

Denna artikel är en längre version av nyhetsbrev nr 3:2000 i serien Systemkonstruktion
Programvara utgiven av Sveriges Verkstadsindustrier, VI. www.vi.se

Extreme Programming:

Programvara får kvalitet – när utvecklarna själv får välja

Är din verksamhet beroende av programvara av god kvalitet? Alltfler är helt beroende av omfattande programvara för att produkter och organisation skall fungera.

Att utveckling av programvara ofta innebär stora risker är välkänt. Budgetutfall och kvalitet varierar minst sagt. Varför blir inte programvara bättre? Är det utvecklarens eller organisationens fel?

Nu tar utvecklarna saken i egna händer. Extreme Programming, XP har fått häpnadsväckande genomslag bland erfarna utvecklare. XP riktar sig direkt till de som gör jobbet. XP blandar avancerad teknik och stenhårt disciplinerad process med snabb feedback. Utvecklarna, idag en bristvara, ställer upp på den tuffa arbetsordningen då de som team ser chansen att få resultat och kvalitet som kan vara stolta över. Dock måste resten av organisationen bli lika bra annars blir obalansen akut.

IBM, HP, Nokia, Motorola, Ericsson mfl ser, efter många år av tunga kvalitetssatsningar på programvara, på möjligheten att genom XP ge den stora gruppen medarbetare, utvecklarna, chansen att själva garantera kvaliteten i programvaran. Flertalet företag arbetar sedan länge med delar av XP, då allt är kända metoder som kombinerats ihop så att de kompenserar varandras brister.

XP handlar om självorganiserande team som arbetar oerhört lyhört för beställarens önskemål med korta cykler och ständig uppdatering av prioriteter. Att kravspecifikationen förändras med beställarens förståelse är ett normalt inslag i XP. Många programvaruprojekt påbörjas innan kravbilden är klar för att bli klara i tid. Exempelvis produkter som baseras på nya standarder måste utvecklas medan standarden fastställs.

SENASTE NYTT!
SPIN-SYD startar XP-grupp!
Se sista sidan!

Introduktion

Extreme Programming, XP, är en hårt disciplinerad lättviktsprocess för programvaru-utveckling som betonar teamwork, kommunikation, feedback, enkelhet och immersiv problemlösning. XP fokuserar på att få ett team att fungera med ett begränsat antal medlemmar. Teamets storlek begränsas bl a av att behovet av skriftlig kommunikation och tillfälliga dokument tenderar att växa exponentiellt i större grupper. Med begreppet lättviktsprocess menas främst en process med låg overhead i förhållande till i slutprodukten värdeskapande tid och energi. XP skapades för att hjälpa ett mindre team att snabbt utveckla stabil programvara i projekt där kravbilden ofta snabbt ändras under projektets gång

VISA skapades i slutet av 60-talet med självorganiserande lättviktsprocesser som var styrda av idéer och värderingar. Tretusen konkurrerande amerikanska banker förenades under VISA-systemets första år. Läget i kritiska projekt markerades av en odiskad kaffekopp som i ett gammalt snöre flyttades längs de whiteboards som utgjorde planen.

Karaktistiskt för lättviktsprocesser är just det starka inslaget av gemensamma värderingar och ideal som ersätter tusentals detaljinstruktioner. Idealet är att befinna sig i ett ordnat tillstånd på gränsen till kaos. Då gynnas syftet med projektet maximalt. Dee Hock arbetar idag med att hjälpa ideella organisationer att minska byråkratin med synsätt som leder till lättviktsprocesser.



XP växte fram, under Kent Becks ledning, ur design patterns-rörelsen med inslag av refactoring och influerat av utvecklingsmiljöer som Smalltalks främsta fördelar. Mycket av XPs värderingar växte fram i Hillside Group, en grupp som startade design patterns-rörelsen. Former och värderingar för XP förfinades och kompletterades i en rasande takt så snart Beck och Ron Jeffreis hade startat en publik process på Ward Cunninghams WikiWik Web. Ward Cunningham, som kände Beck från gemensam forskning på bl a Textronix, hade 1994 skapat ett unikt diskussionsforum på internet. Ett slags anslagstavla för att samla vetande och utbyta tankar. Cunninghams Wiki användes från början mest för att vaska fram design patterns. Dock frestas man att tro att Wiki har betytt ännu mer för XP än för design patterns.

Kent Beck presenterade XP på OOPSLA99 som "Turn all dials up to Ten!" D v s att göra ett antal bra saker så mycket det bara går. Enkel design var bra, vrid upp till max. Att putsa på koden var bra, vrid upp till max. Att bygga produkt ofta var bra, vrid upp till max. Beck testade fram ett dussin bra grepp som kombinerar tekniska, organisatoriska och social insikter. De flesta tekniska metoder testade han först på sig själv.

Kent Beck förenar i XP en samling Best Practices som kompletterar varandra. En viktig skillnad mot liknande processer eller metoder är det starka inslaget av konkreta tekniska lösningar som integrerat stödjer XP. Det tilltalar teknikern i många programmerare. Processorientering har ofta, i grunden, handlat om att anta ett managementperspektiv, vilket inte varit lika attraktivt alla många medarbetare. Genom att även tala till det tekniska intresset hos utvecklare skapar Beck brett stöd i hela teamet för en hårt disciplinerad process.

Extreme Programming Explained

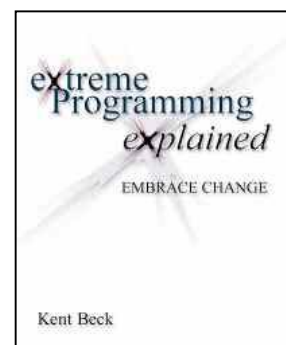
[ISBN 0-201-61641-6](#)

Kent Becks föredömligt tunna och lättlästa bok presenterar XP på ett lättfattligt sätt. Bokens viktigaste lärdom kan vara att rätt attityd och stort engagemang (men utan övertid) ersätter mycket av gängse styrmekanismer, vilka enligt Beck härstammar från ett otidsenligt Tayloristiskt synsätt.

Frederick Winslow Taylor sökte, för ca 100 år sedan under industrirevolutionen, skapa en vetenskaplig produktionsprocess där människor var en resurs precis som järn eller kol. Enligt Taylors tankar vore idealet om vem som helst kunde stiga in från gatan och göra ett visst arbete, hela tiden styrd av ytterst detaljerade instruktioner. Individerna var utbytbara.

Beck hävdar att de flesta utvecklingsorganisationer bygger på Taylors idéer med strävan att individen skall vara oviktig och utbytbar. Kan vi hitta former där individen känner sitt värde utan att få oacceptabel riskbild vid personalomsättning? XP ger en del av svaret där individen får prestera och bli erkänd i teamet, men ingen kod binds till en enskild persons kunskaper och insikter.

