

# TIO STEG TILL HÖGRE UTVECKLARPRODUKTIVITET

Patrik Fredriksson

Inspirerad av min kollega Tobias Fors blogserie "Men vad kan jag göra?" beslutade jag mig för att dela med mig från min samling av tips för att öka din produktivitet som systemutvecklare. I motsats till herr Fors kan jag dock inte lova 99 punkter, men jag vågar nog lova att jag kommer till tio.

Först kan man förstås fundera lite runt varför det är så viktigt att bli effektivare i sin roll som utvecklare. Varför denna besatthet av produktivitet?

Ja, det är en rimlig fråga, antar jag. Det korta svaret är att om du inte är snabb nog så kommer någon annan snart att göra ditt jobb. Om det blir någon i Indien eller din närmaste granne (eller möjligen både och, om du bor i Indien), spelar inte så stor roll. De allra flesta av oss rör oss i en starkt konkurrensutsatt miljö. Förmågan att snabbt leverera fungerande mjukvara av hög kvalitet är en nyckelegenskap hos en framgångsrik utvecklare.

**Förmågan att snabbt leverera fungerande mjukvara av hög kvalitet är en nyckelegenskap hos en framgångsrik utvecklare.**

De lätttrörliga metoder som fler och fler av oss använder fungerar bäst om vi kan jobba i relativt korta iterationer. En väl avstämmd iterationslängd är nyckeln till att balansera önskemålen från beställaren om att kunna påverka och ändra bland kraven, mot önskemålen från utvecklarna att få arbeta i lugn och ro.

Låt oss ta Scrum som exempel. I Scrum är iterationer på trettio dagar vanliga. Tanken här är att du och dina projektmedlemmar ska prestera fungerande mjukvara av hög kvalitet som levererar verklig affärsnytta på trettio dagar. Om det tar fem dagar bara för att bygga mjukvaran så kommer detta arbetssätt aldrig fungera.

Dessutom är det kul att vara snabb! Kanske eftersom det inte är helt vanligt

Nog om det, nu kör vi. Utan någon speciell ordning, tio steg som gör dig till en mer produktiv utvecklare.

## 1. Lär känna din problemomän.

Med största sannolikhet är du framförallt anlitad för dina kunskaper som programmerare. Men som utvecklare av mjukvara jobbar vi nästan alltid i ett problemområde, en domän, som inte nödvändigtvis är teknisk, och inte sällan okänd.

Så det första du bör göra när du kommer i kontakt med en ny domän är att lära dig så mycket det bara går om domänen i fråga. Det finns ofta massor med resurser att tillgå, köp en bok, leta artiklar på Internet, se en film etc. Be någon domänexpert i projektet att tipsa om bra källor! Bjud in dig hos de domänexperter du hittar och fråga "Varför?" tills de slänger ut dig från sina rum. Jag lovar att de allra flesta med glädje kommer att avvara tid för att svara på dina frågor, de kommer att uppskatta att "någon från IT" faktiskt är intresserad av vad de gör.

När du väl har börjat utforska området och lära dig mer, se till att kontinuerligt hålla dig uppdaterad om vad som händer inom domänen.

Du kommer omgående att kunna dra nytta av din nyvunna kunskap. Du kan använda den till exempel för att tillsammans med tekniker som domändriven design utforska domänen ytterligare och skapa kraftfull mjukvara som verkligen adresserar domänens specifika situation. Du kan förenkla kommunikationen med domänexperter och beställare. Du kan också lättare ställa de rätta frågorna i din strävan att maximera affärsnytta.

## 2. Skaffa den bästa utvecklingsmiljö som finns. Och lär dig använda den.

Jag är ledsen, men Emacs fungerar inte längre. Du må ha varit en UNIX-älskande fanboy hela ditt medvetna liv, men att inte utnyttja kraften i en modern utvecklingsmiljö är oprofessionellt.

"Ja, men jag programmerar faktiskt i Ruby, och Ruby är världens bästa programmeringsspråk, det är så enkelt, men ändå kraftfullt och bra, att det inte behövs någon utvecklingsmiljö". Jag hävdar att det där helt enkelt inte är sant. Om du är som jag; är kontinuerlig refaktorering en integrerad del i ditt dagliga arbete som utvecklare. Du gör det utan att ens reflektera över det som en aktivitet längre. Att byta namn på en metod utan stöd från utvecklingsmiljön kommer garanterat att dämpa din framfart. I värsta fall kommer det att stoppa dig från att göra det bara för att slippa plågan av en manuell sök/ersätt.

