Prototypy a testování

Návrh řešení

Návrh řešení by měl obsahovat:

- Popis cíle systému
- Identifikaci uživatelů
- Vymezení hranic systému
- Závěry z analýzy požadavků na systém
- Návrh hlavních funkčních celků (subsystémů)
- Události, na které systém musí reagovat
- Odhad a návrh datové základny
- Technické řešení
- Řešení prototypu

Plán vývojových prací:

- Kdo bude projekt řídit
- Složení vývojového týmu
- Seznam úkolů a jejich priority
- Harmonogram
- Nároky na zdroje a finance

Prototypování

Prototypy slouží k řešení problémů analýzy požadavků.

Důležité: Prototyp je model aplikace, část aplikace nebo její vizualizace za účelem předvedení nebo otestování. Je to mezikrok mezi specifikací a fungujícím systémem**.**

.

Prototypem mohou být **reálné modely** aplikací, ukázky jak bude aplikace ve finále vypadat. Uživatelům pomohou získat představu, jak bude systém vypadat a jak se s ním bude pracovat. Měl by být uživateli předložen čím jak nejdříve, aby uživatel viděl, jak systém vypadá, jak reaguje a jaké je uživatelské rozhraní a mohl se k systému vyjádřit.

Nevýhody:

- Přílišná pozornost na uživatelské rozhraní místo na systém
- Nucení k používání reálného kódu v prototypu, pokud není dostatek času
- Manažeři mají pocit, že produkt už je hotový

Prototyp tedy slouží ke komunikaci se zákazníkem (uživatelem), cílem je pochopení, jak systém funguje a jak jsou realizovány požadavky:

- Uživatel vidí, jak systém pracuje
- Odhalení chyb nebo chybějících částí specifikace

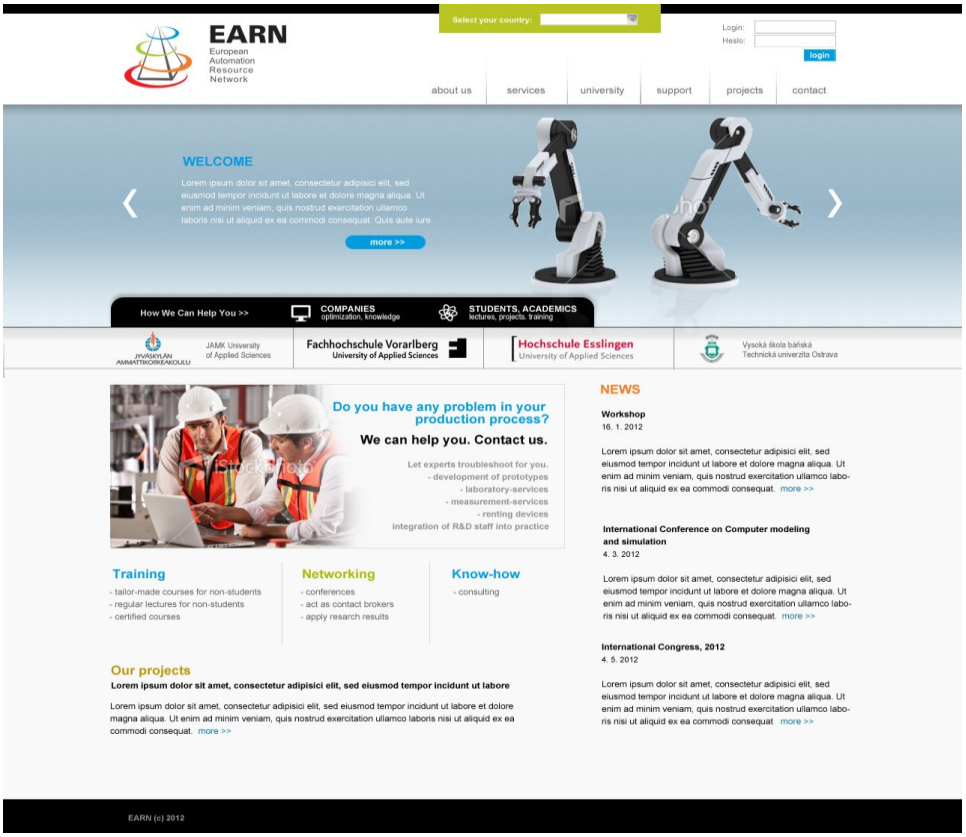
Přístupy k prototypování

- **Evolutionary prototyping** – cílem je dodat fungující aplikaci, prototyp hned na začátku a pak je předefinováván až k finálnímu produktu