Företagsledningen kan vid en yttlig betraktelse avfärda XP som en invitation till hackers. Dock visar erfarenheter från bl a SPIN-SYD, ett nätverk med 30-talet sydsvenska programvaruföretag, att XP först uppmärksammas och omtalas av de mest erfarna utvecklarna. Kent Beck ser, med blandade känslor, att XP i USA används som ett attraktivt lockbete för rekrytering av erfarna utvecklare, inte minst i Silicon Valley och Seattle...



Extreme Programming Installed

[ISBN 0-201-70842-6](#)

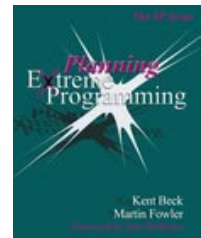
Andra boken om XP. Ron Jeffreis mfl berättar handfast om hur man kan tillämpa XP. Grunden är deras egna erfarenheter främst från Chryslers numera nedlagda C3-projekt som drevs enligt XP. Jeffreis kom dit genom Kent Beck. Beck var kontrakterad av Chrysler för att starta upp och stödja C3-projektet. C3-projektet avbröt för en tid sedan då alla i Chryslers management såväl beställare som betalare hade bytts ut. Då Chrysler i övrigt inte hade någon högre processmognad, krävdes personlig support från vd för aktuella dotterbolaget, ekonomichef och IT-chef. När Chryslers "management rotation program", avsett att ge bred erfarenhet åt chefer, hade bytt ut personerna i alla dessa tre roller, hamnade projektet i ett vakuum. Ingen förstod vad de sysslade med då C3 fokuserade på att producera programvaruprodukten, inte att ständigt lämna rapporter som kan förstås utan tidigare inblick i projektet. Till slut fanns det ingen som försvarade C3 i dess omvärld. Jeffreis och hans kollegor har sedan C3 avslutades även samlat in mycket erfarenheter från andra XP-team. Ron Jeffreis och hans kolleger anlitas mycket som coach och utbildare på XP. Man får i boken många konkreta tips på ömsesidiga sociala regler/överenskommelser, en del av XPs kärna, som varit framgångsrika för XP-team.



Planning Extreme Programming

[ISBN 0-201-71091-9](#)

Detta är den tredje boken i Addison Weseleys serie av föredömligt tunna böcker om XP. Kent Beck beskriver här tillsammans med Martin Fowler (Refactoring, UML Distilled, Analysis Patterns) hur planering, prioritering och estimering sker i XP. Man talar också om hur egna mätetal kan vara värdefulla och om kontraktsförhållanden.



XPs värderingar

Communication

Kunden finns alltid på plats i teamet. All produktionskod skrivs i par. Estimering sker i dialog med kunden.

Simplicity

Sträva alltid efter enklaste tänkbara kod. Bygg aldrig för generell kod, den blir ofta ändå aldrig använd för annat än det omedelbara behovet. Refactoring utvecklar arkitekturen inkrementellt efter behov.

Feedback

Ständig feedback genom kontinuerlig testning, kontinuerlig integration, planning game, kunden på plats.

Courage

Mod innebär här att ha mod att vara rak, att göra sitt bästa, att stiga ur den komfortabla zonen när det gynnar teamet och produkten. Både kund och management måste ha mod att ta ett större ansvar då man exempelvis inte kan luta sig mot tunga kravdokument som en gång godkänts av teamet. Teamet gör sitt bästa och berättar så snart det har problem eller inser att man underskattat någon svårighet.

XP-metoder: Tolv beprövade grepp på startlinjen

Enligt Beck är XPs tolv beprövade metoder, "best practices", en bra uppställning när man står på startlinjen att börja tillämpa XP. Dessa metoder och aktiviteter bygger snabbt insikt och erfarenhet hos utvecklarna inom nyckelområden så att de efter hand kan anpassa verktygslådan efter behov. Om man efter något år fortfarande använder allt i ursprungligt skick anser dock Beck att man verkligen inte arbetar i XPs anda. XP syftar till att team och individ, då erfarenheten växer, strävar efter att optimera/skräddarsy process och metoder efter projektens behov. En slående likhet med strävan på CMM nivå 5, den optimerande nivån.

eXtreme Programmings tolv beprövade grepp:

Vissa grepp kan man ta som individ, utan att få med hela teamet. Vissa kräver samförstånd med teamet, vissa kräver att alla i teamet ställer upp. Det framgår under varje rubrik.

Metaforen är affärsidén för det vi skall bygga? – "Metaphor"

Påverkar hela teamet då det handlar om att sträva mot samma mål...

Ett XP-projekt utarbetar först en definition, metafor, för den produkt man skall bygga. Metaforen ger en klar övergripande uppfattning om vad slutprodukten skall göra. Alla funktionella krav, i form av "user stories", skall gå att härleda till Metaforen.

Metaforen för XP-projektet kan jämföras med ett företags affärsidé. Om affärsidén är tydlig, bra och förstås av alla, kommer tusentals små dagliga beslut enkelt klaras av företagets samtliga medarbetare. Affärsidén skall alltid kunna besvara frågan. "Varför?" Om affärsidén däremot är en samling floskler som bara skall se bra ut, kommer verksamheten att behöva en enorm mängd detaljbeslut från den operativa ledningen. Därför skall en Metafor liksom en affärsidé alltid besvara frågan "Varför?" Ett XP-projekt med en intetsägande Metafor är dödsdömt.

Men inom XP används begreppet Metaphor för att ge frihet att uttrycka sig bildligt. Det innebär i sig att man är mentalt mindre bunden till att beskriva lösningen som en del av visionen. Metaforer ger också möjlighet att beskriva egenskaper genom referenser och liknelser som t ex:

"...med en kattlik smidighet".



Kent Beck använder sin "Driving Metaphor" för att förklara varför planering skall göras längs vägen. Ingen kör bil genom att noga ställa in bilen på målet med ett kikarsikte och för att därefter blint köra ett antal mil. Man justerar hela tiden längs resans gång (Ett problem med kameran gav denna bild ett metaforiskt skimmer:-)

Rak, enkel planering – ”The Planning Game”

Påverkar hela teamet.

Planering görs initialt för att skapa en iterations plan. Därefter sker planering för varje iteration. Denna planering är en gruppövning där hela teamet deltar. ”Kunden på plats” har skrivit ”user stories”, som påminner om use cases med text och enstaka enkla bilder, på kort i storlek A6. User stories är nära släktingar till CRC-cards. En userstory motsvarar en funktion att implementera i systemet. Utvecklarna bryter ofta ner ”user stories” i ”engineering tasks” och ger tidsuppskattningar för varje kort. Engineering tasks blir egna kort. Kunden prioriterar genom att sortera korten i högar. Högst upp i första högen har högst prioritet. Helheten hålls samman av den övergripande Metaforen. Varje userstory skall gå att härleda ur Metaforen. Utvecklarna åtager sig olika kort. Ingen delar ut jobb och ansvar. Man påtar sig ett ansvar, man blir aldrig påtvingad det. Jag upprepar: Varje åtagande är frivilligt, ingen tilldelar någon en arbetsuppgift. Tar någon sig vatten över huvudet lär man sig snabbt då cyklerna är korta. Ett åtagande skall aldrig vara mer än några dagar.

