrammering, och att vi därför kan dra vissa slutsatser om tendenser, speciellt om de som svarar är väldigt överens.

### Exekveringstidsanalys

70% av företagen som svarat gör regelbundet exekveringstidsanalys för att kunna verifiera realtidskraven. Man använder också exekveringstidsanalys för att optimera program och för att jämföra olika algoritmer och hårdvara. Alla som utför exekveringstidsanalys gör det manuellt, med mätningar eller beräkningar för hand. Nästan alla svarande, (94%), skulle vilja ha ett verktyg för att slippa detta manuella arbete.

### Funktioner i ett verktyg

Man vill ha ett verktyg som beräknar kortaste och längsta exekveringstid för ett program eller del av program mot viss hårdvara och kompilator. Man vill ha indataberoende beräkningar, markering av exekverad kod, kontroll av tidsbudgetar, timerpunkter, m fl av de funktioner som föreslogs i enkäten. Här krävs nog mer djupgående diskussioner med intresserade användare för att verkligen hitta funktioner som upplevs som relevanta.

### Begränsningar i C

Tanken på att införa begränsningar i C för att förbättra realtidsegenskaper och exekveringstidsanalys fick dåligt gensvar. Endast ca. 30% av de som svarade kunde tänka sig detta. Rekursion och goto-satser var de delar av språket de flesta kunde avvara.

### Sammanfattning

Sammanfattningsvis kan vi konstatera att det finns ett klart intresse för och behov av ett verktyg som beräknar exekveringstider.

## 3 Enkätsvar

### Fråga 1

Har Du tidigare stött på begreppet exekveringstidsanalys?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 37 | 18 | 0 | 55 |
| 67% | 33% | 0% | 100% |

En klar majoritet av de tillfrågade känner till begreppet. Av de som inte hört begreppet har flera ändå arbetat med analys av exekveringstider, även om man har kallat det något annat.

### Fråga 2

Utför Ert företag regelbundet (manuell) exekveringstidsanalys?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 37 | 17 | 1 | 55 |
| 67% | 31% | 2% | 100% |

Du själv?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 38 | 16 | 1 | 55 |
| 69% | 29% | 2% | 100% |

Cirka 70% av de tillfrågade företagen analyserar regelbundet exekveringstider. Under fråga 3 redovisas syftena för att få fram dessa tider.

Typiska program man analyserar är:

- mindre assemblerprogram
- kritiska kodavsnitt
- tidskritiska applikationer
- interrruptrutiner
- DSP-program i assembler

De metoder som används är:

- emulatorer
- simulator med inbyggd eta (exekveringstidsanalys)
- mätning med logikanalysator
- mätning med oscilloskop och portadresseringar
- med inlagda timer-anrop
- att manuellt räkna på assemblerinstruktioner
- profileringsverktyg

Man kan notera, att ingen av företagen har använt något verktyg. Detta är inte så underligt, eftersom de enda verktyg som finns (vad vi vet) är forskningsprototyper.

De problem man redovisar är:

- svårt med större program i högnivåspråk
- maxtider är svåra att mäta, beroende på interrupts etc.

## Fråga 3

Om Du svarade ja ovan: Vad/vilka är Ert/Ditt främsta skäl till att utföra exekveringstidsanalys?

Syftet med att få fram tider är:

- att hitta avsnitt som tar lång tid
- optimering
- att se om en processor har tillräckliga prestanda
- responstidsanalys
- att kontrollera (verifiera) realtidsprestanda
- det behövs för samplade (cykliska) system
- att säkerställa att exekvering hinns med
- att kontrollera att programmen inte tappar indata
- att säkerställa kapacitetsmarginaler
- att förutsäga exekveringstid
- att klara schemläggningskriterier
- att jämföra algoritmer och välja beräkningsmetoder

Man kan dela in de olika syftena som givits i fyra huvudsyften:

1. Att uppfylla realtidskrav
2. För optimering
3. För att jämföra algoritmer
4. För utvärdering av hårdvara.

## Fråga 4

Bedömer Ni att det vore värdefullt att ha ett verktyg för exekveringstidsanalys, med funktioner ungefär som på föregående sida?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 52 | 2 | 1 | 55 |
| 94% | 4% | 2% | 100% |

Praktiskt taget alla som svarat vill ha ett verktyg för beräkning av WCET med statisk analys. Skälen är flera:

- värdefullt för att slippa mäta
- för att kunna koda och direkt se exekveringstid för olika processorer, frekvenser etc.

