# VYUŽITÍ PŘÍPADŮ UŽITÍ A SCÉNÁŘŮ PŘI TVORBĚ SOFTWAROVÉHO DESIGNU

**Josef Pavlíček**
Sun Microsystems Czech s.r.o., Evropská 33e, 160 00 Praha 6- Dejvice, ČR, Josef.Pavlicek@Sun.Com

## Abstrakt:
Vytvořit uživatelsky přívětivý softwarový produkt je cílem úsilí mnoha vývojářských týmů. Takový produkt se lépe prodává i nabízí. Pod pojmem uživatelsky přívětivý chápeme softwarový produkt nejen hezký na pohled, ale také intuitivní nebo snadný na obsluhu. Často však stojíme před problémem, kdy jednotlivé faktory (intuitivnost, snadnost obsluhy, hezký vzhled) jsou si protichůdné. Finlání řešení by pak mělo být "elegantí". K tomu je zapotřebí využít vhodných inženýrských postupů. O těchto postupech budeme hovořit.

## Abstract:
A lot of developer's teams are trying to develop user friendly software. User friendly product is easy to use. It goes better to offer and sell it. As a term user friendly we understand easy to use with suitable appearance. This software should be intuitive too. Unfortunately, often we are solving problem that these factors (intuitiveness, easy to use, suitable appearance) are in the conflict. The final solution should be "elegant". It is necessary to use some suitable engineering procedures for. This paper speaks about them.

## Klíčová slova
UseCase, Scénář, Design, Software, Produkt, Testování použitelnosti, IDE, NetBeans

## Key Words
UseCase, Scenario, Design, Software, Product,Usability test, IDE, NetBeans

## Úvod
Tvorba softwaru je složitý proces. Lze jej přirovnat k stavbě domu či výrobě automobilu. Ale domy lidstvo staví od počátku věků civilizace. Povozy a nejrůznější stroje již vyrábíme několik století. Máme s nimi již hezkou řádku let zkušeností. Software je ale doménou posledních několika desetiletí. I když pro dnešní generaci je software zcela samozřejmostí (vždyť obsluhuje varné desky, telefony, výtahy, je prostě všude), jeho tvorba je stále opředena mýty a pověrami. Může to být způsobeno i jeho „nehmotnou" podstatou. Na software si není možno sahnout, není možné jej ohnout ani zvážit. Možná právě tato jeho podstata vede k řadě omylů. O některých z nich si povíme v dalších kapitolách. Především však budeme hovořit o způsobu návrhu uživatelského prostředí.

## Komu má být software určen je nutné vědět

Pomocí softwaru nelze uspokojit požadavky všech uživatelů. [1] Alan Cooper na svých přednáškách často ukazuje absurdnost této předsavy pomocí spojení vlastností 3 automobilů. Terénního, sportovního a rodinného. Spojením vlastností těchto vozů vznikne něco, co se nebude líbit nikomu. Nikdy nebude možné vytvořit softwarový produkt natolik obecný, aby vyhovoval všem potenciálním uživatelům. Je tedy nutné jej navrhnout pouze pro určitou skupinu uživatelů. Pro tuto skupinu by měl být naprosto dokonalý (intuitivní, snadno použitelný, s hezkým vzhledem). Ostatní uživatelé nebudou zcela uspokojeni. To je však realita, které se není možné vyhnout. Proto v dnešní době především softwarový design získává stále větší důležitost.

## Tvorba softwarového designu

Softwarový design, především pak design uživatelského prostředí[1], je velmi důležitý. Dobře jej navrhnout není možné „od stolu“. Víme-li pro koho je určen, pak je jeho návrh jednodušší. I tak se ale ukazuje metoda použití případů užití, scénářů, grafického návrhu a testování použitelnosti jako účelná. O sběru uživatelských požadavků na softwarové produkty jsme již hovořili na konferenci Tvorba Software 2005. Předpokládáme, že se s jeho postupy čtenář mohl seznámit. Tudíž se letos zaměříme na případy užití, scénáře a na cílový desing. O testování použitelnosti se zmíníme pouze z důvodů ucelení této přednášky.