Utvecklarna som grupp förhandlar med kunden på plats om vad som kan klaras i den release som man planerar för. Detaljerad planering flera releaser i förväg undviks aktivt. Man uppdaterar dock den övergripande iterations-planen med releaser och datum allteftersom erfarenheten växer. Hittills ej genomförda user stories går igenom, delas upp eller slås samman om erfarenheten visar att det är lämpligt.

Att planera med ”Planning Game” och utföra med korta cykler ger snabbt en konkret känsla för effekterna av planering, estimering och prioritering. Den erfarenheten underlättar väsentligt när man arbetar med stödverktyg för projektplanering och kravhantering. Verktøyen kan underlätta hantering av stora volymer krav och aktiviteter för den erfarna. Men har man någon gång inte planerat för hand kan man lätt missa grovt. Öresundsbron hade stora svårigheter tills man tillsatte en planeringschef som först gjorde en ny översiktsplan med penna och papper. Datorbaserade verktyg hanterade sedan de främst för att hantera stora informationsvolymerna i del-aktiviteter som följde av översiktsplanen.



Extreme Hour är en simulering av ett XP-projekt med betoning på kra-och-planeringsprocessen, The Planning Game. Under XP2000 genomfördes detta som en tävling, Iron Geek med två lag som körde samma simulering.

Tidiga, täta produktreleaser – ”Small Releases”

Kan påverka hela teamet. En individ kan dock leverera ofta.

"Kunden på plats" får en första version redan så snart det finns någon programkod alls som kör felfritt. Redan på "Hello World!"-stadiet! Man skickar därefter, snarast möjligt, ut till "slutkund" ett enkelt, meningsfullt, men robust, system och bygger ut funktionaliteten efter hand. Man släpper nya, skarpa releaser ofta under projektets gång. Med skarp release avses driftsättning eller distribution till kund. Är det en serverapplikation läggs den ut på en server i målmiljön. Skall slutprodukten bli en CD så skapas en fullständig installerbar CD vid varje release. Enligt Becks erfarenhet är releaseintervall på två veckor effektivast. Givetvis medför detta ofta även behov av automatisering när det gäller bygge av kompletta installationspaket.

Enklast möjliga lösning – ”Simple Design”

Kan praktiseras individuellt.

Enklast möjliga lösning som uppfyller samtliga krav. Fokus på att all kod skall göra största möjliga affärsnytta för kunden.

Man anpassar och utvecklar strukturen efter behov genom refactoring.

Kan även göras som "Implementera efter behov". En tänkbar svaghet med "Simple Design" är annars att vissa designidéer inte fångas upp. Genom att uppmuntra teamet att föra anteckningar om idéer i fri form separat ökar chansen att designen hålls ren.

Skriv testkod först av allt– Testa ständigt - ”TestFirst”

Kan praktiseras individuellt

Utvecklarna skriver först kod, ett eller flera testfall, för att testa om kraven uppfylls. Först därefter skriver man koden som skall uppfylla kraven. Man anses vara färdig med koden när alla testfall kör och man inte kan komma på fler testfall för denna "user story" eller "engineering task". Funktionella tester skrivs på grundval av "user stories". Modultester skrivs på grundval "user stories", "engineering tasks" och tekniska aspekter.

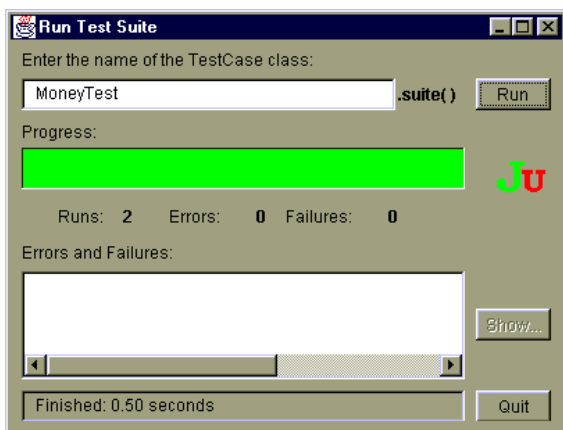
Man vinner även förståelse för konstruktion av funktionen genom att först överväga vad som kan gå fel. När man sedan skriver koden för funktionen skall man lägga till tester om man ser nya tänkbara fel. Man tar sällan bort tester, man uppdaterar dem ibland, men lägger oftast till en ny test.

Erfarenhet styr efterhand vilka typer av tester som teamet går djupast med. Visar det sig att kompilatorer och andra hjälpmedel kan eliminera vissa feltyper effektivare än handskrivna test, väljer man bort jobbet med att skriva nya tester av den typen. Då hamnar den felförebyggande lösningen kanske i kodningsstandarden i stället.

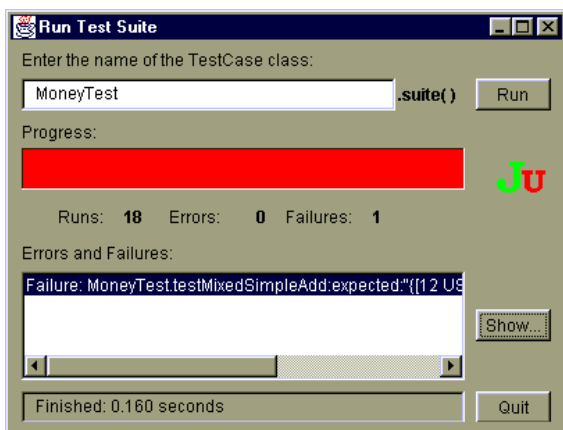
Genom att först skapa testfall bearbetar man problemet innan man implementerar lösningen. Tecknare använder ofta knepet att se på skuggorna runt ett föremål för att hitta rätta konturen. Samma mentala grepp kan anas i TestFirst. Utvecklarna bearbetar problemet genom att skriva tester som visar om problemet löses eller ej. När det är dags att skriva kod för själva lösningen har man redan genomlevt problemet. Man lägger till och uppdaterar tester vid behov också medan man skriver på lösningen. Enkel "tryck på knappen"-test sker kontinuerligt av en växande mängd testfall.