Någon tyckte dock att de hade så enkla program att det går bra manuellt. Ett verktyg får inte heller kosta för mycket.

## Fråga 5

Är genomsnittsberäkningar intressanta, dvs med en specificerad indatamängd, vad får vi för medelvärde på exekveringstiden då vi exekverar programmet mot alla indata?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 38 | 10 | 7 | 55 |
| 69% | 18% | 13% | 100% |

De flesta tycker genomsnittsberäkningar kan ge något, men många kommenterar att de är mest intresserade av min och max-tider.

En av de svarande konstaterar att det är svårt att göra en genomsnittsberäkning på realtidsdata.

## Fråga 6

Vill Ni ha indataberoende beräkningar, dvs vilken exekveringstid får programmet med vissa indata?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 50 | 2 | 3 | 55 |
| 91% | 4% | 5% | 100% |

Stor enighet om att detta är ”ett måste” eftersom indata påverkar exekveringstiderna. Det måste vara enkelt att specificera indata, t ex med intervall eller tabeller. Faktiskt använda indata är ofta en liten delmängd av alla möjliga. Det kan också vara intressant att veta: vad är indata vid en viss exekveringstid? Som indata måste även ovanliga händelser som avbrott och värden vid olika systemmoder kunna anges.

Man kan konstatera att dessa funktioner, som upplevs som nödvändiga, saknas i de flesta forskningsprototyper idag.

### Fråga 7

Skall man kunna ange tidsbudgetar för kodavsnitt, funktion eller program? Varning genereras om programmet kan överskrida budgeten.

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 43 | 8 | 4 | 55 |
| 78% | 15% | 7% | 100% |

Denna möjlighet uppfattades som värdefull av drygt tre fjärdedelar av de tillfrågade.

### Fråga 8

Är följande funktion intressant: med specificerade indata, för vilka indata överskrider programmet tidsbudgeten?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 38 | 13 | 4 | 55 |
| 69% | 24% | 7% | 100% |

En majoritet uppfattade denna funktion intressant.

### Fråga 9

Är det intressant att ha möjlighet att utesluta vissa delar av programmet (t ex felutgångar) ur beräkningen vid exekveringstidsanalys?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 40 | 7 | 8 | 55 |
| 73% | 13% | 14% | 100% |

En klar majoritet uppfattade denna möjlighet som viktig, eftersom vissa delar av program ibland inte har hårda tidskrav.

### Fråga 10

Givet vissa indata, är det intressant att visa vilka delar av programmet som då kommer att exekveras?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 45 | 5 | 5 | 55 |
| 82% | 9% | 9% | 100% |

Det var en stor enighet om att detta var mycket intressant. Ideer som kom fram var att detta skulle kunna underlätta testning, för att hitta worst-case och för att använda främmande kod.

### Fråga 11

Skall man kunna lägga in ”tom kod” som bara tar en viss tid (min och max) att utföra? Detta kan användas för att studera hypotetiska fall.

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 34 | 13 | 8 | 55 |
| 62% | 24% | 14% | 100% |

Denna fråga besvarades ganska jakande (64%). Förslag på områden där detta skulle kunna användas var prototyping och att kolla prestandamarginaler.

### Fråga 12

Vilken upplösning vill man ha på exekveringstidsanalysen? Sats, basblock, kodavsnitt, funktion eller program?

Frågan konstaterades vara lite dumt formulerad då ca. hälften (49%), naturligtvis vill kunna ha så hög upplösning som möjligt. Av de som dock kunde tänka sig att begränsa upplösningen såg de flesta kodavsnitt som en lagom nivå (27%).

### Fråga 13

I vissa typer av exekveringstidsanalys identifieras död kod (dvs kod som aldrig exekveras), risk för overflow mm. “på köpet”. Är det intressant att presentera detta, i textvyn eller i den grafiska vyn?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 42 | 5 | 8 | 55 |
| 76% | 9% | 15% | 100% |

Detta tyckte det flesta (76%) var en trevlig funktion. Flera påpekade dock att kompilatorn brukar hantera detta. Frågan skulle nog formulerats så att analysen kan upptäcka mer död kod än kompilatorn normalt kan. Andra förslag på liknande analys var overflow och indexkontroll.