- **Throw-away prototyping** - prototyp se týká požadavků, které nejsou dostatečně jasně definovány, slouží k pochopení a definici požadavků; vývoj systému s prototypem nesouvisí

Evoluční prototyping je vhodný tam, kde ve specifikaci nelze vše předem definovat. Má blízko k „agilnímu“ vývoji. Silnou stránkou je řešení chybějících, zmatených či špatně definovaných požadavků. Uživatelé tím, že od počátku pracují s aplikací, jsou současně proškoleni na užívání. Nevýhodou je složitější řízení vývoje.

Throw-away prototyping slouží k předvedení a experimentům, není vytvářen s tím, že bude fungovat jako finální systém. Výhodou je rychlá tvorba.



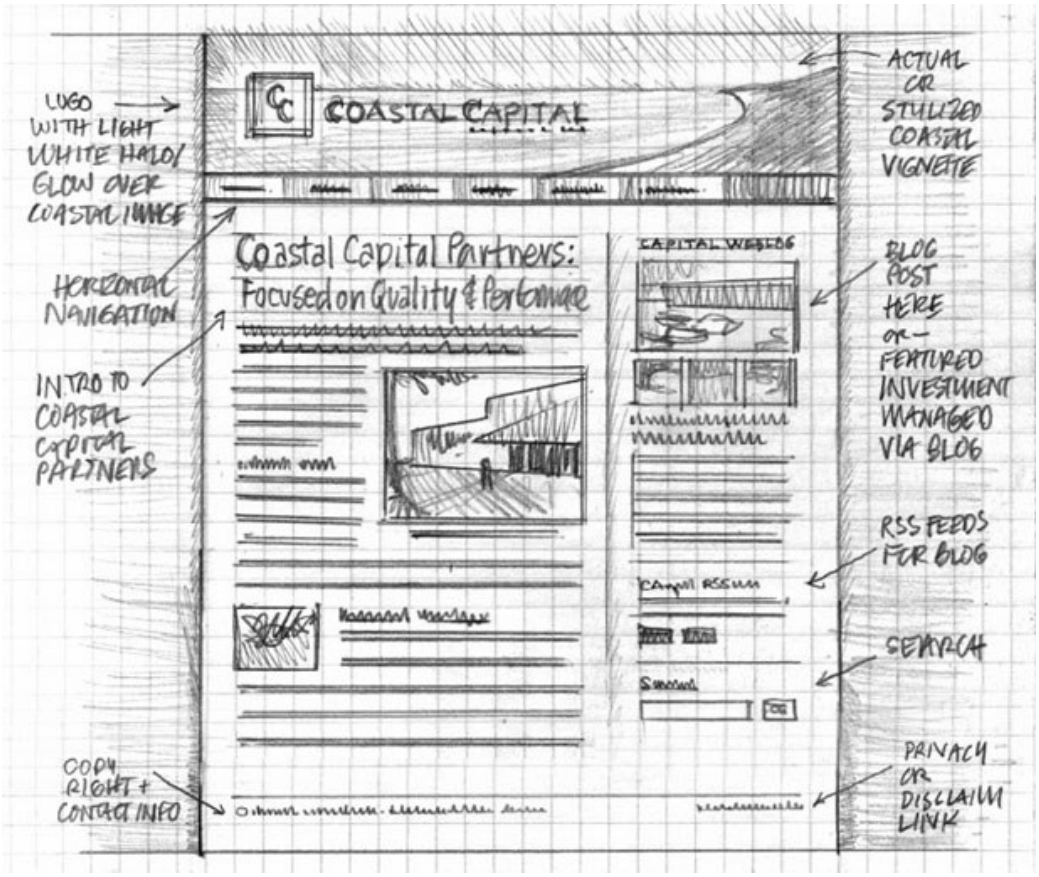
Obrázek 1 Ukázka mock-up webové aplikace
[Zdroj: projekt EARN, © Řepka – Danel]

Mock-up

Mock-up je nakreslený vzhled obrazovek (papír, elektronicky) s omezenou nebo nulovou funkcionalitou. Na obrázku 2 vidíme ukázku mock-up pro webovou aplikaci, nakreslený v CorelDraw.

