



# Fejlesztési módszerek és eszközök

a Verhás & Verhás Szoftver Manufaktúrában © 2009

## Java technológia, üzleti filozófia

A Verhás & Verhás Szoftver Manufaktúrában (VVSC) Java technológiával fejlesztünk. Ez nem csak azt jelenti, hogy a fejlesztéseket Java nyelven készítjük (ami nem is mindig igaz, hiszen ha úgy adódik, akkor használunk más nyelveket is), hanem azt a filozófiát, ami a Java nyelvhez kapcsolódik.

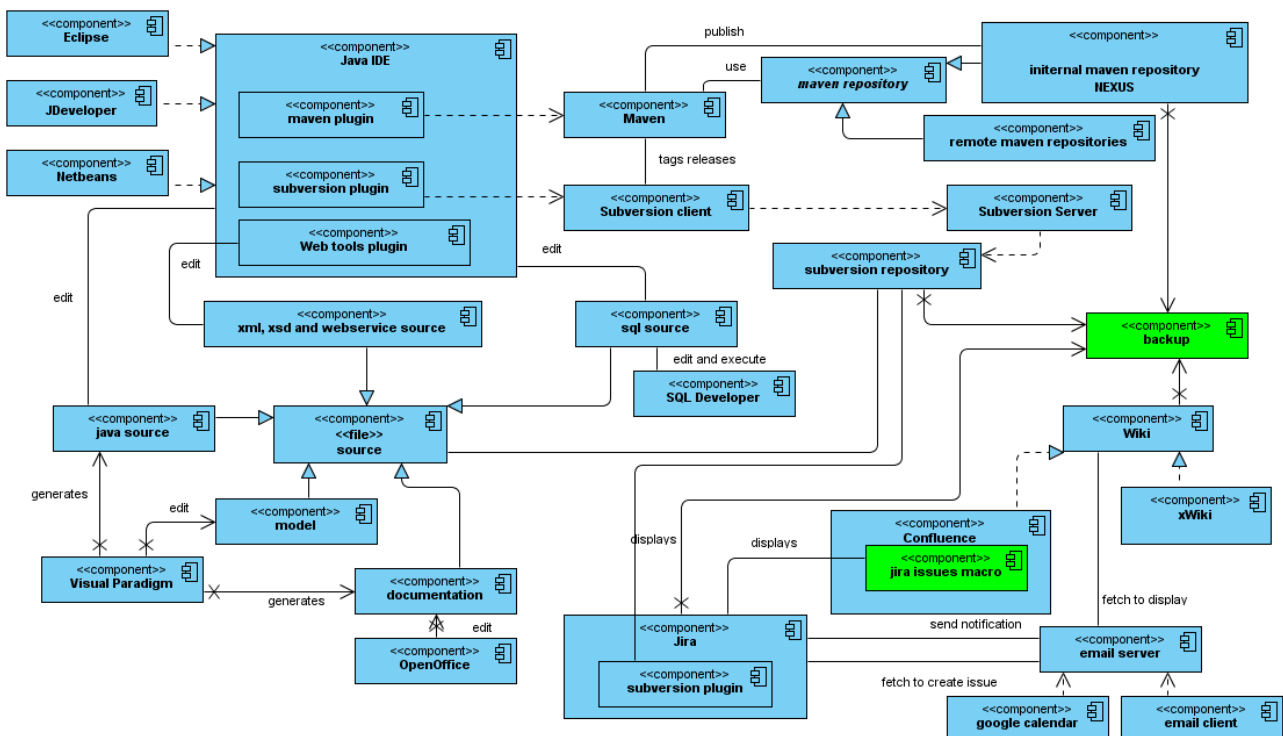
Ez a filozófia az open source. Ez nem feltétlenül azt jelenti, hogy a programok ingyenesen és szabadon használhatók. Ahhoz, hogy értéket lehessen termelni annak az ellenértékét meg kell fizetni, a semmiből nem lesz valami, valamint „nincs olyan, hogy ingyen ebéd”.

A szabad forrás számunkra azt jelenti, hogy a tudás (ami konkrét esetekben a programok forráskódját jelenti) szabad. Amikor az ügyfél fizet, akkor nem azért fizet, mert valamit mi tudunk, amit ő is tudhatna, de nem tud, mert titokban tartottuk, hanem a szaktudásért, amelyet szállítunk neki.

technológiához kapcsolódó ipari szabványosítási eljárásokban az új megoldások nem úgy keletkeznek, hogy egy fejlesztő cég kifejleszt egy új rendszert, és utána azt lehet használni minden előnyével és hátrányával (hibáival), hanem szabványok születnek, amelyeket sok szállító delegált szakemberei határoznak meg, és ezeket több szoftverfejlesztő is implementálja. Ezáltal az egyes modulok használatával a fejlesztett környezet nem kötődik az egyes szállítókhoz, a modulok aránylag könnyen cserélhetők a rendszerben.

Ez a filozófia az ügyfeleink számára garantálja, hogy a megoldások szabványosak, a szabványok a felhasználók széles rétegeinek igényét elégíti ki, és nem egy szállító üzleti igényeinek megfelelően lettek kialakítva, és minimális a gyártóhoz való kötődés (vendor lock-in) veszélye.

