Tillförlitlighetsmodeller: Giltighetsområde och tillämpning

Wohlin, C.

Proceedings Software Metrics and Quality Workshop,
Snogeholm, Sweden, 1986.

TILLFÖRLITLIGHETSMODELLER : GILTIGHETSOMRÅDE OCH TILLÄMPNING

Claes Wohlin, Institutionen för Teletrafiksystem, LTH

Inledning

Följande utspelade sig i en skoaffär i Lund.
En dam kommer in med två par sandaler och stegar fram till expediten.
Kunden säger:

- Min dotter och jag köpte dessa häromdagen och har använt dem en gång. Nu är det så att skinnet här på sulan har lossnat. Det tycker jag är dålig kvalitet.

Expediten reagerade omedelbart på det sista påståendet och gick i försvarsställning.

- Nej, dålig kvalitet är det inte. Det är mycket fint skinn och vi kan lätt fästa det igen.
- Det är ju bra, men jag kallar det dålig kvalitet om skinnet lossnar när man endast använt dem en gång, stod damen på sig.
- Nej, dålig kvalitet är det inte. Däremot kan det ju hända att de är dåligt gjorda !

Därmed ansåg expediten diskussionen avslutad och gick för att åtgärda felet.

Skulle expeditens resonemang kunna motsvaras av följande ? :
Om vi programmerar i ett språk av hög kvalitet (vilket specifikt språk det skulle vara lämnas därhän) så kommer också vårt utvecklade system att hålla hög kvalitet, oavsett hur vi rent praktiskt går tillväga.

Vad är kvalitet ?

I en annons för kläder får vi svaret:
"Kvalitet är en känsla".

Men svaret på frågan är kanske inte så enkelt. Vi måste också ta reda på vad som bygger upp denna känsla av kvalitet. Utgående från anekdoten ovan kan vi dra slutsatsen att kvalitet är ett mycket tänjbart begrepp och betyder olika saker beroende på vem man frågar. Detta leder till slutsatsen att det är viktigt att definiera kvalitet ur allas synvinkel, såväl kund som användare, tillverkare, tekniker och ekonomer etc.

Vi skall dock inte behandla denna frågeställning här, utan nöjer oss med att studera en mycket viktig kvalitetsaspekt; Tillförlitlighet.

Programvarutillförlitlighet

För att få en uppfattning om tillförlitligheten vill vi kunna bestämma faktorer som; återstående antal fel, tid mellan fel, det vill säga överhuvudtaget prediktera den fortsatta felutvecklingen hos programvaran. Detta kan vi göra med hjälp av tillförlitlighetsmodeller. En första uppfattning av antalet fel i programvaruprodukten kan vi få ur komplexitetsmåtten. Denna skattning kan sedan användas som indata till våra tillförlitlighetsmodeller. Resultaten från modellerna kan förbättras efterhand som projektet framskrider genom insamling av felstatistik, och med hjälp av denna kan vi förfina skattningarna av modellernas parametrar. Genom att skatta felutvecklingen är det möjligt att prediktera exempelvis en lämplig tidpunkt för att släppa programvaruprodukten samt att prediktera hur allokeringen av testresurser bör ske, för att klara av att hålla utlovade leveranstider och utlovad kvalitet.

Modellernas giltighetsområde

En realistisk prediktering av den framtida felutvecklingen är naturligtvis ett måste för att kunna dra några slutsatser från den. Detta är i hög grad beroende av att man använder tillförlitlighetsmodeller som är väl anpassade till den situation som råder. Modellerna måste vara rimliga med avseende på en rad faktorer som till exempel utvecklingsmiljö, hjälpmedel, applikation, testmiljö etc. För att finna de modeller som är möjliga att använda för våra produkter krävs en noggrann undersökning, klassificering och jämförelse av existerande modeller, samt en studie av vår egen miljö. Ett första steg mot en jämförelse av existerande modeller presenteras i (1).

Resultaten av undersökningen kan leda till:

- Ett antal modeller är möjliga.
- Ingen modell är rimlig.

Om ett antal modeller är möjliga kan vi antingen:

- Välja en av dem.
  Frågan är bara hur vi skall välja ut den. Vi skall inte välja en mer komplicerad modell än nödvändigt, utan den bör väljas med stor omsorg med avseende på den information vi vill ha ut.

