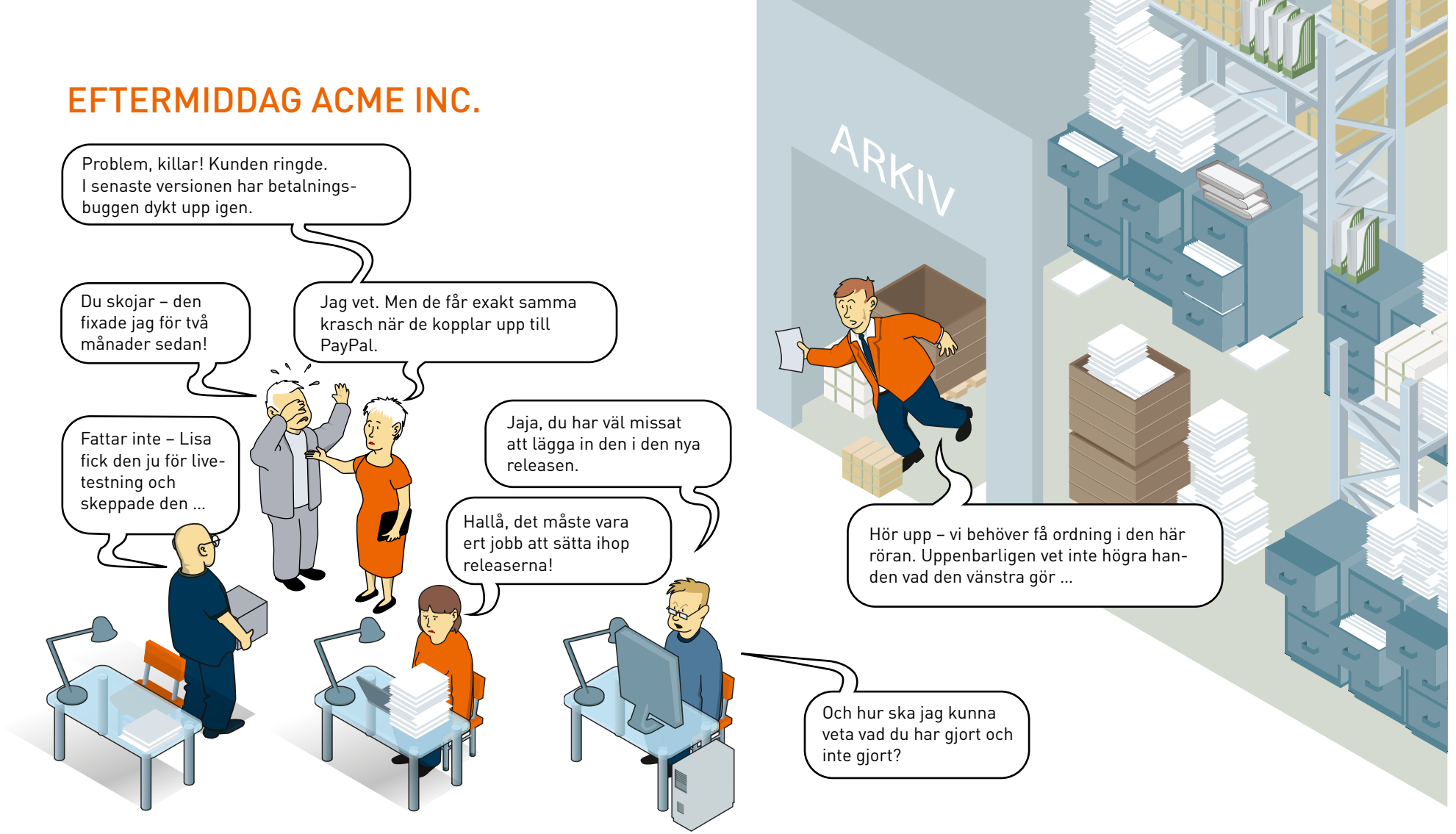
SOFTHOUSE

CONFIGURATION MANAGEMENT

på fem minuter

EFTERMIDDAG ACME INC.



HAR NI ORDNING OCH REDA?

Hur ser det ut när ni utvecklar produkter eller kod på ert företag?