Alla tester körs automatiskt, så ofta som möjligt. Ett enkelt men effektivt ramverk, Junit, skapades av Kent Beck tillsammans med Erich Gamma. Gamma, mest känd för första boken om patterns, finns numera på IBM-bolaget OTI. Junit utvecklas i Java som open source (www.junit.org) och nya versioner överförs efterhand till en mängd olika programspråk ex CppUnit för C++, och VbUnit för Visual Basic osv för språk som Delphi, Smalltalk, Perl, Python, Powerbuilder, Eiffel. Dessa kallas samlat xUnit. Testfallen skrivs i samma programspråk som den testade koden.

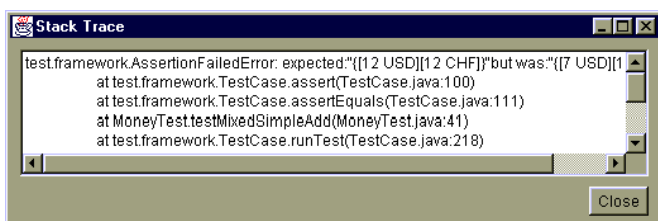
Junit-ramverket läggs in som en liten källkodskomponent i applikationen. Utvecklare dockar in sina testfall i Junit. Varje utvecklare kan sedan köra sedan hela teamets alla testfall med ett enkelt klick på en "Run"-knapp i ett särskilt fristående fönster. Testerna räknas upp, antal runtime-fel (ex felaktigt filnamn, utan felhantering) och negativa testutfall räknas upp. Om inga fel uppstod lyser fönstret grönt. Om ett eller flera fel uppstod lyser fönstret rött. Då finns felen loggade i ett loggfönster. I Junit, dock inte i alla xUnit för andra språk, kan man för varje fel spåra anropskedjan och se vilken eller vilka anropsnivåer som bidrog till att testen inte lyckades. Ett slags integrering av enkel testadministration och mini-debugger.



En lyckad körning av testfallen. Allt är grönt! Det är avsiktligt frestande enkelt för utvecklarna att trycka på "Run"-knappen ofta. Effekten blir små mikro-inkrement med en regressionstest för varje litet steg.



Nu har vi byggt upp fler tester och hittar en test som inte går igenom efter en ändring av koden. Klickar man på "Show" visas anropskedjan för testfallet som inte gick igenom. Rött signalerar test som inte gick OK.



Anropskedjan kan visas för varje testfall som inte går igenom. I vissa utvecklingsmiljöer kan man dessutom klicka på ett anrop i kedjan och komma direkt till rätt ställe i källkoden.

Test Infected: Utvecklare lär sig älska att skriva tester!

Erich Gamma och Kent Beck skriver i artikeln "TestInfected: Programmers Love Writing Tests": "Varje utvecklare vet att de bör skriva tester för programkod. Få gör det. Det vanligaste svaret på frågan 'Varför inte?' är 'Jag hinner inte just nu.' Detta blir snabbt en ond cirkel där man testat allt mindre då tidspressen ökar. Men ju mindre man testat, desto instabilare blir koden och därmed sjunker produktiviteten. Ju mer osäker och osäkrare man känner sig, ju mer tidspress upplever man. Utvecklare bränns lätt ut av denna onda cirkel".

Ständig förbättring av koden – "Refactoring".

Kan praktiseras individuellt.

Refactoring innebär att omarbete programkod med bibehållet yttre observerbart beteende. En grundbult för att detta skall vara rimligt är effektiv versionshantering av programkoden med täta incheckningar. Refactoring förfinar designen inkrementellt. Ställer höga krav på konfigurationshantering.

I praktiken kan refactoring utföras som allt från "sök och byt" i editor, sekvenser av scriptprogram i awk, perl el dyl som ändrar om kod, till avancerade super-editorer, refactoring browsers som bygger på avancerad kompilator teknik och som förstår det aktuella programspråket. Refactoring browsers kan automatiskt utföra syntaktiskt korrekta omstrukturering av programkod och data. Liknande teknik har ofta tillämpats i "software renovation"-projekt, på senare tid framförallt i jakten på 2000-buggar. Ericsson använde kring 1998 forskningsresultat i form av verktyg från Universitetet i Amsterdam för att modernisera och snygga upp Plex-kod för nya AXE-generationer.

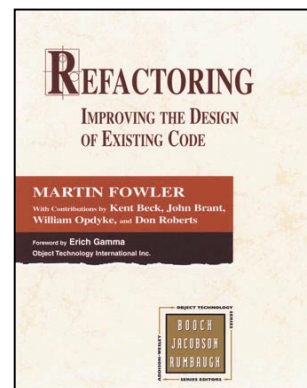
Några XP-värderingar om programkod och struktur:

Ett och endast ett ställe – Kod skall ej vara dubblerad.

Håll det enkelt – Enklast design vinner.

Ändamålsenlig men löst kopplad kod. (Stödjer TestFirst)

Refactoring är aktiviteten då man "snyggar upp" och lever upp till dessa värderingar. Martin Fowlers bok är idag referensverket [ISBN 0-201-48567-2](https://www.amazon.se/dp/0201485672)



All kod skrivs i par – "Pair programming"

Kan praktiseras individuellt.

XP-utvecklare skriver all kod som ingår i slutresultatet i par. Märk väl, två utvecklare på ett tangentbord! Utvecklare bör ha egen arbetsplats för personliga och explorativa moment. Men all "skarp" kod skrivs i par.

Många har sjungit parprogrammeringens lov. Senast har prof Laurie Williams visat, i kontrollerade experiment, att programkod utvecklad i par utvecklas på nästan halva kalendertiden, ger vida bättre kvalitet och kostar ca 15% mer mantimmar.

Men man sitter inte fast med en person hela projektet. Du bjuder in någon att parprogrammera för en viss "user story". Fungerar det bra kan man ta en till, osv. Någon annan kan bryta in och ber dig att parprogrammera en annan "user story" så snart ni är klara. Den sociala regeln är att aldrig tacka nej till en inbjudan att parprogrammera. Den som bjuder in har frivilligt accepterat ansvar för en uppgift ni löser gemensamt. Inbjudan och frivilligt accepterat ansvar är kraftfulla sociala stödmekanismer för XP. Artigheten att inte tacka nej till inbjudan är en annan.

Alla äger koden – "Collective ownership"

Kan praktiseras individuellt.

All kod ägs gemensamt. Alla kan ändra överallt. Detta kräver givetvis kraftfull, rigorös versionshantering och konfigurationsstyrning.

Med rätt stöd på plats kan ändringar som slår i flera moduler utföras mycket snabbt. Friheten att göra ändringar i flera moduler stöds dessutom av TestFirst, Refactoring och Coding Standards.

Bygg hela tiden - "Continuous Integration"

Påverkar hela teamet.

"Daily Build is for whimps" brukar Kent Beck säga när han talar om "continuous integration"

XP-team bygger hela sin produkt många gånger om dagen.

Det behövs för att alla testfall skall kunna köras kontinuerligt och att alla skall kunna arbeta parallellt. Behovet av högt automatiserad konfigurationshantering är uppenbart.

