

Ramverk räddar .NET-projektet



NetSD är ett ramverk som räddar projekt från galopperande utvecklings- och förvaltningskostnader. Ramverket stöder programmeraren genom att grundstrukturen i en applikation snabbt kan genereras upp.

Ramverk räddar .NET-projektet

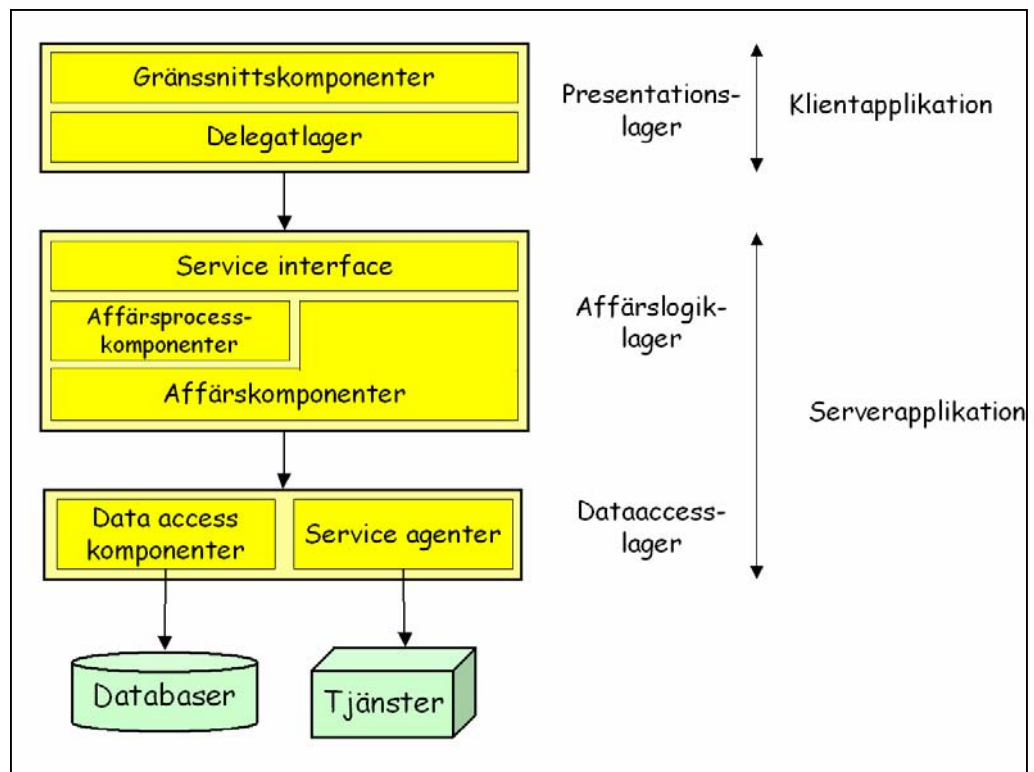
Val av arkitektur har stor betydelse genom att det styr ett systems totala livscykelkostnad. Hänsyn måste framför allt tas till hur förutsättningarna kan komma att förändras i framtiden. Vilka klienter skall kunna komma åt applikationen? Hur stort blir antalet användare?

I efterhand är det mycket kostsamt att rätta till eller byta ut en felaktig arkitektur. Ofta är det så dyrt att man tvingas leva med de misstag som görs i det inledande utvecklingskedet.

NetSD är ett ramverk som bygger på en skiktad arkitektur. Skiktningen är standardiserad med ett givet antal lager oberoende av applikation. Fördelen med en enhetlig arkitektur framkommer speciellt i förvaltningsfasen, där förvaltare ofta inte är samma personer som utvecklat applikationen. Med en standardiserad skiktning blir det lätt för nya personer att förstå arkitekturen och därmed snabbt bli produktiva.

En skiktad miljö gör det möjligt att återanvända samma serverkod i olika applikationer. Oberoende om klienten utgörs av en mobiltelefon eller PC.

Arkitekturen i NetSD är skiktad enligt följande modell. Det är en SOA, en Service Oriented Architectur, där all logik har en given hemvist.