- Händer det att fel och buggar ”uppstår från de döda” (som i exemplet här ovan).
- Har ni utvecklat rutiner för att rätta ett rapporterat fel i flera samtidiga versioner av produkten?
- Är arbetet med att få ut en ny version komplicerat, tidskrävande och ett hinder för nya uppdateringar?
- Saknar ni ett komplett och överskådligt arkiv av tidigare versioner?

Då kanske ni behöver vässa företagets kompetens när det gäller CM, Configuration Management – konsten att hålla reda på versioner, komponenter och kompletteringar. Detta gäller alla organisationer som bedriver något slags avancerad produktutveckling, oavsett om det handlar om mjuk- eller hårdvara.

CONFIGURATION MANAGEMENT 1.0

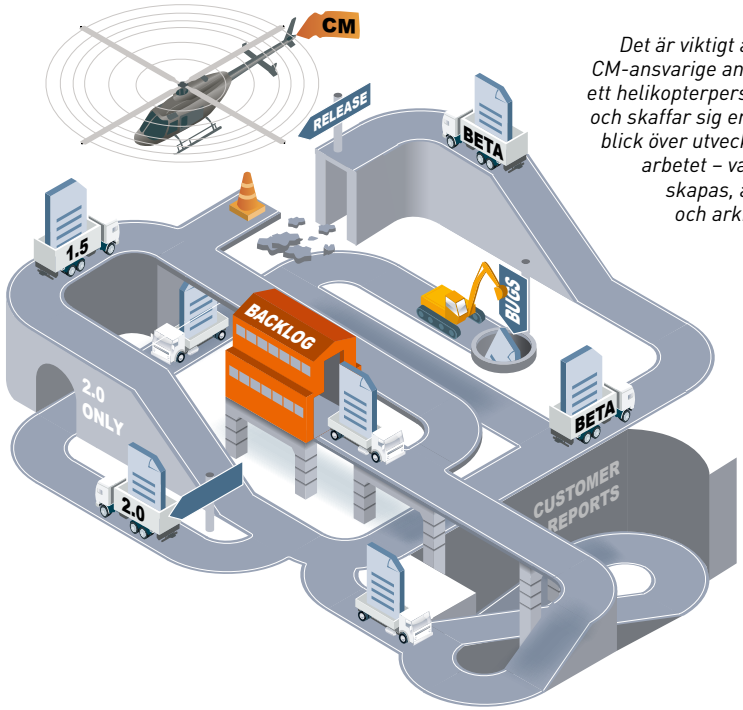
CM går ut på att skapa ordning och spårbarhet där det behövs. De uppenbara fördelarna handlar mycket om att undvika krångel, problem och misstag, till exempel genom att

- skapa tydliga regler för versionsbenämning, dokumentation, lagring av filer etc.
- effektivare kunna spåra vad en leverans innehåller
- enklare kunna se vilka buggar och problem som hör ihop med olika versioner av produkterna.

I stora organisationer som utvecklar omfattande system är CM helt nödvändigt eftersom hundratals programmerare kan vara inkopplade på samma projekt. Här är det lätt att motivera medarbetarna eftersom alla inser att formaliserat CM-arbete är skillnaden mellan ordning och kaos.

I mindre företag kan det däremot finnas lite större motstånd mot CM-arbete. Det kan helt enkelt kännas onödigt formellt i en organisation där alla känner varandra och har överblick över projekten. För att motivera medarbetarna gäller det att framhålla de fördelar som CM-arbetet leder till – och sluta tala om det som ett nödvändigt ont.

Alla tillämpar vi CM på något sätt idag. I sin allra enklaste form kan det se ut som i följande exempel:



Det är viktigt att den CM-ansvarige anlägger ett helikopterperspektiv och skaffar sig en överblick över utvecklingsarbetet – vad som skapas, ändras och arkiveras.

Exempel 1: Det gemensamma textdokumentet

Hans och Tage skriver en bok tillsammans där de jobbar i ett ordbehandlingsprogram. Versionerna ligger i en mapp i molnet. Varje gång någon av dem jobbar med filen gör han en ny kopia av ordbehandlingsfilen, lägger till dagens datum samt en signatur. Filernas namn blir

2013-07-01_boken_hans.doc
2013-07-02_boken_tage.doc
2013-07-03_boken_hans.doc
...