Ja, det dynamiska typsystemet i populära språk utgör en utmaning för verktygsleverantörerna. Men det tog många år av mediokra utvecklingsmiljöer för Java innan IntelliJ visade hur man gör. Vi tar det vi kan få. Men se till att lära dig verktyget. "A fool with a tool is still a fool", som det så fint brukar heta.

### 3. Versionshantera.

Skaffa ett effektivt versionshanteringssystem och lär dig det. Subversion är ett fritt tillgängligt system som tillhandahåller det du behöver för att börja versionhantera din kod, bara att hämta hem och installera. Stöd finns med största sannolikhet redan för din favoritutvecklingsmiljö.

Genom att kombinera effektiv versionshantering med enhetstester kan du säkert och effektivt utforska alternativa vägar och nya idéer utan att vara rädd att gå vilse. Skaver någon del av koden? Gör en ny gren, testa alternativa vägar. Knåda koden. En framgångsrik refaktorering försäkrar att du kan hålla hög hastighet i ditt utvecklingsarbete även framöver. Ledde förändringen till en försämring? Inga problem, rulla tillbaka förändringarna. Prova igen.

Ett system för versionshantering hjälper dig även att bolla flera olika leveranser och fortsatt utveckling. En väl fungerande process för versionshantering garanterar att du aldrig behöver fixa en bug mer än en gång.

### 4. Lär dig vilka färdiga kodbibliotek och API:er som finns. Använd dem.

En sak som skiljer en produktiv senior utvecklare från en långsam nybörjare är kunskapen om vilken kod som redan finns tillgänglig, och förmågan att använda den. Om du kommer på dig själv med att tänka något i stil med "det här problemet borde någon annan behövt lösa" så är det med stor sannolikhet också fallet. Lär dig att använda core-biblioteken såväl som andra tillgängliga API:er. Skriv aldrig mer en egen loggningsklass, en egen databaspool, en egen metod för att fylla ut strängar med mellanslag eller en klass för att hantera tid och datum. Och aldrig, aldrig ett eget ramverk för att spara objekt i databaser. Allt detta finns. Testat och förfinat av i flera fall tusentals användare. Google är din vän.

### 5. Bygg med en knapptryckning.

Vi har nosat på det tidigare. Om man ska kunna arbeta med korta iterationer och snabb återkoppling från testare, användare och beställare krävs en effektiv byggprocess. Allt som ska behöva stå mellan din källkod och en färdig modul som går att produktionssätta är en knapp på en Wikisida.

Kombinera dina enhetstester och ditt versionshanteringssystem med ett system för kontinuerlig integration så är du alltid redo att snabbt göra en ny leverans. Fixa buggen, sätt en tag i versionshanteringssystemet, tryck på knappen på Wikisidan och hämta en kopp kaffe. När du är tillbaka är din leverans klar. Testad och med nygenererad dokumentation.

Det finns många mer eller mindre bra verktyg för detta. Prova!

## 6. Google.

Google är din vän. Rätt använt kan Google, eller annan kraftfull sökmotor, förvandlas till världens mest sofistikerade och uppdaterade supportorganisation. Behöver du förkovra dig i din domän? Försöker du lösa en knepig bugg? Söker du exempel för ett API du inte använt förut? Letar du efter en effektiv algoritm? Behöver du veta hur långt 12 engelska mile är i googol ljusår? Googla! Med lite träning kan du bli oerhört effektiv i att använda detta hjälpmedel för att lösa allehanda problem.

## 7. Lär dig använda kraften i ditt operativsystem.

Kortkommandon och liknande genvägar är utmärkta hjälpmedel i din strävan efter produktivitet.

Om det operativsystem du använder saknar det du behöver eller är ickeintuitivt att använda, ja då får du helt enkelt se till att dekorera det med vad som behövs.

Total Commander för Windows må vara ruskigt fult och småknepigt att lära sig. Men en person som behärskar verktyget hinner göra en nästan osannolik mängd filrealterade operationer på den tid det tar för dig att starta Utforskaren.