## Dekompozice uživatelských požadavků do případu užití

Během práce uživatele s uživatelským prostředím vzniká řada případů, ve kterých se uživatel musí rozhodnout, co udělá dál. Rozdělíme-li celý jeho pracovní postup (anglicky jej nazýváme „flow“) na jednotlivá přerušení a rozhodování co udělat dál, získáme tím případy užití.

## Případ užití

Každý případ užití je samostatný blok činnosti uživatele. Případy užití jsou psány z pohledu uživatele.

## Scénář

Během případu užití uživatel předpokládá nějakou odezvu od systému. Systém se pak uživateli snaží nerůznějším způsobem odpovědět či pomoci. Chování systému v tomto případě nazýváme scénářem. Odpovědi systému by neměly narušovat tzv. flow uživatele. Scénář popisuje, jak systém reaguje na chování uživatele

1Samozřejmě nikterak nepodceňujeme architekturu, použité technologie atd.

## Vzor případu užití:

- Uživatel píše kód v Editoru IDE
- Během psaní kódu uživatel očekává odpovědi Editoru na připavené zkratky
    - např: tab+space vyvolá nápovědu
- Uživatel předpokládá
    - že mu Editor IDE poradí, kde udělal chybu
    - kontextovou nápovědu podle typu činnosti
    - automatické zobrazení módu činnosti
    - automatické doplnění kódu
    - atd...

## Vzor scénáře:

- Uživatel píše kód v Editoru IDE
- Stavový řádek IDE zobrazuje
    - pozici kurzoru uživatele
    - zobrazuje mód psaní
        - INSert (kurzor bliká a má tvar tenké čárky)
        - OVeRwrite (kurzor bliká a má tvar silné čárky- „X“ pixelů)
            - systém přepisuje řetězec v editoru jiným řetezcem zadávaným uživatelem
    - Během psaní kódu systém nabízí automaticky doplňení kódu
        - za tečkou zobrazí seznam tříd či metod (záleží na kontextu práce)
    - Během psaní kódu systém podtrhává syntakticky špatné řetězce
        - Pokud uživatel zajede kurzorem na řádek s chybou, systém uživateli zobrazí ve stavové řádce IDE informaci o chybě

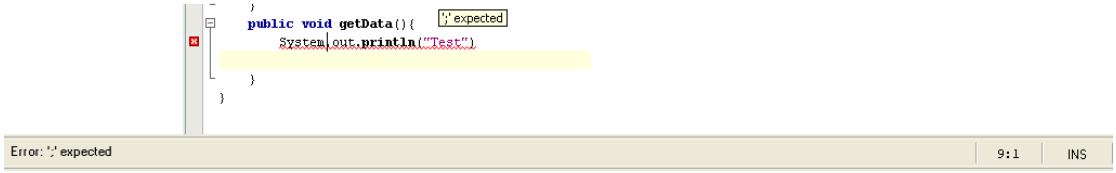
# Varianty návrhu uživatelského prostředí

Návrh uživatelského prostředí je možné dělat nejrůznějším způsobem. Každý způsob má řadu výhod ale i omezení.