Denna manuella form av versionshantering är lätt att förstå men den har brister. Vad händer t.ex. när de två författarna sparar en ny version på var sitt håll?

Exempel 2: Den kommersiella mjukvaran

Företaget Microapples viktigaste produkt är ett layoutprogram. Så här ser release-loggen ut:

0.99	*Beta (testversion)*
1.0	*Första versionen*
1.0.1	*Buggfix*
1.1	*Uppdatering (buggfixar & funktioner)*
1.5	*Uppgradering*
2.0	*Omfattande uppgradering*
2.0.1	*Buggfix*
osv.	

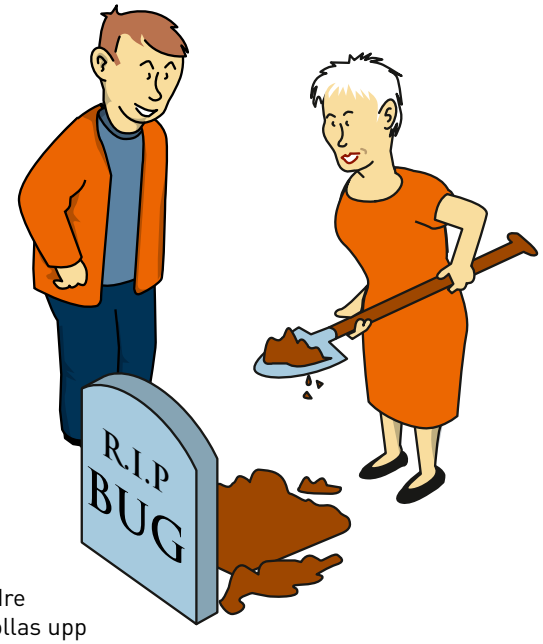
Detta är klassisk versionsidentifiering för mjukvara. Det är viktigt att se till att det finns ett så tydligt regelverk som möjligt för numreringen.

FÖRDELAR MED CM

CM är så mycket mer än ”ett system för att inte röra till det”. Väl fungerande rutiner och verktyg ger en hel rad positiva effekter.

Utvecklingsarbete blir snabbare.
CM höjer tempot i utvecklingsarbetet på många olika sätt. Först och främst möjliggör det att parallella utvecklingsteam kan arbeta i samma projekt på ett kontrollerat sätt (se nedan). Dessutom förenklar det systemanalyser, felsökning, testning, mätning samt uppföljning.

”Kontrollerat kaos” möjliggörs.
Ju bättre organisationen blir på CM, desto mer avancerade projektmodeller kan den tillämpa, med t.ex. parallella, självständiga utvecklingsteam, snabba iterationer och tät återkoppling från marknaden. CM säkerställer t.ex. att buggar inte ”återuppstår från de döda” och att kända problem åtgärdas vid rätt tidpunkt i produktionsflödet.



Organisationen får ett helhetsgrepp.
Alla intressenter får bättre överblick över hur mjukvaran utvecklas, vilka varianter som skapas samt vad som har ändrats mellan olika versioner. Sist men inte minst är det lättare att hitta bland äldre versioner när något behöver kollas upp eller återskapas.

Produktstrukturering ökar flexibiliteten.
CM gör det enklare att dela upp och hantera mjukvaran i moduler – något som ger många fördelar. Bl.a. kan man snabbare skapa nya produktvarianter när man utgår från befintliga moduler. Man får även en effektivare mjukvaruutveckling när arbetet delas upp på de olika modulerna.

CM-ANSVARIG – EN ROLL I FÖRÄNDRING

Beroende på organisationens storlek kan rollen som CM-ansvarig – även benämnt Configuration Manager – antingen utgöra en deltidssyssla för en person som även gör något annat eller en heltidssyssla. Oavsett vilket måste den CM-ansvarige besitta en stor förståelse för hela utvecklingsprocessen och ligga ett steg före i arbetet. Dagens mångfald av utvecklingsmodeller och arbetssätt gör att den CM-ansvarige måste vara anpassningsbar och förstå hur olika team jobbar.

