Programmering 2: Dokumentation m.m.

Magnus Silverdal

10 maj 2019

1 En struktur för dokumentation av ett programmeringsprojekt

Det är möjligt att tänka sig en mängd olika varianter av dokumentationsmallar för ett programmeringsprojekt. Det beror på projektets karaktär och på personlig preferens. Det viktiga är att ha en struktur att följa. Grunden för dokumentationen är en vetenskaplig rapport. Det ska finnas en problembeskrivning (Syfte/frågeställning) där problemet presenteras. Sedan kommer en beskrivning av lösningen, komplett eller i delar. Lösningen kan bestå av flödesscheman, klassdiagram, algoritmbeskrivningar, pseudokod och beskrivande text. Det är viktigt att förklara tanken bakom lösningen eftersom uppdragsgivaren eller kunden inte kan förväntas vara intresserad av att analysera den färdiga koden. Här ska också beskrivningen av API't finnas, en redogörelse för de olilka klassernas och metodernas funktion och användning. Ofta delar man in de olika delarna av lösningen i systembeskrivning och en algoritmbeskrivning. Uppdelningen är naturlig eftersom det ofta är möjligt att byta algoritmer i ett befintligt system, alternativt ändra systemet (t.ex. datastrukturer) utan att ändra i algoritmerna. Den andra delen av resultatet är testkörningar och fallstudier som presenteras och analyseras. Som sista del i rapporten skrivs en diskusion med reflektioner över uppgiften, din lösning o.s.v.

Till detta kommer framsida, abstract, innehållsförteckning, källförteckning och bilagor i de fall detta behövs. Det är också god sed att ta med en användarhandledning av något slag och den bör finnas lättillgänglig, till exempel i samband med sammanfattningen eller innan lösningen. Om det har karaktär mer av en manual passar den bättre som en bilaga. Givetvis ska källkod och annat data bifogas rapporten på lämpligt sätt, t.ex. som en zip-fil.

1.1 Struktur

1. Framsida
2. Abstract/sammanfattning
3. Innehållsförteckning
4. Syfte/frågeställning
5. Teori, bakgrund
6. Beskrivning av lösningen
    - Användarhandledning
    - JavaDoc/API
    - Systembeskrivning/Klassdiagram
    - Algoritmbeskrivningar
7. Testkörningar
8. Diskussion
9. Bilagor

Väljer man att redovisa sin dokumentation med hjälp av en Wiki bör motsvarande element också finnas med men dispositionen blir lite annorlunda.

2 Tips på teknik

Beroende på vilket system som rapporten skrivs i behövs olika verktyg, men några är alnmängiltiga och här kommer några bra resurser att titta närmare på.

2.1 Flödesdiagram/Klassdiagram

Ett bra och gratis verktyg för att skapa olika typer av diagram är Dia. Där finns möjlighet att exportera till olika format och även till LaTeX-kod för de som är intresserade av det.

2.2 Algoritmbeskrivningar

Oavsett hur algoritmbeskrivningar är skrivna är det god sed att skriva dem i en indenterad punktlista. För att få till en punktlista i flera nivåer i LaTeX fungerar inte standardversionen av `enumerate` men det finns en ganska enkel lösning.

2.3 Källkod

I normalfallet presenteras inte källkoden eftersom den oftast distribueras med rapporten. Ibland kan det dock finnas ett behov av att ta med en del av källkoden i rapporten. Ett enkelt sätt att återge källkod är att använda ett typsnitt med konstant teckenbredd (sk monospace). Det är ännu bättre att försöka få med indentering och formatering så bra som möjligt. I Word är en möjlighet att använda en formatmall men den bästa metoden att skapa ett objekt och sedan importera källkoden från sin editor, alternativt kopiera koden med formatering från editorn. Vad som fungerar bäst beror på vilken editor som används. I LaTeX finns paketet `listings` som tillåter import av källkod från en mängd olika språk. Utseendet kan styras med en rad olika inställningar och källkoden får samma status som en figur, en tabell eller liknande.

2.4 Javadoc för att automatiskt generera API'n

Sedan JDK 1.5 finns verktyget javadoc för att skapa dokumentation i HTML-format för klasser och metoder. Verkyget finns inbyggt i de flesta IDE och använder information i kommentarer i källkoden för att skapa ett API som liknar javas officiella API. De kommentarer som ska läsas av javadoc inleds med en extra asterisk, dvs `/**` istället för `/*`. Titta i de automatgenererade stubbarna i Eclipse eller Netbeans för att få ett hum om formatet. Det finns ett antal taggar tillgängliga för att beskriva arvstruktur, metoders argument och returtyper och så vidare. Du bör vara bekant med `@param`, `@return`, `@exception` och `@see`. Notera också att javadoc behandlar den första meningen i en kommentar speciellt och att du bör ha en tom rad mellan texten och dina taggar. Läs mer på Oracles sida om javadoc eller i den här lite mer lättlästa sammanfattningen. Eftersom javadoc genererar htmlkod av kommentarerna är det helt giligt att lägga in html-kod i kommentarerna i källkoden.

Ska den HTML-kod som javadoc genererar inkluders i ett dokument finns olika alternativ. Ett är att använda en doclet som genererar API't i LaTeX-format eller i pdf-format

Eftersom det till stor del handlar om att parsa olika typer av textfiler (som java, html eller latex) finns det mängder av små hack att testa. Här är ett som skapar diagram från ascii-kod i javadoc-kommentarer.

3 Avslutning

Att skiva en rapport över ett programmeringsprojekt behöver inte vara speciellt krävande. Genom att använda rätt verktyg under arbetets gång kommer rapportskrivandet att bli i stort sett automatiserat. Med bra diagram och javadoc är större delen av jobbet gjort innan rapporten ens är påbörjad.