- "Multi-modelling"

  Vilket betyder att vi använder många modeller och ser vad det resulterar i. Ett problem blir här att välja resultatet, då modellerna knappast kommer att ge samma svar. Och om de skulle ge samma svar, vad beror det på ? Modellerna eller att resultatet är det riktiga ? En undersökning, se referens (1), av två till synes olika modeller, "Jelinski-Moranda's De-Eutrophication Model" och "Goel-Okumoto's Non-Homogenous Poisson Process Model", gav till resultat att de i det närmaste var ekvivalenta. Modellerna antager inte samma fördelning, men anpassar antalet detekterade fel med samma medelvärde, nämligen

  $N(t) = a*(1-exp(-b*t))$

  a - initialt antal fel
  b - proportionalitetskonstant

  Detta betyder att så länge som vi inte studerar något annat än medelantalet inträffade fel vid en viss tidpunkt, ger modellerna precis samma resultat. Det krävs dock, för att kunna styra ett projekt, att vi även studerar variationen av antalet inträffade fel vid en specifik tidpunkt, detta är en problemställning som vi återkommer till under "Exempel på en tillämpning".

Båda dessa fall är områden som kräver ytterligare studier och målet måste vara att ta fram en handbok för tillförlitlighetsmodeller, där modellerna är klassificerade med avseende på tillämpning och den information vi får ut ur dem.

Skulle däremot ingen modell vara tillämplig, blir vi tvungna att utveckla egna modeller eller acceptera att vi inte kan uttala oss om den framtida felutvecklingen. Ett accepterande av det sistnämnda innebär ett stort steg i riktning mot att tappa kontrollen över produkten.

Det kanske viktigaste av allt är att inte bara ta en modell och använda den, utan att noggrannt studera dess antaganden och tillämpningsområde. Vi måste veta att modellen är realistisk för att kunna sätta någon tilltro till resultaten från den.

I (2) presenteras ett exempel på hur en undersökning av en specifik testmiljö kan leda fram till att en tillförlitlighetsmodell, som är anpassad till rådande förhållanden, utvecklas. Denna undersökning ledde också fram till att vi kan konstatera att vi inte kan förvänta oss att finna en global modell, utan att vi behöver olika modeller under olika faser i programvarans livscykel. Modellerna måste anpassas till respektive fas och när en övergång till en ny aktivitet sker måste vi acceptera att vi behöver en ny tillförlitlighetsmodell.

Flertalet av de "klassiska" programvarutillförlitlighetsmodellerna är anpassade till driftsliknande förhållanden, vilket för det mesta inte råder under test av ett system.

Det är mycket viktigt att utveckla modeller som är tillämpbara under test. Om vi har tillförlitlighetsmodeller för detta har vi en möjlighet att bestämma programvarans tillförlitlighet vid olika tidpunkter, till exempel beräknad release-tidpunkt. Därmed har vi kommit in på en mycket viktig tillämpning av våra modeller, nämligen som hjälpmedel för ekonomisk optimering av vårt programvaruprojekt.

Expempel på en tillämpning

Det är allmänt känt (och accepterat) att kostnaden för rättning av fel ökar ju längre vi har hunnit i vårt projekt, samtidigt som det (naturligtvis) inte lönar sig att testa hur länge som helst. Detta enkla resonemang leder fram till att det är uppenbart att det finns ett ekonomiskt optimum, då vi skall avsluta testen och släppa produkten. Antag att vi undersöker hur kostnaderna fördelar sig och på så sätt kan komma fram till en ekonomisk modell för våra programvaruprojekt. Det är ju sedan helt naturligt att vi vill testa tills vi uppnått detta optimum.

Det finns då två stycken principiellt olika angreppssätt när det gäller testfilosofin:

- resurserna bestämda, det vill säga vissa resurser har tilldelats projektet och det kan inte ändras.
- release-tidpunkten bestämd, det vill säga vi måste vara klara till en viss tidpunkt oavsett vilka resurser som krävs.

Den normala situationen är oftast en hybrid av dessa principiellt olika tillvägagångssätt. Eftersom vi nu antar att vi funnit eller utvecklat realistiska tillförlitlighetsmodeller för vår miljö kan vi nu beräkna

- förväntad tidpunkt för att uppnå optimum och hur den varierar, det vill säga med bestämda resurser.