• När **vattenfallsorienterade utvecklingsmodeller** används är CM-arbetet som mest intensivt tidigt och sent i projektflödet: först intensiv planering i de tidiga projektfaserna och sedan uppföljning, indrivning av leveranser samt administration av ändringshanteringen i de senare faserna.

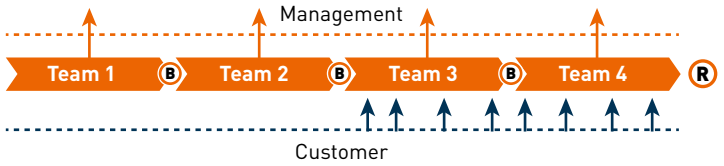
• **Iterativa och inkrementella utvecklingsmodeller** gör att komplexiteten i CM-arbetet ökar dramatiskt; det är som om flera projekt löper samtidigt där man jobbar med samma produkt. Det blir fler artefakter att hålla ordning på och dessa får dessutom mer komplicerade livscykler – något som ställer större krav på både verktyg och processer.

• **Agila modeller** ökar parallelliteten i arbetet ännu mer och låter dessutom teamen själv bestämma hur de ska jobba. Den CM-ansvarige måste här vara mycket lyhörd, ha stor förståelse för teamens arbete och själv låta sig vägledas av de agila principerna. I många fall blir det naturligt att teamen själva tar hand om det rent operativa CM-arbetet. Den CM-ansvarige får då en expert- och coachroll som fångar upp problem och idéer och avgör vilka som ska implementeras.

På nästa uppslag ska vi se närmare på dessa utvecklingsmodeller.

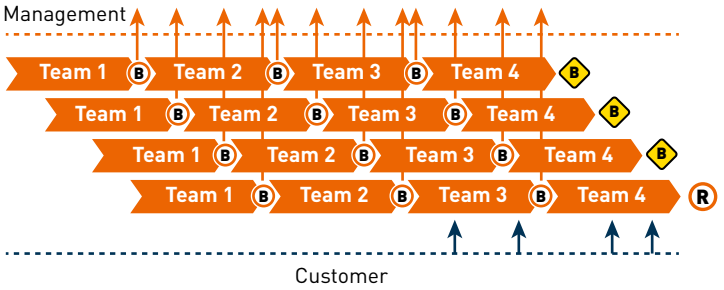
CM I OLIKA MODELLER

Vattenfallsorienterad utvecklingsmodell



Ur ett CM-perspektiv är vattenfallsmodellen enkel: det är ett enda spår att hålla reda på, indelat i arbetsfaser. Baselines, statusrapporter och felrapporter blir därför relativt enkla att hantera. Det som dock kan komplicera bilden är att man ofta har många change requests att hantera.

Iterativ utvecklingsmodell



I CM-sammanhang kan en iterativ utvecklingsmodell beskrivas som flera samtidiga vattenfallsprojekt. Eftersom iterationerna är delvis parallella blir situationen naturligtvis mer komplex.

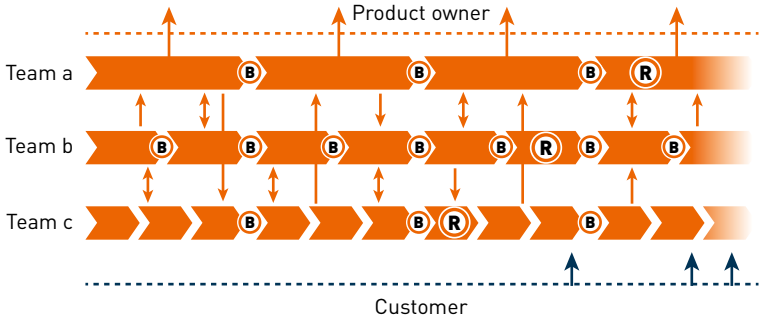
Här måste man
- hålla reda på vad som ingår i vilken iteration
- vara tydlig i sin rapportering
- ha en väloljad hantering av felrapporter (vilka fel ska rättas i vilken iteration?).