Wireframe

Wireframe neboli **drátěný model** aplikace ukazuje její strukturu a vizuální řešení – rozkreslené obrazovky a umístění jednotlivých elementů. Časté je užití drátěných modelů u webových aplikací. Wireframe může mít různé podoby, od skic na papíře až po použití speciálních softwarových nástrojů pro vytváření drátěných modelů (např. Visual Paradigm).



Obrázek 2 Ukázka wireframe webové aplikace
[Zdroj: http://www.webdesignerdepot.com/2009/07/using-wireframes-to-streamline-your-development-process/]

Typy wireframů

- **Textový wireframe**. Jedná se o např. odrážkami vytvořený seznam položek a funkcí, které by měl web obsahovat. Je to nejjednodušší možný wireframe.
- **Blokový wireframe**. Tento již využívá bloky (čáry, obdélníky, čtverce), aby definoval rozmístění a velikost jednotlivých prvků. Ty pak mohou obsahovat text, který popisuje jejich obsah nebo funkci.

- **Podrobný wireframe**. Podrobný wireframe dopodrobna popisuje každý v něm použitý element, je vytvářen v přesných proporcích, je popsána funkčnost, obsah, prostě vše.
- **Proklikávací wireframe**. Tento může být nástavbou blokového nebo podrobného wireframu a jeho obrovskou výhodou je, že jednotlivé stránky (např. návrh titulky, přehledu a detailu zboží, košíku a objednávky) jsou propojeny, takže lze mezi nimi klikat jako ve skutečném webu. S jeho pomocí lze např. již předběžně provést malé uživatelské testování.

Prototype fidelity

Termínem fidelity označujeme přesnost prototypu.

Low-fidelity prototyp – nekompletní, ukazuje jen část, hrubá skica, obrázek
High-fidelity prototyp – prototyp je blízko finálnímu produktu.

Horizontální a vertikální prototypy

- **Horizontální prototypy** – zaměření na funkcionalitu a souvislosti na úkor implementace
- **Vertikální prototypy** – jen část systému s důrazem, aby prototyp směroval k finálnímu produktu

Testování

Testování SW má podstatný vliv na kvalitu dodaného produktu.

Náklady na odstranění chyby stoupají, v čím pozdější fázi životního cyklu aplikace je chyba nalezena.

Na druhé straně, vytvořit zcela bezchybný SW není možné. Je tedy třeba zvolit vhodný kompromis mezi množstvím testů a uvedením produktu do provozu.

Důvody neopravení chyby:

a) Není dost času
b) Není to chyba (nesprávně pochopená funkce)
c) Oprava chyby je riskantní (oprava může způsobit poškození již ověřené části)
d) Oprava nestojí za to (pravděpodobnost výskytu je nízká)

Typy testů

Black Box a White Box testy

Test typu **Black Box** znamená, že testujeme aplikaci, od které nemáme k dispozici zdrojový kód.

U **White Box** testů je zdrojový kód k dispozici a můžeme tedy k testování využít znalost vnitřní struktury nebo simulovat určité stavy pomocí debugeru.

Grey Box - omezená znalost interních datových a programových struktur za účelem navrhnutí vhodných testovacích scénářů, které se realizují na úrovni black box (např. testy interface).

Statické a dynamické testování

Statický test znamená např. revizi kódu, revizi analytických podkladů, test funkční specifikace.

Dynamické testování znamená, že testujeme aplikaci, která je spuštěna.