Nem gondoljuk azt, hogy a másik nagy vonulat, a Microsoft technológiára épülő megoldások műszakilag rosszak (nem rosszak, sőt). Nem gondoljuk azt, hogy ezt



Ezzel a szaktudással az ügyfelek (szakemberei) is rendelkezhetnek, de az esetek többségében gazdaságilag nem éri meg nekik ebbe a szaktudásba fektetni, mert nem ez a fő üzleti tevékenységük. Ehelyett megbíz minket, akik a szoftverfejlesztés szakértői vagyunk, és megfizeti a szaktudásunkat és hozzáértésünket.

A nyílt forrás ezen kívül nem csak a szaktudást jelenti, hanem az alkalmazott rendszerek nyitottságát is. A Java

a filozófiát azért követik a Java technológiai szállítók, mert ők a jók, és a Microsoft pedig a rossz, gonosz. Azt gondoljuk, hogy a két irányzat egy piaci, gazdasági, technológiai környezetben alakult ki olyan módon, ahogy azt az egyes piaci szereplők piacon elfoglalt helye diktálta, lehetővé tette. Viszont reméljük, hogy ezzel a filozófiával, piaci attitűddel a megrendelőink jól járnak.

És most nézzük meg, hogy milyen eszközöket használunk.

## **Eszközök**

A fejlesztés során tervezünk (UML), programozunk (Java kód szerkesztő), tesztelünk, tárolunk (verzió kontroll) fordítunk, kiadást kezelünk (release menedzsment), dokumentálunk, hibajegyeket kezelünk.

## **UML tervezés**

Az UML tervezéshez korábban a Poseidon (Gentleware) szoftvert használtuk, de 2007-ben, miután a szoftver egy közepes méretű projektnél használhatatlanul lassú volt áttértünk a Visual-Paradigm szoftverre. A szoftvert nem találomra választottuk. Olyan szoftvert kerestünk, amelyik jól használható, megfizethető árú, Windows-on és Linux-on egyaránt használható. A szoftver kiemelkedő tulajdonsága (és szerintünk ez alapvető lenne minden UML tervező rendszernél), hogy az UML-t nem mint ábrákat kezeli, hanem mint modell struktúrát, amelyet természetesen meg is tud jeleníteni diagrammokon.

## **Java IDE**

A Java kód fejlesztéshez általában a NetBeans fejlesztőeszközt használjuk (jelenleg), de korábban használtunk Eclipse és ritkábban JDeveloper eszközt is. Nem vagyunk NetBeans, vagy Eclipse „hívők”: azt használjuk amelyik elérhető verziója éppen jobb, de azért nem cserélgetünk évente többször.

## **Maven**

A Java projektek build-jéhez Maven-t használunk. Ismerjük az Ant programot is és a Maven-t is, ezért használunk Maven-t. A Maven használata Ant után egy kicsit nyűgös, de ha a fejlesztő hajlandó lemondani a megszokásairól, és elfogadja a Maven által kínált (és szabvány szerűen elterjedt) konvenciókat, akkor sokkal egyszerűbben használható. Lerövidül egy projekt elindításának az ideje, nem kell minden egyes projekthez Ant scriptet írni, még ha azok hasonlóak is.

A Maven-t a Sonatype NEXUS programjával együtt

használjuk.

## **JIRA, Confluence**

A projektmenedzsment támogatására az egyes feladatok nyilvántartására az Atlassian cég JIRA termékét használjuk. A dokumentumokat ugyancsak az Atlassian wiki szoftverében a Confluence-ben tartjuk nyilván.

A JIRA szoftvert annyira sikeresen használjuk, hogy volt olyan partnerünk, aki saját projektmenedzsmentjéhez is a JIRA rendszerünket kezdte el használni, és ez folytatódott akkor is amikor mi már nem is vettünk részt a projektben :-)

Adtunk el JIRA rendszert projekt menedzsmentre műszaki berendezések kiskereskedelmével foglalkozó középállalatnak, gyakorlatilag testreszabás nélkül, és olyan alkalmazást is kialakítottunk JIRA alapokon felhasználva a termék programozhatóságát, amelyek teljesen speciális munkafolyamatokat hajtottak végre (pl. követelés menedzsment).

## **Subversion**

A program kódokat, dokumentációkat (és az üzleti dokumentumokat) SVN-ben tároljuk. Ehhez használjuk a NetBeans illetve az Eclipse beépített kliens programjait, a maven scm-jét, parancssoros SVN klienst, Windows-on a TortoiseSVN programot, valamint hamarosan a Groowiki programot, ami webes felületen jeleníti meg az SVN tartalmát.

A Maven és a subversion rendszer össze van hangolva, így a Maven-nel automatikusan tudunk új snapshot és verzió release-t készíteni. Az SVN a JIRA rendszerrel is együttműködik, minden egyes commit esetében meg lehet adni, hogy melyik JIRA issue-hoz tartozik a beküldés, és ez megjelenik a JIRA issue-ban is.

## **Kapcsolat, cégalapítás**

Verhás & Verhás Szoftver Manufaktúra Kft.  
Verhás Péter, ügyvezető  
<http://www.verhas.com>  
Fax: +36(1)5772383