När det gäller change requests kan situationen faktiskt bli något enklare. Man har ju möjlighet att lägga in en ändring i en iteration som ännu inte är påbörjad och behöver då inte ändra något som är under arbete.

Förklaring:

B Baseline Statusrapport R Release Change request B Beta release

Agil utvecklingsmodell



När agila utvecklingsmodeller används är komplexiteten ännu större eftersom man fokuserar mer på ansvar i varje team, täta leveranser och en ständigt pågående testverksamhet. Individuella iterationer för varje team pågår parallellt och inte nödvändigtvis med samma periodicitet.

Kraven på statusinformation är höga både ”uppåt” i organisationen och mellan teamen. Varje iteration eller uppgift blir här en baseline. Det är viktigt att administrativa rutiner är lättviktiga och enkla att följa eftersom det är utvecklingsteamen som kommer att utföra det mesta av det arbetet.

Ändrade krav eller change requests är däremot ett naturligt inslag i en agil organisation och de agila modellerna har fungerande rutiner för hur man hanterar dem. Däremot är det fortfarande mycket viktigt med spårbarhet så att man vet vad som implementerats i vilken ändring.

Hur självbestämmande teamen ska vara är en avvägningsfråga. En vanlig lösning är att de håller i det operativa CM-arbetet och att den CM-ansvarige intar en expertroll utanför teamen med ansvar för problemlösning och förbättringsarbete. Ytterligare uppgifter för den CM-ansvarige är att genomföra utbildningar samt att hjälpa teamen utveckla sin förståelse för betydelsen av CM – inte minst konsekvenserna av slarv och genvägar.

VERKTYG FÖR CM

VERSIONSHANTERING

Verktyg för versionshantering fungerar enligt olika principer.

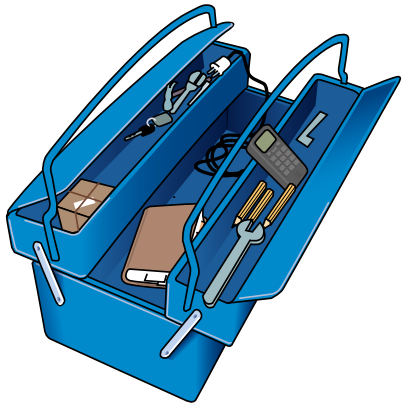
- **Enkla verktyg** skapar en versionshistorik vilket gör det möjligt att återskapa tidigare versioner av filerna. De gör även att det inte går att skriva över ändringar som andra lägger in. Vissa av dem ger dessutom bättre stöd för arbete längs parallella spår och kan hantera större system.

- **Avancerade verktyg** ger ännu bättre stöd för hantering av större system och olika typer av konfigurering. Dessutom får man ofta mer funktionalitet och spårbarhet genom systemets hela livscykel samt spårbarhet även vid byggen och integration av mjukvara.

Det finns idag en uppsjö av verktyg för versionshantering. Små företag behöver sällan en stor kommersiell lösning. Då är det bättre att först fundera över vilka krav man har på sin utvecklingsmiljö och sedan se över marknaden – antingen själv eller med hjälp från någon som har mer erfarenhet.

Den senaste trenden inom versionshantering arbetar med distribuerade arkiv där alla användare har en egen klon av arkivet. Man jobbar då på sin egen dator till dess att man väljer att synkronisera sin version med en server eller en annan användare.

Gratisverktyg: Subversion, Git, Mercurial, Fossil, Veracity
Kommersiella verktyg: Accurev, Perforce och PVCS version manager, Team Foundation Server



ÄRENDEHANTERING

Verktyg för ärendehantering innefattar mycket processkontroll. Här är det alltså viktigt att man väljer ett verktyg som antingen

- implementerar ett arbetssätt som ligger nära det man vill åstadkomma.
- erbjuder fullständig anpassningsbarhet (men då krävs det att man är villig att lägga ner resurser och tid på att anpassa det).

Det är ofta viktigt att verktyget för ärendehantering går att integrera med verktygen för versionshantering, leveranshantering och kravhantering.