Testy specifikace

Je specifikace:

- Úplná
- Správná (nejsou v popisu chyby?)
- Přesná a jednoznačná
- Konzistentní (nejsou jednotlivé části v konfliktu?)
- Relevantní (jsou navržené funkce potřebné?)
- Proveditelná (je to realizovatelné v rámci rozpočtu a času?)
- Testovatelná

Obvyklé chyby specifikace:

- Obsahuje kvantifikátory vždy, nikdy, každý, žádný, všichni…
- Snaha někam dotlačit – je nabíledni, určitě, evidentně…
- Seznam není úplný – a tak dále, například…
- Vágní specifikace – někdy, něco, obvykle, zpravidla, většinou…
- Nejednoznačné kvantifikátory - dobrý, laciný, malý, stabilní
- If rozhodování bez specifikace else větve

Dynamické testování černé skříňky

a) **Test-to-pass (test splněním)** – kontrolujeme, zda aplikace vyhovuje minimální požadované funkčnosti
b) **Test-to-fail (test selháním)** – snaha najít chybu nebo aplikaci „shodit“

Testování dat (vstupů)

a) Test hraničních podmínek – délky řetězců, zadání čísla větší než je maximum/minimum atd.
b) Test subhraničních podmínek – testování vnitřních omezení (např. položky integer)
c) Test výchozích, prázdných, nevyplněných a nulových hodnot
d) Test na zadání neplatných nebo nesmyslných údajů

Testování stavů (logiky aplikace)

Součástí těchto testů by mělo být také

a) **testování race condition (souběhu).** Příkladem je např. uložení stejného dokumentu z dvou aplikací, současný přístup do databáze, sdílení portů, sdílení tiskárny, stisk klávesy v průběhu zavádění programu…
b) **test opakováním** – co se stane, když se nějaká akce neustále opakuje (např. klikání na tlačítko pro zobrazení formuláře…), hledají se např. memory leaks (zahlcení paměti)
c) **stresové testování** – test běhu aplikace při nejhorších možných podmínkách (málo paměti, málo místa na disku…)
d) **load test (zátěžové testy)** – simulace velkého počtu uživatelů, velkého počtu přístupů, velkého množství transakcí…

Testování konfigurace

Běží aplikace na všech možných kombinacích hardwarových komponent?

Tento typ testu je obtížný na realizace obvykle nemáme většinu HW k dispozici. Lze využít publikovaných HW specifikací…

Testování kompatibility

Dokáže vytvořený software pracovat souběžně s jiným SW?

Backward compatibility – zpětná kompatibilita – pracuje s předchozími verzemi tohoto software? Je datově kompatibilní?

Forward compatibility - dopředná kompatibilita – bude kompatibilní s budoucími verzemi?

Testování multijazyčné podpory

Lokalizace není synonymem pro slovo překlad, ale je to soubor celého komplexu opatření. Zahrnuje kromě vlastní aplikace také lokalizaci dat (např. použití UNICODE), lokalizaci horkých kláves, třídění textů, směr psaní, lokalizace textů v grafice, formátování dat.

Problémy s formátováním dat u multilanguage aplikací:

a) měrné jednotky
b) číselné údaje
c) měna (symbol a jeho umístění)
d) zobrazení datumu
e) zobrazení času (12/24)
f) kalendář (juliánský, gregoriánský, arabský…)
g) adresy (např. tvar PSČ)
h) telefonní čísla
i) velikost papíru a obálek pro tisk

Počítalo se s lokalizací od počátku návrhu?

Zásadní pro lokalizaci je nemít texty uprostřed kódu!

Usability test (Test použitelnosti)

Test použitelnosti - jak snadné je produkt zprovoznit a používat ho (snadnost, efektivita, přesnost, zapamatovatelnost…), formou black boxu.

Je aplikace:

- intuitivní na ovládání?
- konzistentní? (např. co se týče použité terminologie, umístění tlačítek…)
- flexibilní?
- dodržuje standardy? (např. pozice ano – ne tlačítek v dialozích)
- dostupná pro handicapované?
- V souladu s dokumentací?

Unit testy

Unit (jednotka) – samostatně testovatelná část aplikace. V objektově-orientovaném programování odpovídá většinou třídě. Používají se také při regresním testování.

- Provádí vývojář
- Byly použity vhodné algoritmy, návrhové vzory ?
- Ověření správnosti fungování dílčí části kódu
- Stubs, mock object, fakes, …
- Scrum, TDD – unit testy se vytvářejí před kódem

Downgrade test

Je možné se vrátit k předchozí verzi?

Test instalace

Lze aplikaci nainstalovat a poté bez problémů odinstalovat?

Long Term Test
Jak se aplikace chová, je-li trvale spuštěna.