Ingen övertid – "40-hour Week"

Kan påverka hela teamet.

Trötta utvecklare gör misstag. XP-team jobbar inte onödig övertid. Vettigare att sköta hälsan, må bra och vara effektiv när man jobbar.

Att föreskriva 40-timmarsvecka är mer kontroversiellt i USA. Men den sociala regeln inom XP-team säger att man inte jobbar övertid under två veckor i rad.

"Kunden på plats" är med i teamet – "On-site Customer"

Påverkar hela teamet

Ett XP-projekt är oerhört beroende av feedback. Därför sitter en kundrepresentant ("Kunden på plats") som är domänexpert med i teamet. Dvs en person som vet så mycket som möjligt om hur produkten skall fungera i sin tänkta tillämpning.

Kunden på plats skriver krav, prioriterar och svarar på frågor från utvecklarna. Kundens representant sitter i teamet hela tiden men kan jobba med sina övriga arbetsuppgifter då teamet inte behöver input eller feedback.

Man eliminerar tidsödande och kostsamma arbetsdokument mellan kund och team som ej har något värde när produkten är klar. "Kunden på plats" får dock ett mycket större ansvar att styra rätt.

Kodningsregler för frihet – "Coding Standards"

Kan påverka hela teamet.

För att alla skall kunna hitta rätt i den gemensamma koden krävs en ordentlig kodningsstandard. I parprogrammering turas man om att koda. Då gäller det att ha klara regler och ett slags enkelhetens estetik som ger rak och lättförståelig kod där man inom paret kan avlösa varandra vid tangentbordet "mitt i en mening". Regelkontroll kan automatiseras med verktyg sprungna ur kompilator teknik.

Mätningar i XP

I ett XP-projekt mäter man 5 variabler: Kostnad, Tid, Kvalitet (både teknik och kundtillfredsställelse), Omfattning samt Hastighet. Hastighet är en kvot mellan total arbetstid och arbetstid som gjort nytta för produkten. För tankarna till projektstyrningsfilosofin Earned Value. Tidigare inom XP kallades Hastighet (Velocity) för Load Factor. I boken Planning Extreme Programming uppmuntrar Kent Beck och Martin Fowler XP-team att använda ytterligare mätetal som känns meningsfulla. Man förordar att dessa mätetal presenteras som grafer som finns uppsatta där hela teamet kan ta del av dem. Exempel på dessa andra mätetal get Beck som t ex längd av parprogrammerings-pass samt total tid i parprogrammering. Då parprogrammering är den tid som tillbringas för att skriva produktionskod ger Beck exempel på hur man kan få insikt om att man jobbar för långa pass eller planerar för lite i förhållande till förra iterationen genom att titta på detta mätetal. Alla mätetal bör relateras till tidigare iterationer.

Några andra lättviktsprocesser

ASD - Adaptive Software Development (Jim Highsmith)

CRYSTAL LIGHT - Alistair Cockburns metodmatris där man själv väljer ut en metodvariant som passar projektet.

DSDM - Påminner i sociala delen mycket om XP. Saknar dock XPs attraktiva tekniska inslag.
www.dsdm.org

FDD - Feature Driven Design (Peter Coad)

OSS - Open Source Software

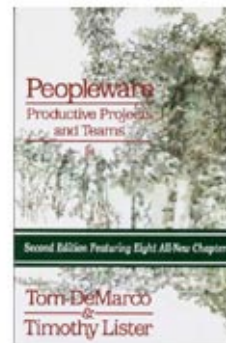
SCRUM - Varje iteration är en tjurrusning med kaos i hälarna. Mellan iterationerna återfinner man en viss ordning. www.controlchaos.com

Är XP rätt väg?

Tom DeMarco - en av författarna till den banbrytande boken "Peopleware"

Jag tycker att XP är det mest betydelsefulla som händer inom programvaruområdet just nu. Jag förutser att Kent Beck kan få större betydelse i detta skede än vad Watts Humphrey och CMM fick i det skede vi just lämnat.

Jag oroar mig dock för arbetsmiljön i XP. Jag önskar att jag kunde tro att folk som arbetar som Chryslers C3-projekt i ett enda stort rum, också har enskilda rum där man kan få vara ifred. Jag önskar jag trodde det, men det gör jag inte.



Alistair Cockburn konstaterar att det krävs tre förutsättningar för att XP skall bli framgångsrikt: låg kostnad att ändra i gränssnitt, tät kommunikation samt automatiserad regression-testning.

Beck refererar till "Crossing the chasm" och säger att det stora steget för XP blir att bli accepterat av den stora massan företag. Då måste XP ses som ett rimligt riskalternativ och behöver en etablerad image.

Varför kallas det XP?

På frågan om hur han valde namnet extreme programming svarade Beck skämtsamt att han strävade efter ett namn som tunga metodgurus som Grady Booch aldrig skulle kunna ta i sin mun.

Dock har Rational, där Booch är en frontfigur, annonserat att man kompletterar Rational Unified Process, RUP för att kunna innefatta även XP. (Rational strävar efter att täcka in de flesta moderna metoder med RUP, t ex DSDM). I en kommande tredje edition av Boochs klassiska "Object Oriented Analysis and Design with Applications." är XP-profilen Robert C Martin medförfattare tillsammans med Grady Booch och James Newkirk. I ett kapitel beskriver Martin en lättviktsprocess, identisk med XP, i RUP-terminologi. Martin kallar skämtsamt processen dX, vilket skall läsas som XP lätt maskerat genom att vända bokstäverna upp och ner.

Det tyngre perspektivet

Watts Humphrey, CMM:s urfader, gillar XP!

Humphrey uttalar sig oväntat positivt om XP i ett debattinlägg för IEEE. Han påpekar visserligen att Kent Beck inte stödjer sig på insamlade data i sin ursprungliga XP-bok, men säger samtidigt att Humphreys egen forskning kring team-processer, TSP, har samlat data som ger starkt stöd för XP. Invändningarna mot XP gäller främst att XP i sin ursprungs form inte skalar till team större än 10-15 personer, samt att XP inte är beskrivet i detalj. Vilket är helt korrekt.

XP kan leda direkt till CMM nivå 4

Humphrey verkar hos Software Engineering Institute, SEI som "äger" och utvecklar CMM-standarden. En av hans kollegor Michel Paulk gör gällande, i ett annat debattinlägg för IEEE, att ett XP-team direkt skulle kunna certifieras på CMM nivå 4 om man kompletterar XP med vissa mätningar och införliva resultaten i feedback-styrningen som redan finns i XP.