Gratisverktyg: BugTracker.NET, Bugzilla, Fossil, Mantis, Veracity, Eventum
Kommersiella verktyg: Tracker, Jira, Serena Issue and Defect Management, IBM Rational Team Concert

INTEGRATION OCH LEVERANS

Detta är ett område som blivit allt viktigare i takt med introduktionen av agila metoder. När utveckling pågår i parallellt arbetande team behöver man ge stöd för att kontinuerligt uppdatera sitt eget spår med det som hänt i andra team och på huvudgrenen. Eftersom man levererar ofta, både internt och externt, kan stöd för en så automatisk leveranshantering ge stora effektiviseringsvinster. Dessa verktyg har starka kopplingar till både versionshantering och ärendehantering och det därför är nödvändigt att dessa kan integreras på ett bra sätt.

Gratisverktyg: Buildbot, Jenkins, Hudson, Travis CI, Testflight
Kommersiella verktyg: CruiseControl, Electric Commander, TeamCity, Bamboo

Slutligen några ord om verktyg för **kravhantering**. Dessa räknas normalt inte till CM-verktygen, men det är ofta viktigt att de går att integrera med verktygen för versionshantering och ärendehantering. Anledningen är att man gärna vill automatisera spårbarheten genom utvecklingen. Genom att integrera med kravhanteringsverktyget kan man göra det möjligt att till exempel automatiskt skapa rapporter – om bl.a. vilka krav som implementeras vid vilken tidpunkt, eller när ett fel som påverkar ett visst krav uppstod och rättades.

CASE: VÄRDET AV EN KONFIGURATION

Följande case är en fiktion, baserad på verkliga situationer som uppstått på företag i Skandinavien.

Bakgrund
På Tanto Telecom utvecklar man ett stort system för fjärrstyrning av robotanläggningar. I anslutning till att releasedatumet närmar sig intensifieras testningen i labbet. Ett stort antal buggar och brister har påträffats och organisationen har under flera dagar haft ett stort antal rättningar att hantera. Nu har dock alla tester passerat utan problem vid teststationerna. Det enda som återstår är att leverera samma konfiguration till arkivet för släppta produkter och se till att dokumentationen är komplett.

Problem
Tanto Telecom har ingen dedikerad CM-organisation, utan har gett ansvaret till utvecklare som också löser andra uppgifter. Nu i slutfasen gör en oerfaren men engagerad CM-ansvarig en sista dubbelkoll i tysthet och blir förfärad: det som håller på att sparas för release stämmer inte med det som godkänts i testerna! Genom att bygga en ny testkonfiguration och köra testerna bekräftar han att den release som är på väg att skeppas fortfarande är full av buggar.

Förklaring
Ledningen på Tanto Telecom drar i nödbromsen och tillsätter en analysgrupp. Efter ett antal timmars intervjuande, analyserande och letande kryper sanningen fram. Problemen har sitt ursprung i det faktum att processen för leveranser till testarna uppfattas som onödigt komplicerad. Detta har gjort att många utvecklare har börjat använda sig av genvägar.

Komplikation
Problemet fördjupas ytterligare när man inser att många utvecklare har skickat felrättningar till test utan att spara versionsmärkt källkod i arkivet. Den fungerande versionen går därför inte att återskapa.

Lösning
Det är lätt att tro att lösningen här ligger i att se till att leveransrutinerna efterföljs och att hindra alla genvägar. Det är emellertid en kortsiktig lösning som kommer att minska förtroendet för CM istället för att stötta verksamheten.

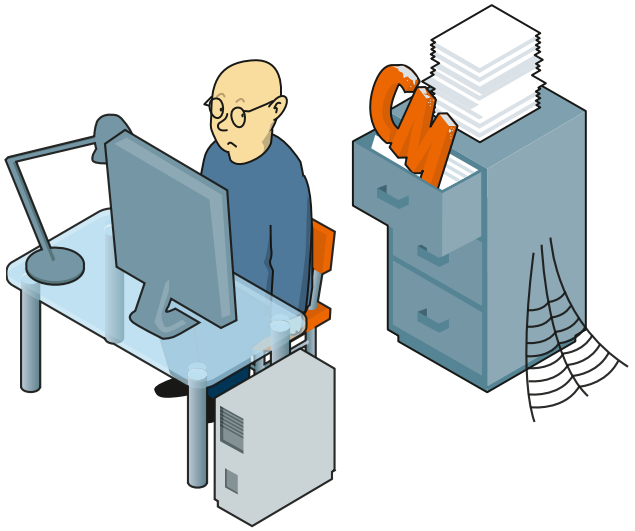
Istället ska lösningen bestå av att man dels gör utvecklarna uppmärksamma på effekterna av deras agerande, dels tillsammans med utvecklarna tar fram ett leveransförfarande som inte känns onödigt komplicerat och där de ser värdet av processen. När man så har provat den nya leveransprocessen kan man successivt automatisera den.