### Fråga 14

Skall man kunna ha flera nivåer av analys, t ex snabb, med grov beräkning, eller mer tidskrävande, med mer exakt beräkning?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 33 | 15 | 7 | 55 |
| 60% | 27% | 13% | 100% |

De flesta var postiva. Dock skilde sig åsikterna om man föredrog en snabb eller mer exakt analys. De flesta såg dock en exakt beräkning som viktigast.

Ideer som kom fram var att olika analys kunde användas i olika stadier i utvecklingsarbetet: Först en snabb under kodning och sedan en exakt för att garantera slutresultatet.

## Fråga 15

Skall man kunna lägga in annoteringar, t ex för att markera beroende mellan kodavsnitt? Exempel på annoteringar är max antal gånger man kommer gå i en loop eller att om en viss gren i en if-sats exekveras så kommer aldrig en viss gren i en annan if-sats att exekveras.

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 20 | 10 | 25 | 55 |
| 36% | 18% | 46% | 100% |

Denna fråga var svår att förstå för många (frågan har formulerats om sen enkäten skickades ut). Många (46%) svarade inget alls eller ställde sig undrande till frågan. Någon såg dock problemet vid analys av komplicerade strukturer och konstaterade att annoteringar kan vara behövligt vid bruk av C och pekare.

## Fråga 16

Om annoteringar läggs in, skall då motsvarande runtime-kontroller läggas in automatiskt? Vad skall i så fall hända om programmet överskrider beräknad exekveringstid?

Då denna fråga kan ses som en följdfråga till fråga 15 var andelen svar här ännu mindre. Några (3 st) var dock överens om att någon form av rapportering av överträdelser skall kunna genereras under (test)körningar men exekveringen skall fortsätta som vanligt. Några (2 st) påpekade att om man skall stoppa eller bara logga felet beror på den aktuella applikationen.

## Fråga 17

Skall man kunna lägga in timerpunkter? Med timerpunkter menas punkter i programmet mellan vilka tidsåtgång vid exekvering beräknas.

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 50 | 1 | 4 | 55 |
| 91% | 2% | 7% | 100% |

Detta var en funktion som nästan alla som svarade ville ha. Några hade redan idag emulatorer med motsvarande egenskaper, mest använda för hårdvaruanalys.

## Fråga 1 (programmeringsspråket C)

Bör man kan införa begränsningar i C för att förbättra språkets realtidsegenskaper och möjligheterna att beräkna mer exakta exekveringstider?

| Ja | Nej | Ej svar | Summa |
|---|---|---|---|
| 11 | 23 | 21 | 55 |
| 20% | 42% | 38% | 100% |

Denna fråga svarade inte alla på, vilket kan tolkas som att alla inte använder C som utvecklingsspråk. De flesta av de som svarade tyckte dock att inga begränsningar i språket skulle få göras. Någon tyckte till och med att C är tillräckligt begränsat i sig. Förbudstanken ogillades av vissa som såg rekommendationer (tex. genom företagets programmerarhandbok) för att uppnå bättre realtidsegenskaper som ett bättre alternativ. Om några begränsningar skall göras så var den allmänna meningen att de inte skall vara alltför stora samt vara väl motiverade. Vissa såg dock begränsningar som acceptabla om de kan öka tillförlitligheten och bättre realtidsegenskaper kan uppnås.

## Fråga 2 (programmeringsspråket C)

Vilka begränsningar skulle Ni kunna tänka Er (t ex ta bort rekursion, pekararitmetik, funktionspekare, generella while-loopar, goto-satser, etc)?

Denna fråga besvarades genom med olika "grader av borttagning". Några ogillade dock tanken på att ta bort något överhuvudtaget (2 av 17 st). De flesta som svarade tyckte att goto:s och rekursion kunde tas bort (12 av 17 st). Att begränsa pekare och pekararitmetik gillades av vissa (4 av 17 st) men ogillades av de flesta. Någon tyckte också att while-loopar kunde tas bort. Vissa former av pekarrestriktioner kunde också tillåtas av några. Det påpekades också att införa begränsningar för att analysverktyget skall fungera kommer att upplevas som irriterande.

## Övriga kommentarer

Övriga kommentarer var att man måste kunna analysera program och interruptrutiner var för sig, att man måste även kunna få exekveringstiderna från olika biblioteksrutiner som man använder samt att ett WCET-verktyg blir riktigt intressant när det integreras med en debugger. Önskemål om att analysen skulle kunna användas i olika sammanhang och då integrera olika hårdvaruarkitekturer, schedulering, interupthantering och operativsystemsanrop framfördes också.