På samma sätt kan en van användare av Quicksilver i Mac OS X hinna starta två program, byta låt i iTunes, göra en snabb beräkning, checka ut en fil från Subversion, skicka ett mail, samt boka ett möte i Google Calender innan den genomsnittliga användaren ens lyckats starta Word. Och dessutom göra det med stil.

## 8. Fokusera.

Om du är som jag så är du lite för nyfiken, och dessutom aningen disträ. Det är helt enkelt lätt att tappa fokus.

Om du sitter i ett kontorslandskap kan ett par hörlurar kanske fungera. Eller varför inte lämna rummet om du behöver bearbeta ett speciellt utmanande problem.

Använd de funktioner som finns i de verktyg du använder för att avskärma dig från e-post och andra program som vill stjäla ditt fokus. Använd fullscreen-läget till exempel.

Slå av avisering av nya meddelanden. Det finns även verktyg som specifikt skapats för att hjälpa till med fokusproblemet. Jag skriver detta i ett verktyg som heter WriteRoom, helt fantastiskt. För Windows finns motsvarande funktion i Dark Room.

### **9. Se till att ha bra hårdvara.**

Små skärmar, långsamma diskar, alltför begränsat minne och till och med långsamma tangentbord kan kraftigt inverka på din leveransförmåga. Det som är tråkigt med dessa problem är att man som programmerare i en större organisation är beroende av någon annan för att kunna lösa dem. Ibland står en oförmåga att förstå vad det kostar att inte ha bra verktyg mellan dig och en ny dator. Andra gånger kan det vara något märkligt policybeslut av typen "alla ska ha 17-tumsskärmar, det är rättvist". Men som alla utvecklare vet, en 17-tums skärm med dålig upplösning må vara utmärkt för att tråkla ihop presentationer i PowerPoint, men rätt så olämplig för programmering.

Det här kan vara saker som ser dyra ut på pappret (även om det nu för tiden knappast är så länge), men jämför man med vad din tid som utvecklare kostar, ja, då är det en struntsumma. Jobbar du snabbare med mer minne installerat så kommer denna kostnad tjänas in på mycket kort tid. Datorkraft är billigt, hjärnkraft inte fullt lika så.

### **10. Omge dig med bra kolleger.**

Genom att bygga team tillsammans med andra högpresterande utvecklare kommer du att inse att ni tillsammans kan leverera prestanda som väl överstiger summan av de ingående delarna. Ni sporrar varandra, kan accelerera inlärning och snabbt lösa problem som uppkommer. Det här är kanske det svåraste tipset att genomföra. Men försök. Du kommer bli belönad.

OK, jag skickar med två bonustips:

### **11. Sträva alltid efter att lära dig mer.**

Ju mer du vet, desto bättre grund har du för dina beslut. Försök omsätta dina nyvunna kunskaper i praktisk handling snabbt, för att testa om de håller och förstärka inlärningen. Omfamna förändring! Men se upp för CV-driven utveckling, särskilt vad gäller ny teknik. Det senaste är inte alltid det mest lämpliga. Glöm inte heller att man faktiskt rätt så sällan behöver använda alla tekniker som finns på en gång.

### **12. Var utvilad.**

Jag kan inte nog understryka detta. Systemutveckling är en aktivitet som kräver fokus, mental närvaro och kreativt tankearbete. Inget av detta finns på plats om du inte är utvilad. Se till att få den sömn du behöver

Efter att ha läst dessa tips kanske du tycker att det här var en lång lista av självklarheter som redan appliceras av alla systemutvecklare. Du använder i alla fall tipsen redan idag! Det är ju utmärkt i så fall. Tipsen är baserade på mina erfarenheter, och dessa säger mig dessutom att alltför ofta saknas viktiga komponenter som skulle kunna öka produktiviteten markant.

Jag vet, baserat på erfarenhet, att ju fler delar från denna lista på tips som du applicerar, desto mer produktiv kommer du att bli. ■ ■ ■



**Patrik Fredriksson arbetar som konsult och mentor i mjukvaruutveckling på Citerus. Patrik är speciellt intresserad av design och arkitektur i Java samt produktivitet i utvecklingsprojekt. Maila honom på patrik punkt fredriksson at citerus punkt se.**