Akceptační test
Testy prováděné při předání aplikace; většinou součástí smlouvy, někdy dohodnuté na počátku projektu. Idea je, že pokud aplikace projde akceptačními testy, měla by být objednatelem převzata a zaplacena.

Performance test
Test rychlostí odpovědi aplikace na činnost uživatele nebo na události.

Security test
Test zabezpečení aplikace. Součástí mohou být také testy implementace integritních omezení.

Recovery test
Vzpamatuje se aplikace po selhání HW? Co se stane, když se vypne počítač v situaci, kdy aplikace je spuštěna?

Regresní testy
Opakované provádění testů na již otestovaných částech. Snaha zjistit, zda oprava chyby nezpůsobila nefunkčnost jiné části aplikace.

Automatizace testování
- Monitor
- Ovladač (skript nahrazující klávesnicové vstupy)
- Maketa (stub) – přebírá výsledek sw (např. místo tiskárny se tisk ukládá do souboru)
- Záznam maker
- Generátory „šumu“
- Nástroje pro stresové a zátěžové testy (např. Microsoft Stress)
- Analytické nástroje
- Nástroje typu „hloupá opice“ (náhodné klikání nebo vstupy)
- Nástroje typu „polointeligentní opice“ – rozpozná nalezení chyby
- Nástroje typu „inteligentní opice“ – nástroj ví kde je a co dělá

Alfa testy
Interní testy ve vývojovém prostředí, které ale provádí někdo jiný než vývojový tým.

Beta testy
Zapojení externích osob do testování, např. potenciální uživatelé. Není zde jistota, že beta testeři otestují vše. Výhodou je ale např. ověření kompatibility s HW kombinacemi.

Strategie testování
Kolik osob, časový plán testů, stanovení metrik…

SLA – Service Level Agreement

Dohoda o poskytované službě, v kontextu provozu systémů se jedná o podmínky pozáručního servisu (cena servisu, rychlost reakce po nahlášení problému atd…)

Audit

Co je to audit?

- Ujištění o stavu (kvalitě) IT/ICT, které poskytuje třetí strana
- Názor, zda posuzovaná věc je nebo není v souladu se stanovenými kritérii
- Audit není opakovatelný – vždy jsou jiné podmínky

Typy auditu:

- Finanční
- Audit souladu (Compliance Audit) s normami
- Audit operační (výkonu) – nezávislé prověření fungování organizační entity

V ČR:

- Zákon 542/1992 Sb. o auditorech
- Od 1997 je ČR člen ISACA (IS Audit and Control Association)

Typy auditu:

- Založený na kontrolách
- Procesně orientovaný
- Substantivní (transakce)

Dále můžeme audit z hlediska času dělit na jednorázový a průběžný.

Typy auditu dle zaměření:

- Audit legálnosti
- Audit bezpečnosti
- Audit výkonnosti
- Forenzní (soudní) audit

Shrnutí hlavních bodů kapitoly:

- Black Box test – nemáme k dispozici zdrojový kód
- White Box test – máme zdrojový kód, lze využít debugger
- Testování okrajových podmínek
- Zátěžové testy
- Testy instalace a odinstalace
- Stress testy a long term test
- Vícejazyčná podpora
- Alfa a beta testy
- Automatizace testování
- SLA
- Audit

Použitá literatura:

1) PATTON, R. Testování softwaru. Vyd. 1. Praha: Computer Press, 2002, xiv, 313 s. Programování. ISBN 80-722-6636-5.
2) Testování software – portál [online]. Dostupné na www: <http://testovanisoftwaru.cz/>
3) SOMMERVILLE, I.: Softwarové inženýrství. Brno: ComputerPress, 2013. ISBN: 978-80-251-3826-7

Kontrolní otázky:

1) Co je to prototyp?
2) Jaké jsou základní dva směry prototypování?
3) Co je to mock-up?
4) Co je to drátěný model (wireframe)?
5) Proč je nutné věnovat vysokou pozornost testování software?
6) Jaký je rozdíl mezi Black Box a White Box testy?
7) Co je to dynamické testování?
8) Co se kontroluje při testech specifikace?
9) Jaké znáte základní typy testů?
10) Co je to alfa a beta testování?
11) Co je to SLA?
12) Co je to audit?
13) Typy auditu