Det är viktigt att någon eller några i organisationen känner ett ägarskap för leveransprocessen. I större organisationer är det vanligt med en CM-ansvarig som får detta på sin lott, men det viktigaste är att det är någon som har kontakt med verksamheten och som kan upptäcka när processen behöver justeras.

KOMMA IGÅNG

Så kommer du igång:

1. Samla en grupp människor som är intresserade av de CM-relaterade områden där ni upplever svårigheter, t.ex. leveranser, versionshantering, ärendehantering.
2. Se till att gruppen inte blir helt verktygsorienterad. Det är viktigt att fundera över arbetssätten.
3. Se till att skapa CM-relaterade aktiviteter som är naturligt kopplade till övriga processer på företaget (utvecklingsprocess, projektstyrning, kravhantering etc). En isolerad CM-process blir nämligen så gott som alltid en hyllvärmare som inte löser några problem på sikt.
4. Kom igång med införandet så snart ni har identifierat områden att förbättra. När ni börjar nysta i problemen kommer ni att hitta nya frågor att lösa. Håll förbättringsarbetet levande, inför förändringar steg för steg och utvärdera resultaten.
5. Tillsätt en CM-ansvarig som kan ha övergripande koll på området. Denne ska inte vara den som utför alla CM-relaterade arbetsuppgifter utan vara experten som hjälper resten av organisationen. Det är inte nödvändigtvis en heltidsroll.



LÄS MER

Litteratur

- Wayne Babich, *Software Configuration Management: Coordination for Team Productivity*, ISBN 0201101610
- Steve Berczuk & Brad Appleton, *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*, ISBN 0201741172
- William J Brown, Hays "Skip" McCormick, Scott W Thomas, *Anti Patterns and Patterns in Software Configuration Management*, ISBN 0471329290
- Anne Mette Jonassen Hass, *Configuration Management Principles and Practice*, ISBN 0321117662
- Mario E. Moreira, *Adapting Configuration Management for Agile Teams: Balancing Sustainability and Speed*, ISBN 0470746637

ITQs CM-utbildning

- http://www.itqnord.se/educations/configuration-management/view

SNESCM

- http://snescm.cs.lth.se/Common/index.html

CM-kurs på LTH

- http://fileadmin.cs.lth.se/cs/Education/EDAN10/

CM, Configuration Management, är konsten att hålla reda på versioner, komponenter och kompletteringar för att därigenom undvika krångel, problem och misstag. Det är viktigt att inte betrakta CM som ett nödvändigt ont – istället bör man fokusera på de fördelar som skapas, t.ex. att

- ledtiderna i utvecklingsarbetet minskar
- det ges möjlighet att arbeta med mer "kreativt kaos"
- organisationen får ett bättre helhetsgrepp
- det blir enklare att dela upp och hantera mjukvaran i moduler.

CM-ansvarig personal måste besitta en stor förståelse för hela utvecklingsprocessen, ligga ett steg före i arbetet samt – inte minst – förstå och respektera hur olika team jobbar.

Softhouse Consulting

Stockholm
Tegnérgatan 37
111 61 Stockholm
Tel: 08-410 929 50
info.stockholm@softhouse.se

Göteborg
Lilla Bommen 1
411 04 Göteborg
Tel: 031-760 99 00
info.goteborg@softhouse.se

Malmö
Stormgatan 14
211 20 Malmö
Tel: 040-664 39 00
info.malmo@softhouse.se

Karlskrona
Campus Gräsvik 3A
371 75 Karlskrona
Tel: 0455-61 87 00
info.karlskrona@softhouse.se

www.softhouse.se

SOFTHOUSE