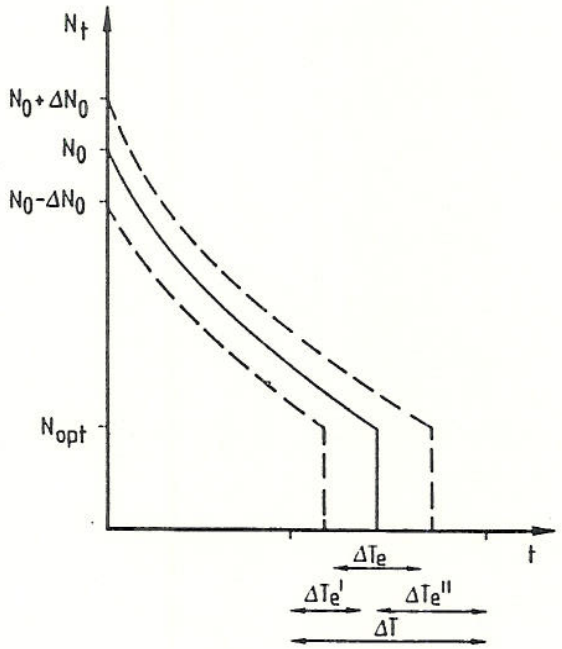
eller

- bestämma resurserna som krävs för att med en viss sannolikhet uppnå vår utlovade release-tidpunkt.

Vi kan naturligtvis genomföra ovanstående beräkningar med alla tänkbara kombinationer av resurser och release-tidpunkter. Utgående ifrån resultaten kan vi dra slutsatser om vilka åtgärder vi behöver sätta in för att klara av de krav som ställs på vår programvaruprodukt, både vad det gäller tillförlitlighet, tid och ekonomi.

En oerhört viktig aspekt angående beräkningar ovan, är att vara medveten om och planera för de stokastiska variationer som vi har i uppträdandet hos programvarufel. Det är inte realistiskt att bara räkna med medelvärden utan vi måste också ta hänsyn till varianserna. I figur 1 visas det principiella resultatet som är en direkt följd av den stokastiska process som ligger bakom programvarufelen och dess upptäckt. Följande definitioner har använts:

* N_t - antal fel vid en godtycklig tidpunkt t
* N_0 - initialt antal fel
* ΔN_0 - osäkerheten i N_0
* N_{opt} - antal återstående fel då vi har uppnått det ekonomiska optimum
* ΔT_e - osäkerheten i tidpunkten då vi uppnår det ekonomiska optimum
* ΔT - den totala osäkerheten i tidpunkten då vi uppnår det ekonomiska optimum



Figur 1 Antalet återstående fel i programvaruprodukten och variationen hos tidpunkten då vi uppnår vårt ekonomiska optimum.

I figur 1 ser vi hur antalet fel initialt varierar inom intervallet (N_0-ΔN_0, N_0+ΔN_0) och hur de sedan avtar med tiden. Vi ser hur tidpunkten då vi når N_{opt} varierar för de tre fallen då antalet fel startar på N_0-ΔN_0, N_0 och N_0+ΔN_0. Detta ger upphov till en total osäkerhet i tidpunkten som motsvarar ΔT.

ΔT är med stor säkerhet alldeles för stort och det krävs en rad åtgärder för att komma tillrätta med problemet. En noggrannare undersökning av konsekvenserna av olika fördelningar för tid mellan fel presenteras i (3), där även problemet, beräkningar och åtgärder diskuteras närmre. Vi kan dock konstatera att vi har ett antal tänkbara åtgärder, för att med större säkerhet kunna uttala oss om när vi uppnår det ekonomiska optimum, och de är

- minska initialt antal fel
- bättre skattning av initialt antal fel
- bättre testmetoder

Om vi inte lyckas med detta finns två möjligheter, båda oekonomiska

1. Vi testar för länge, för att vara på den "säkra" sidan.
2. Vi avslutar testen för tidigt, vilket kan leda till en ineffektiv, otillförlitlig, okontrollerbar och ej underhållsbar programvaruprodukt.

Avslutning

Detta är en tillämpning som kanske inte är möjlig idag, men den visar på tillförlitlighetsmodellernas potential så som hjälpmedel för styrning och kontroll av ett programvaruprojekt. Modellerna kan fungera som objektiva mått och stödja projektledningen. Till exempel när det gäller planering av tid, resurser och ekonomi, för utveckling av programvaruprodukter av hög kvalitet.

Referenser

(1) Wohlin, C., 1986, "A Comparison Between Two Software Reliability Models: Jelinski-Moranda's De-Eutrophication Model and Goel-Okumoto's Non-Homogenous Poisson Process Model", Technical Report, Lund Institute of Technology, Lund, Sweden.

(2) Wohlin, C., 1985, "Software Testing and Reliability for Telecommunication Systems", Technical Report, Lund Institute of Technology, Lund, Sweden.

(3) Wohlin, C., and Vrana, C., 1986, "A Quality Constraint Model to be Used During the Test Phase of the Software Lifecycle", Proc. 6th Int. Conf. on Software Engineering for Telecommunication Switching Systems, Eindhoven, The Netherlands.