XP möter CMM i Indien

Vi har inom svenska SPIN initierat en aktivitet via SPIN Bangalore i Indien för att få över några XP-profiler att studera indiska team på CMM nivå 5. Det finns mycket som tyder på det naturliga indiska sättet att arbeta i team väl överensstämmer med XPs intentioner. Swami Namph på Motorola i Bangalore betonade indiska teamens naturliga rollintegration. (Se vidare VI nyhetsbrev Systemkonstruktion i programvara nr 2 år 2000)

Om det är möjligt att XP tillför liknande egenskaper till team utanför Indien kan vi kanske börja närma oss Indien när det gäller CMM-mognad.

Kent Beck säger själv att när han pratar med människorna bakom CMM så är man rörande överens. Men inom många företag i västvärlden tillämpas CMM på ett mindre gynnsamt sätt, i värsta fall som skådebröd.

Vilka verktyg behövs för XP?

Ritverktyg med begränsad kodgenerering ratas till förmån för generella, bättre ritverktyg som Visio. Två gånger under konferensen XP2000 hölls handuppräckning om vilka som föredrog ett generellt ritverktyg för UML-diagram som Visio framför Rational Rose och liknande. Visio vann stort båda gångerna. En av de som uppmanade till handuppräckning, och som själv röstade för Visio, hör till de ursprungliga utvecklarna av Rational Rose...

Alltför många användare sägs undvika kodgenerering, och använder CASE-verktyg enbart som extremt dyra ritlådor. Det generella ritprogrammet Visio har funnit sedan början av 90-talet. Inriktar sig främst på schematisk linjegrafik som organisationschema, flödesdiagram, eller som en parentes UML-diagram. Då de generella egenskaperna för strukturerad grafik i Visio drivs av en långt större användarbas, är ritegenskaperna överlägsna samtliga CASE-verktyg på marknaden. Microsoft köpte Visio 1999.

Inom XP uppmuntras däremot användning av konfigurationsverktyg, versionshantering och testautomation. Avancerade verktyg för semantiskt korrekt kodförbättring genom refactoring är också favoriter. Dock finns idag kvalificerade refactoring-verktyg enbart i Smalltalk-miljö.

XP drar fulla hus

XP2000 - den första internationella konferensen

Delegater från HP, Ericsson, Nokia, Nortel, IBM, Motorola, Daimler Chrysler, Siemens samt indirekt Microsoft samlades i slutet på juni i år på Sardinien för den första internationella konferensen om XP: eXtreme Programming and Flexible Processes in Software Engineering - XP2000.

Den som följt mer avancerad litteratur och debatt inom XP, software engineering och computer science fick sitt lystmäte av kändisar: Kent Beck, Erich Gamma; Martin Fowler, Ralph Johnson, Robert Martin, Don Wells, Ron Jeffreis, Alistair Cockburn. Mycket av XP kommer från den design patterns community som växte upp sedan boken Design Patterns av Erich Gamma mfl kom ut 1995.

Intressant var bl a erfarenheter av att XP accepteras lättare än andra processer av team med forskarbakgrund.

Dom som fruktade ett fanatikermöte om XP blev lättade. De flesta som var där hade infört XP helt eller delvis på ett pragmatiskt sätt. Vilket är helt i XPs anda att odla djupare förståelse för teknik och process. Dave Thomas från IBM, Martin Fowler, m fl profiler uttryckte sin glädje att finna att XP-folket verkligen lever som de lär, flexibelt och med respekt för andras idéer.

Intresset för XP är extremt stort. När universitetet i Calgary planerade för konferensen på Sardinien räknade man med ca 60 internationella gäster. Det kom 161 delegater från hela världen, varav 15 från Sverige. XP2000 gav mersmak och nästa års konferens presenteras redan på www.xp2001.org

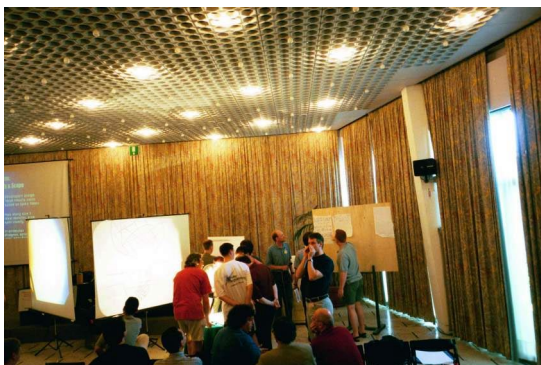


Bild ovan: XP2000-konferensen innebar mycket utspel och handfasta övningar. Här är ännu en bild från Iron Geek som hör till de mer omtalade inslagen på XP2000. Hela övningen tog 70 minuter och refererades av två kommentatorer. Formen är en kombination av Extreme Hour och det japanska originalet till svenska tv-serien "Kockduellen" som i USA vunnit stor popularitet med titeln "Iron Chef" (Kent Becks lag förlorade till allmän munterhet.)

Bild till höger: Robert Martin gav flera kärnfulla föreläsningar under XP2000. Hans företag ObjectMentor satsar hårt på XP



Kent Beck mfl frontfigurer inom XP glömmor, ivrigt diskuterande, bort att bada i Sardiniens salta vatten på XP2000-konferensens sista kväll. Min familj var inte den enda som skakade på huvudet och tog en bild från stranden



Kent Beck på turne i Sverige

Kent Beck besökte Sverige 25-26 september. I Lund drog han fullt hus LTH under LUCAS:s timslånga XP-seminarium. I Linköping överträffade han alla förväntningar när han lockade 100 deltagare till IDA:s heldagskonferens SOFT-22.



Görel Hedin, Per Runeson, Kent Beck, Boris Magnusson, Karl-Erik Årzén vid ett XP-seminarium vid LUCAS i Lund. Lunds Tekniska Högskola uppmärksammade tidigt XP.



Kent Beck signerar sin XP-bok åt många av de hundra deltagarna på XP-seminariet SOFT-22 i Linköping

Erfarenheter från företag

Motorolas Ron Crocker publicerade på OOPSLA2000 ett papper om erfarenheter av XP som metamodel för att samordna utvecklingsteam på skilda platser i världen. I Motorolas modell motsvaras individen i ett vanligt XP-team av ett platsteam. Varje platsteam kan välja sina egna interna arbetsmetoder. Samordningen sker dock enligt XP-principer. Intressant var också att den legendariska Motorola-enheten i Bangalore, första civila verksamhet i världen på CMM nivå 5 1993, deltog i detta stora meta-XP projekt. Crocker indikerade att detta XP-projekt över tiden engagerade mellan 100 och 500 personer. Dock fungerade inte alla delar av XP i ett så stort projekt.

Henrik Warne från Ericsson var en av delegaterna på XP2000 i juni. Hur långt har man anammat XP nu i höst?