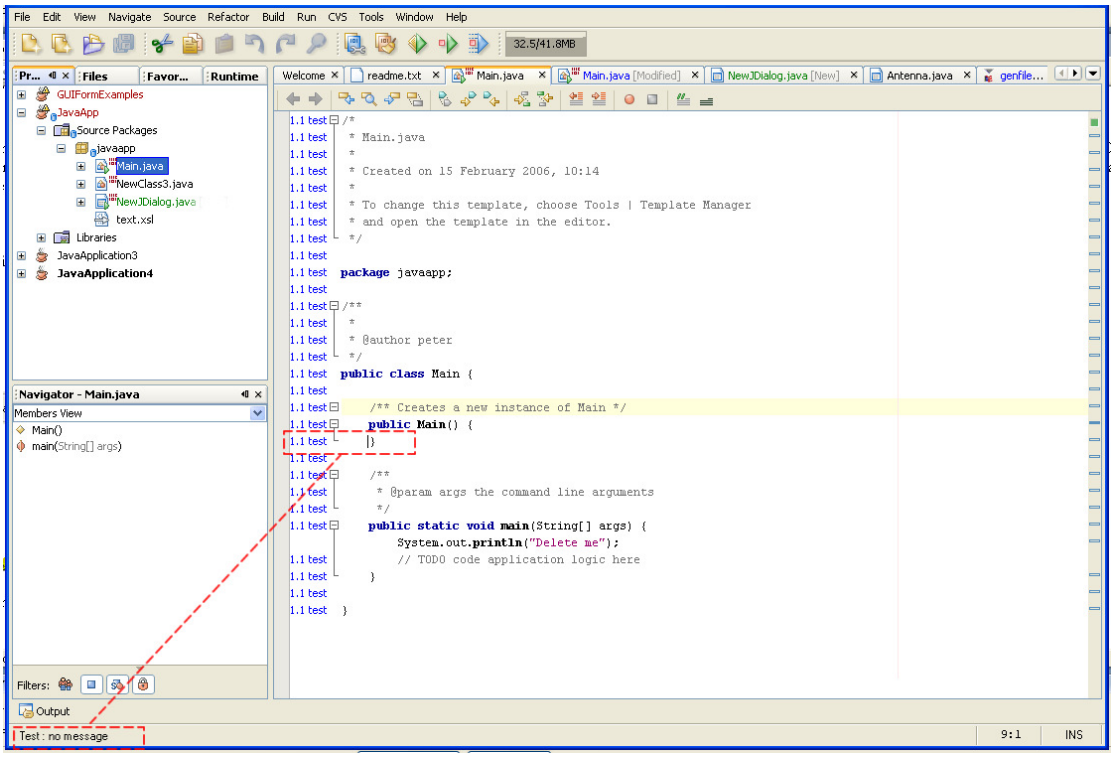
## Z nejběžnějších můžeme vyjmenovat:

- **Slovní popis** - je rychlý, chybí jeho názornost
- **Design na papíře** - je rychlý, těžko se převádí do digitální podoby, budí neseriozní dojem
- **ASCII návrh** - je rychlý, je dostatečně názorný. Bohužel je hodně schematický
- **Grafický návrh** - je velmi názorný. Jeho tvorba zabírá hodně času
- **Funkční prototyp** - je dokonale názorný. Spotřebuje velké množství času a nakonec se vyhodí

Obr 1: zobrazení chyb ve stavové řádce




Obr 2: Zobrazení anotací




## Testování použitelnosti uživatelského prostředí

Testování použitelnosti[2] uživatelského prostředí se provádí v okamžiku, je-li některý ze způsobu návrhu hotový. Velmi dobrý způsob je vytvoření uživatelského prostředí na základě grafického nebo ASCII návrhu. Je-li uživatelské prostředí ve stavu použitelnosti (beta verze), pak je vhodné provést testování použitelnosti. Byly-li případy použití a scénáře správně navrženy a grafický vzhled uživatelského prostředí vhodný, neměl by uživatel v době testování použitelnosti přerušit své flow. Dojde-li k jeho přerušení je vhodné studovat proč.

2Provádí se ve speciálních laboratořích. Věnovali jsme mu již několik přednášek. Domníváme se, že jej není nutné uvádět zde.

# Závěr

Je jistě celá řada způsobů  návrhu uživatelského prostředí. V mnohých případech postup zde popsaný nemusí být ideální. Může být pomalý. V případě tvorby komplexního vývojového prostředí, sloužícího široké komunitě programátorů (ale ne všem programátorům) k psaní programů v Javě, se tento postup ukazuje jako výhodný. Záleží na čtenáři, zda tento postup použije, či jej zavrhne a využije jiný.

## LITERATURA

[1] ALAN COOPER, The inmates are running the asylum. A Division of Macmillan Computer Publishing 201 West 103rd Street, Indianapolis, Indiana 46290
[2] BEYER HOLTZBLATT, Contextual design, Morgan Kaufmann Publisher, ISBN – 1-55860-411-1
[3]MIKE KUNIAVSKY, Observing the user experinece, Morgan Kaufmann Publisher, ISBN 1-55860-923-7