NetSD är standardiserat med avseende på antal lager, oberoende av applikation.

Gränssnittskomponenterna består av visuella komponenter, t ex formulär och dialoger i en Windowsapplikation.

Delegatlagret fungerar som serverapplikationens ställföreträdare på klienten och kapslar in de tekniska detaljerna vid anrop av serverapplikationens tjänster. Gränssnittskomponenterna anropar metoder i delegatlagret som skickar anropet vidare till serverapplikationens service interface.

Service interface är serverapplikationens gränssnitt och exponerar dess tjänster mot omvärlden. Service interfacet innehåller alltså logik för att hantera kommunikation men ansvarar också normalt för autentisering och behörighetskontroll. Om tjänsterna skall exponeras med olika gränssnitt, skiljer man ofta ut kommunikationslogiken i separata service interface. Medan logik för autentisering och behörighetskontroll flyttas till ett underliggande fasadlager.

Affärsprocesskomponenterna är processororienterade och kan utnyttja en eller flera affärskomponenter.

I affärskomponenter samlas logik för att hantera affärsobjekten, dvs vad som brukar kallas affärslogik. Affärskomponenter anropar objekt i dataaccesslagret för att hämta och spara data. De kan även anropa andra affärskomponenter.

Görs anrop mellan affärskomponenter är det viktigt att se till att man inte skapar cirkulära referenser, d v s komponent A anropar komponent B som i sin tur anropar komponent A. Vanligtvis utser man i detta fall komponenter som är "samordnande" och därmed får man komponenter i affärslagret som har olika roller.

Dataaccesskomponenterna innehåller logik för läsning och skrivning av data till databasen. Dataaccesskomponenter får inte anropa varandra utan eventuell styrlogik skall ligga i affärskomponentlagret. Komponenterna i detta lager svarar normalt mot en tabell och innehåller metoder för att göra select, insert, update och delete mot tabellen.

Service agenter utgör gränssnitt mot externa tjänster.

Web Services för kommunikation

I NetSD används Web Services för kommunikation mellan server och klient. Det innebär att HTTP och SOAP används som överföringsprotokoll respektive meddelandeformat. Fördelen är att Web Services ger ett standardiserat gränssnitt, som kan användas även från t ex en Unix-plattform.

För att förbättra säkerheten används Web Services Enhancement, som bygger på antagna och föreslagna standards. Bl a ingår kryptering av överföringen.

I vissa applikationer kan det av prestandaskäl vara nödvändigt att använda ett effektivare protokoll än Web Services. NetSD ger då möjligheten att istället utnyttja .NET Remoting. Det enda som behöver bytas är då service interfacet mot klienten.

Att köra klient- och serverapplikation i separata processer ställer vissa krav på de parametrar och returvärden som skickas över processgränserna genom att dessa måste gå att formatera som en ström. Vid användning av TCP/binary måste databärarna vara serialiserbara medan användning av SOAP innebär att parametrar och returvärden måste kunna representeras som XML-dokument.

I .NET finns två principiellt olika val för databärare

- Dataset som är en inbyggd ADO.NET-klass
- Egna affärsobjekt (custom business entities)

Dataset som kan ses som .NETs inbyggda affärsobjekt. De har många fördelar genom att de är serialiserbara och i övrigt anpassade för att transportera data via remoting. De kan också kopplas till visuella komponenter via data binding och hantera uppdateringar via sk delta-dataset.

Behovet av att skapa egna affärsobjekt beror mycket på typ av applikationen. Vid tillämpningar av registervårdskaraktär är behovet mindre. Huvudfunktionen är här att läsa upp en träffbild på klienten, öppna och förändra ett objekt och slutligen spara förändringen. Svarar objekten mot enkla master-detail databastabeller, typ order/orderrad, finns normalt inget behov att bygga ett orderobjekt. Här duger funktionen hos dataseten bra.

Men om applikationen hanterar så komplexa datastrukturer att de naturligen beskrivs som egna affärsobjekt, har man två val.

Antingen skapas objekten i serverapplikationen och transporteras till klienten. Fördelen är att data har samma representation på klient och server. Å andra sidan måste hantering av förändringar göras med egen logik.