Henrik Warne, Ericsson Radio Systems, Kista:

Vi har provat delar av XP efter hand. Vi jobbar med programvara för nya generationer av mobiltelefonens basstationer. Testförfarandet med kontinuerlig testning har fått omedelbart genomslag hos oss. Vi utvecklar i Java och kan direkt använda Kent Becks testramverk Junit "på originalspråket". Våra utvecklare regressionstestar hela tiden med "ett tryck på knappen" som kör växande gemensamma testserier där alla kan stoppa in nya tester.

Parprogrammering har varit mycket givande för de flesta som velat pröva. Sitter man vid tangentbordet, så kan man ställa frågor rätt ut i luften och kollegan är direkt med utan att man behöver förklara. Management har varit positiv till parjobb då man ser det som kompetensspridning



Chryslers C3-projekt satt i ett enda stort rum. Foto: Ron Jeffreis, www.xprogramming.com

samt för nya utvecklare ett slags positiv lärlingsprocess. Ökad mantid har inte oroat i den ekvationen. Utvecklare med inbiten vana att jobba själv kan däremot vara negativa innan de provat.

Vi tittar just nu på att tillämpa snabb kontinuerlig kravprioritering enligt XP i vissa projekt.

Det vi inte tror på i XP är Becks "Open Workspace" i bemärkelsen att ständigt ha teamet i ett enda stort rum fullt med datorer som Chryslers XP-projekt C3. *Se även inlägg av Tom DeMarco, förf anm.*

Slutsatser

Boehms "Cost of Change" gäller men effekten minskas av XP

Barry Boehms klassiska modell säger att det alltid är billigast att hitta fel så tidigt som möjligt. Kurvan över kostnad för att avhjälpa ett fel steg exponentiellt ju längre produkten kom i processen. Nackdelen är om man inte kan föreställa sig hur det fungerar och designfasen drar ut på tiden.

Beck hävdar i sin XP-bok att XP plattar till den exponentiella kurvan. Cockburn hävdar däremot att XP ger effekten att man alltid befinner sig i början av exponential-kurvan...

Tekniska och ekonomiska faktorer bäddar för XP

XP blir möjligt bl a tack vare kraftfulla billiga datorer och avancerade utvecklingsverktyg som blivit såljs i massupplaga. Då kan en standard PC eller laptop innehålla verktyg som tidigare krävde, med dåtidens mått, kraftfulla dedicerade Unix-servers.

Utveckling inom software engineering med ex design patterns, kompilatorteknik och nätverksorientering verkar för att det skall vara tekniskt möjligt att kunna hålla fokus i koncentrerade releaser av programkod.

XP förbättrar inte hela organisationen

XP ger, rätt tillämpat, ett team gynnsamt beteende för programvaruutveckling. Det gör dock inget för resten av organisationen. Är XP den enda programvaruprocessen, krävs hela tiden stor förståelse på individnivå från resten av organisationen. Det är en del av orsaken till att XP kan uppfattas som cowboyverksamhet. Chryslers C3-projekt som var ett tidigt större XP-projekt för löneadministration blev till slut omöjligt att fortsätta av skäl som inte teamet kunde påverka. Resten av organisationen arbetade på en helt annan mognadsnivå när det gällde programvaruutveckling. Det management som startat projektet hade gått till andra uppgifter. Interna konflikter inom Chrysler ledde till att C3-projektet lades ner. XP blev en tid ett internt slagträ inom Chrysler.

Vi tror att CMM eller motsvarande mognadsprocess bör tillämpas på hela organisationen med fokus på management medan XP tillämpas på utvecklingsteam.

XP kan få den stora personalgruppen utvecklare att fungera bra i en organisation som på managementnivå mognas med CMM. CMM tar ett helhetsperspektiv som tilltalar management, medan XPs resultatorientering tilltalar utvecklarna. När XP och CMM samverkar på rätt sätt kan vi få verkligt snabb mognad av stora programvaruutvecklande organisationer.

XP lättare för det mindre företaget?

Av många skäl, inte minst politiska, kan det vara lättare för team i mindre företaget att anamma XP på en gång i ett eller flera projekt. I större företag kan man försöka introducera utvalda delar stegvis. Detta ökar dock risken för misslyckande då flertalet av XPs olika metoder har misslyckats på egen hand. Det är just interaktionen mellan de olika fördelarna hos olika metoder som kompenserar för nackdelarna hos enskilda metoder. I de flesta fall elimineras helt nackdelarna med andra stödjande metoder, i enstaka fall överträffar fördelarna för helheten metodens nackdelarna. Därför kan ett

projekt som tillämpar XP fullt ut åtnjuter fördelar som man går miste om när man tillämpar enstaka metoder.

XP driver teknisk och social kompetens

Många har upplevt, eller hört talas om skräckprojekt där varje utvecklare har begränsad kompetens och bara kan "var sitt enda kapitel i grundboken", särskilt i de senaste årens webb-frenesi. samt i många kunds specifika projekt bemannade med resurskonsulter. Beck har mycket erfarenhet av kunds specifik utveckling med blandade team. Därför har XP designats för att driva kompetensspridning och kommunikation. Medlemmar i ett XP-team delar naturligt kunskap genom parprogrammering och kollektivt ägande.

Den utvecklare som trivs i ett XP-team får därmed på köpet en bred personlig kompetensutveckling. Många upptäcker att även de mest erfarna får nya insikter av att arbeta med mindre erfarna. Med parprogrammering blir detta särskilt tydligt.

XP är starkt feedback-orienterat och därigenom i sig en lärande process. Det smittar av sig.

Utvecklarna kräver bättre arbetsformer!

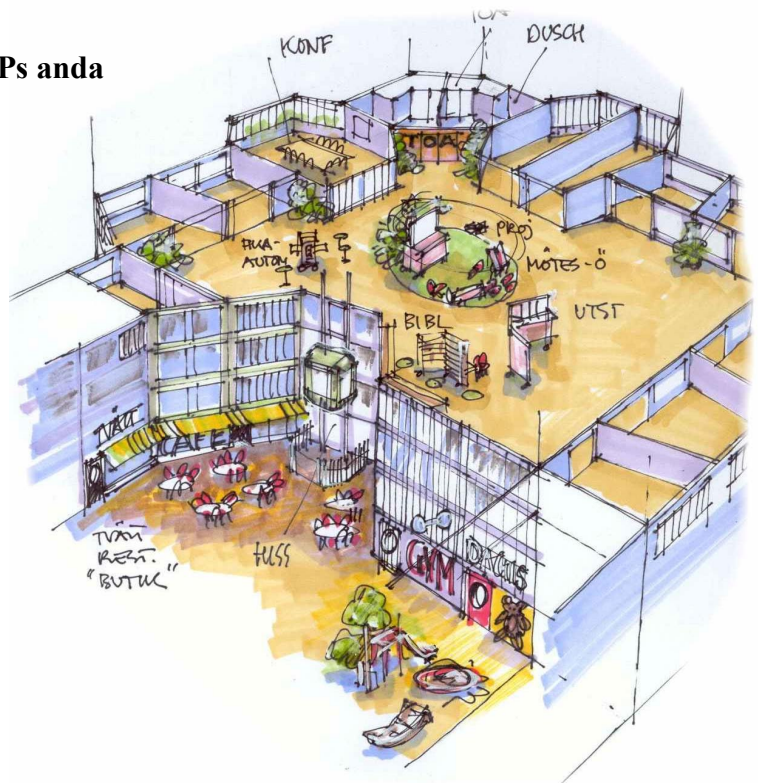
Martin Fowler sa i sitt avslutningstal på konferensen XP2000: "If you can't change your organization – Change your organization. This market is no time to be unhappy with your job!" En direkt uppmaning att byta arbetsgivare om man vantrivs och inte lyckas förbättra sin nuvarande organisation.

