

Softwarefactory

Wat is het doel van een softwarefactory?

Bedrijven willen hun toepassingsontwikkeling verder automatiseren om de softwarekwaliteit te verbeteren en de ontwikkeltijd te verkorten. Nieuwe software moet aan hoge eisen beantwoorden, en snel inzetbaar zijn. Met een softwarefactory structureer je het ontwikkelproces en maak je het beheersbaar. De projectmanagers kunnen de evoluties nauwgezet opvolgen. De ontwikkelaars krijgen betere ondersteuning. Een softwarefactory zorgt dus voor kwaliteitsvolle, snelle en beheersbare toepassingsontwikkeling.

Vanwaar de naam *factory*?

Maak gerust de vergelijking met een echte fabriek. Machines en robots vervangen er de manuele activiteiten. Meetpunten controleren de status van de productie en testen de kwaliteit. Precies hetzelfde gebeurt in een softwarefactory. Door bijvoorbeeld automatisch code te genereren of met templates te werken vermijd je handmatig codeerwerk. Dat werkt sneller en correcter. Vanuit een goed ontwikkelmodel kun je gemakkelijk nagaan of de code aan de doelstellingen beantwoordt: levert ze de juiste resultaten op? Door regelmatig en vanaf het begin van het ontwikkelproces tests in te bouwen, weet je snel of de code in orde is.

Waaruit bestaat een softwarefactory?

Een softwarefactory is een totaalaanpak die ontwikkeltools, richtlijnen, templates en systemen perfect op elkaar afstemt. Universele kant-en-klaaroplossingen die je bij iedere klant kunt neerzetten, zijn er niet. Het ene team legt de nadruk op tools voor codegeneratie, het andere op kwaliteitscontrole of projectopvolging. De implementatie van een factory is een zoektocht naar de juiste componenten voor de uitdagingen van uw bedrijf.



Wat is het verschil tussen een softwarefactory en een applicationframework?

Een *framework* situeert zich bijna uitsluitend op het werkterrein van de ontwikkelaar. Het is een bibliotheek van code die de developer helpt om in een vastgelegde structuur te werken. Een framework schermt hem af van de complexiteit en de vele mogelijkheden van .NET of J2EE. Een *factory* gaat verder. Ze omvat het totale ontwikkelproces, van de behoeftenanalyse tot de implementatie. De doelstellingen van een factory zijn ook breder: verbetering van de productiesnelheid en de kwaliteit.

Hoe ondersteunt een softwarefactory de levensloop van een toepassing?

- De eerste stap is de bepaling van de **vereisten en werkwijzen** op basis van templates. De informatie wordt centraal bewaard, met toegang voor ieder teamlid – bv. op een intranetportaal. Ook alle latere documenten, zoals analyses en bugrapporten, komen hier terecht. Richtlijnen bepalen hoe de documenten beheerd worden en hoe de teamleden ze gebruiken. Een projectmanager bijvoorbeeld definieert vanuit de analyse *iteraties* en *workitems* of opdrachten. Die worden geïntegreerd in een broncontrolesysteem, zodat de ontwikkelaar ze ziet in zijn eigen ontwikkelomgeving. Afgewerkte opdrachten vinkt hij aan. De vooruitgang is daardoor perfect onder controle. Een factory is dus een win-win voor het project én de ontwikkelaars.
- Een volgende stap in de factory kan de **generatie van softwarecode** zijn. Daarvoor zijn metadata nodig. Die leg je vast in een grafisch model, dat je uittekent met tools die je in de ontwikkelomgeving integreert. Populair zijn bijvoorbeeld de *domain specific languages* van Microsoft. Vanuit het model wordt de code gegenereerd, in overeenstemming met de door de klant gekozen architectuur en het gebruikte framework. Een groot deel van het werk is geautomatiseerd, met een minimale kans op fouten. Het model heeft nog een andere belangrijke functie: het vereenvoudigt de communicatie tussen de ontwikkelaars en de mensen die de vereisten beheren.