Det andra valet är att transportera data från server till klient via dataset och bygg objekten på klienten.

Minimera handskrivna kod

Ett mål med NetSD är att minimera den mängd kod som programmeraren behöver skriva. Skälet är naturligtvis att minska antalet fel och förenkla underhållet.

För transaktionshantering finns basklasser som underlättar programmeringen. Genom attribut på funktioner och klasser sköts hanteringen automatiskt.

Stöd finns för både lokala transaktioner med en singular datakälla och distribuerade transaktioner mot flera datakällor. Byte mellan transaktionsmodellerna kan göras med några få kodändringar.

Basklasserna är skrivna i C#, vilket är det språk som NetSD är baserat på.

Ofta finns krav på att den färdiga applikationen skall fungera mot flera olika databaser. Då får normalt programmeraren skriva mycket kod för att hantera olikheter i leverantörernas olika drivrutiner.

Problemet är löst i NetSD genom att införa ett skikt mellan applikation och databas som hanterar olikheterna. Samma kodbas kan alltså användas för både MS SQL Server, Oracle, Sybase etc.

Stöd för utvecklaren

En annan bärande tanke med NetSD är att hjälpa utvecklaren att följa arkitekturen. Därför finns möjlighet att i Visual Studio generera ut grundstrukturen för applikationen, en prototyp. Inbyggda regler kontrollerar att funktioner inte implementeras i fel lager. Exempelvis får en affärskomponent inte innehålla funktioner som anropar databasen.

NetSD består också av en handbok som innehåller anvisningar och beskrivningar av arkitekturen. Bland annat finns arbetsbeskrivningar för hur kod skall dokumenteras. NetSD bygger på den inbyggda XML-kommenteringen i .NET samt anvisningar för hur XML-element ska användas. Mallar och standarder finns som stöd för dokumentationen.

I NetSD finns även rekommendationer för hur koden skall skrivas. Det gäller bl standards för hur olika element ska namnges; variabler, klasser etc. Ramverket innehåller också regler för strukturering av kod, projektinställningar i Visual Studio samt tips och anvisningar för kodning.

Att bara införskaffa ett ramverk av typ NetSD räcker dock inte för ett lyckat utvecklingsprojekt. Framförallt måste projektmedlemmarna utbildas i ramverket.

Det är också viktigt att följa upp hur arkitekturen följs genom kodgranskning, speciellt under inledningen av utvecklingsarbetet. Granskningen kan vara en avgörande framgångsfaktor. Ofta blir noggrannheten i granskningen inledningvis avgörande för kvaliteten i fortsättningen.

Lyckade projekt

Ramverket NetSD utgör idag grunden för flera lyckade projekt.

I en av tillämpningarna vill kunden utnyttja affärskritisk information på ett bättre sätt.

Kundens data finns utspridda i ett stort antal databaser på olika driftplatser och underhålls av ett stort antal program. Ansvaret för programmen och databaserna är delat mellan olika leverantörer och anställda. Problem finns med både åtkomst, kontroll och datakvalitet.

Behov finns även att förbättra möjligheterna att analysera informationen; exempelvis att identifiera målgrupper för marknadsaktiviteter, skapa bättre kunskap om kunders beteende etc.

Lösningen består i att upprätta ett centralt datalager som innehåller "all" information. Det centrala datalagret är också ansvarigt för att replikera och transformera information.

Applikationen är utvecklad med NetSD och som databas används MS SQL Server.

Decerno

Decerno bygger pålitliga system – ett löfte som innebär flera saker. Vi bygger stabila verktyg som skänker trygghet i arbetet. Samtidigt framtidssäkra system, som är lätta att anpassa när verksamheten förändras.

Pålitlighet innebär också säker leverans på utsatt tid och inom givna kostnadsramar. För att skapa pålitliga system krävs ett långsiktigt tänkande. Både teknik och metoder måste stödja kvalitet i form av bl a testbarhet, enkelt underhåll och tillförlitlighet. Decerno har därför ett långt drivet kvalitetsarbete.

Under de tjugo år Decerno varit verksam har vi drivit över hundra projekt till ett lyckat slut.