Kent Beck tar avstånd från managementkonsulter och vill själv gärna ses som en duktig utvecklare bland andra. Men XP handlar egentligen om empowerment, de senaste årens trend bland managementkonsulter. (Empowerment i någon form blir trendmässigt ofta populärt när efterfrågan på kompetent personal vida överstiger tillgången.)

Runt om i världen skapas platser i XPs anda

Kent Beck planerar ett utvecklingscenter med inriktning på forskning kollaborativa utvecklingsprocesser i Oregon, USA där företag kan sätta team att arbeta i XPs anda.

Beck blev under XP2000-konferensen intresserad av ett liknande projekt i Sverige (Product Software Center - Helsingborg) som Erik Lundh är engagerad i sedan mer än två år tillbaka. Ken Auer, en annan av XPs frontfigurer, håller i USA på att bygga upp en "XP Software Studio" med positiva drag av konstskola.



Ett hus i XPs anda kan komma att byggas i Helsingborg.
Vidare info: psc-hbg@compelcon.se

Referenser

Webb

<http://www.extremeprogramming.org>

www.xprogramming.com

www.c2.com/cgi/wiki?ExtremeProgrammingRoadmap

collaboration.csc.ncsu.edu/laurie

www.pairprogramming.com

www.junit.org

www.objectmentor.com

www.xp2001.org

Don Wells XP-site. Rekommenderas!

Utmärkt om XP, samlar varianter av xUnit

Här föddes XP! (Eller formades i vart fall mycket!)

Prof Laurie Williams har mätt nedlagd tid vid parprogrammering.

Webbsite med fokus på parprogrammering

Testramverket Junit

Objectmentor lär ut XP med Kent Beck

Info om nästa års XP-konferens på Sardinien

Böcker

Mer information om varje bok: Klicka på ISBN-numret!

ISBN	Titel	Författare	Förlag
9177385268	I en kaordisk tid - Dee Hock plastkortets skapare och grundare av VISA	Dee Hock	Svenska Förlaget
0201616416	eXtreme Programming eXplained : Embrace Change	Kent Beck	Addison-Wesley
0201708426	Extreme Programming Installed	Ann Anderson, Chet Hendrickson, Ronald E. Jeffries	Addison-Wesley
0201710919	Planning Extreme Programming	Kent Beck, Martin Fowler	Addison-Wesley
0201485672	Refactoring : Improving the Design of Existing Code	Martin Fowler et al	Addison-Wesley
0932633439	Peopleware : Productive Projects and Terms, 2ed	DeMarco Lister	Dorset House Publishing
0201895358	CRC Card Book	Bellin, David Simone, Susan Suchman	Addison-Wesley
0201633612	Design Patterns	Gamma Helm Johnson	Addison-Wesley
0201548097	Introduction to the Personal Software Process	Watts S. Humphrey	Addison-Wesley
0201545977	Managing Technical People	Watts S. Humphrey	Addison-Wesley

Om författaren

Erik Lundh verkar som konsult i det egna bolaget Compelcon AB. Erik är idag främst inriktad mot företag med massprodukter som kräver programvara. Han har, sedan snart 20 år, verkat i egna och andras projekt, i små och stora verksamheter. Alltför många akuta räddningsuppdrag i verksamheter, med tekniska och mänskliga kriser, drev med tiden Erik att kombinera strategisk helhetssyn med teknisk insikt. Med strävan att förstå både ledningens och utvecklarnas situation. Idag delas tiden mellan coaching, verksamhetsutveckling, enstaka styrelseuppdrag, viss teknisk utveckling, familjen samt internationellt nätverksbyggande bl a genom SPIN-SWEDEN. Föreningar som Mensa och Svenska Uppfinnareföreningen ger ytterligare innovativa infallsvinklar.



Erik nås enklast per mail: erik.lundh@compelcon.se eller tel: 042 18 44 84.

Regionalt i Skåne är Erik en av de mer aktiva i nätverket SPIN-SYD (www.spin-syd.org). Bland övriga medlemmar i SPIN-SYD återfinns bl a ABB, Axis, C-Technologies, Ericsson, Sigma, Telelogic, Tribon, Q-Labs. Även där arbetar Erik för att dela erfarenheter både regionalt och internationellt, t ex med Bangalore, Indien. De tre senaste åren har Erik sponsrat SPIN-SYDs årliga konferens tillsammans med Q-Labs. Eriks arbete i SPIN-SYD är ideellt.

Erfarenheterna från SPIN-SYD har Erik, tillsammans med Per Beremark, C-Technologies, förpackat som ett startup-kit för nya regionala SPIN. Erik och Per hoppas att kunna hjälpa igång minst ett regionalt SPIN under 2001. Kontakt: nya-spin@compelcon.se

Erik initierade för drygt två år sedan projekt för att skapa en grupp etablerade industrier med massprodukter vilka kräver användbar och tillförlitlig programvara, exempelvis mobiltelefoner. Ett R&D-center för programvara som attraherar forskare och utvecklare av världsklass att arbeta i små team nära högskolan. Arbetsformer, intressanta projekt och bra familjeliv lockar spetskompetens internationellt. XP är en hörnsten i detta projekt.

SPIN-SYD startar XP-grupp!

Erik kommer under våren 2001 att leda en XP-grupp inom SPIN-SYD. Syftet är att deltagarna skall få förmåga att arbeta enligt XP-principer. Gruppen drivs som ett samarbete mellan SPIN-SYD och LUCAS-institutet vid Lunds Tekniska Högskola. Vikt läggs vid att få verklig insikt om de bakomliggande tankar och värderingar som förmade XP. När man väl förstår XPs tanke kan man själv skraddarsy sin process efter aktuellt projekt. Gruppen kommer därför att ha praktiska XP-övningar för att få erfarenhet, men också känsla, för processen.



Ett mål för gruppen är att kunna visa verkliga XP-färdigheter vid SPIN-SYDs årliga konferens fredagen den 4 maj 2001.

Intresserade av XP-gruppen kan få kontakt via: xp-wg-2001@compelcon.se