Ordina Belgium offices

B-3001 Heverlee, Interleuvenlaan 15H • tel. +32 (0)16 27 00 80
 B-3560 Lummen, Bosstraat 52/2 • tel. +32 (0)13 38 00 00
 B-9820 Merelbeke, Guldensporenlaan 84/1 • tel. +32 (0)9 280 23 90
 B-2627 Schelle, Boomsesteenweg 28 • tel. +32 (0)3 866 00 22

www.ordina.be

- In een factory kun je ook een **continuous integration system** opzetten. De code zit dan in een centraal broncodecontrolesysteem, zoals Microsoft *Team Foundation Server* of IBM *Rational ClearCase*. Daarnaast draait een systeem dat op geregelde tijdstippen de broncode downloadt en compileert tot *executable code*. Het testteam kan snel met de toepassing aan de slag en geeft feedback via het projectportaal. De projectmanager volgt de kwaliteit op de voet en maakt nieuwe workitems aan wanneer het nodig is. Automatisering en kwaliteit gaan hand in hand.

Hoe verhoogt een softwarefactory de kwaliteit van de code?

De kwaliteit verbetert onder meer door het snellere verloop van het ontwikkelproces. Er is dus extra tijd voor kwaliteitscontrole. De belangrijkste techniek om de kwaliteit te controleren, is werken met *unittesting*. De ontwikkelaar bepaalt vooraf precies wat hij van een stuk code verwacht. Bij het testen werkt hij altijd met dezelfde data. De verwachtingen liggen dus vast, en de data ook. De code is de enige variabele. Leidt een test tot een fout resultaat, dan betekent dat automatisch dat de code niet werkt zoals voorzien. De oorzaak opsporen wordt een stuk eenvoudiger. Door de combinatie van *unittesting* met *continuous integration* ontdek je sneller kwaliteitsproblemen, en los je ze sneller op. Het projectportaal speelt opnieuw een belangrijke rol.

Zijn er verschillende vormen van softwarefactory's?

De doelstellingen en uitgangspunten van de klant verschillen, en dus de softwarefactorymodellen ook.

- Teams die werken met ontwikkeling in offshoring willen de nieuwe broncode van nabij opvolgen. Een centraal broncodesysteem is voor hen dus heel belangrijk.
- Ook de methodologie van de klant bepaalt het uitzicht van de factory. Bij een iteratieve Scrum-methodologie ziet de structuur van de templates voor documenten en workitems er anders uit dan bij een klassieke watervalmethodologie.
- De keuze van het ontwikkelplatform speelt een kleinere rol. Daarvan moet een factory juist zoveel

mogelijk onafhankelijk zijn. De factory benadert ontwikkeling vanuit een hoger abstractieniveau.

- De belangrijkste differentiator voor softwarefactory's is het *servicelevel* van de toepassingen waarvoor je de factory inzet. Voor grote bedrijfskritische applicaties ligt de lat voor kwaliteit en beheersbaarheid heel wat hoger dan voor kleine applicaties met een kort leven. Die kun je niet belasten met een zware factory. Net zoals je een applicatie met een hoog servicelevel niet kunt aanpakken met een te lichte factory.

Hoe bouw je een softwarefactory op?

Het is een goed idee om te starten vanuit een toepassing die ontwikkeld is zonder een factory. Via een doorlichting van het manuele ontwikkelproces bepaal je hoe het efficiënter kan. De keuze van tools en methodologie is heel belangrijk, net als de begeleiding van de projectteams bij de ingebruikname ervan. De teamleden moeten de onderdelen van de factory immers correct toepassen. Waak er over dat de technologie in dienst staat van de mensen en niet omgekeerd. Structureer zorgvuldig de documenten en templates, met aandacht voor de wisselwerking met de onderliggende methodologie. In een volgend stadium kunnen testtechnieken, modelering en codegeneratie op punt gesteld worden. In een parallel traject bouw je een broncodecontrolesysteem en een continuous integration system op.

Hoe kan Ordina u helpen?

Een softwarefactory is geen out-of-the-box product. Het Ordina Competence Center Software Factories ondersteunt talrijke tools, en begeleidt klanten bij het bepalen van de factory en de implementatie ervan. Ordina leidt u ook op om met uw softwarefactory-omgeving te werken. De belangrijkste taken van het Competence Center zijn R&D, productevaluatie, opbouw van kennis over tools en methodologieën, kennisdeling en projectbegeleiding.



ORDFAQsoffacNED1558

Ordina Belgium offices

B-3001 Heverlee, Interleuvenlaan 15H • tel. +32 (0)16 27 00 80
 B-3560 Lummen, Bosstraat 52/2 • tel. +32 (0)13 38 00 00
 B-9820 Merelbeke, Guldensporenlaan 84/1 • tel. +32 (0)9 280 23 90
 B-2627 Schelle, Boomsesteenweg 28 • tel. +32 (0)3 866 00 22

www.ordina.be



CONSULTING | ICT | OUTSOURCING