2. Vývoj uživatelského rozhraní. Problém testování. Zásady tvorby dokumentace. Počítačová ergonomie. Práce v týmu.

Osnova

1. Vývoj uživatelského rozhraní
2. Problém testování
3. Zásady tvorby dokumentace
4. Počítačová ergonomie
5. Práce v týmu

Výklad

1. Vývoj uživatelského rozhraní

Návrh a vývoj UI má specifické problémy a tudíž je vhodné koncipovat a vyvíjet UI jako samostatný subsystém. Je potřeba přesvědčit management, že UI je skutečně důležité. Analýza 31 projektů v 1993 (Jakob Nielsen) ukázala, že vývoj UI spotřebuje od 4% do 15% celkových nákladů na projekt (ideál je cca 10%).

Kvalita UI ovlivňuje:

- celkovou spokojenost zákazníka (je to „obal systému“)
- složitost ovládání systému
- dobu zaškolování, rychlost a snadnost učení + ovládání
- náklady na provoz (doba zaškolování, počet chyb, rychlost práce se systémem)

Zásady návrhu UI:

- analýza a specifikace požadavků
- návrh UI
- realizace prototypů a jejich testování s uživateli
- integrace UI se zbytkem systému a testování UI společně s uživateli
- sledování vlastností UI za provozu a vylepšování vlastnosti UI

Vývoj UI je silně ovlivňován:

- psychologií, dovednostmi znalostmi budoucích uživatelů (začítečník preferuje snadnost a zapamatovatelnost, pokročilý rychlost)
- testování lze provést pouze tzv. *uživatelským testováním*, tzn. sledujeme lidi při práci
- vlastnosti a zkušenosti uživatelů se mění, je třeba počítat se začátečníky i pokročilými
- musíme myslet také na ergonomii

Rozhraní je třeba vyvíjet spolu s uživateli, kteří s ním budou později opravdu pracovat (např. skladník bude testovat UI pro sklad, ne že to otestuje za něj manažer). Dělají se prototypy, ty se otestují a podle výsledků se udělají další atd.

Při analýze, návrhu a testování UI se využívají následující techniky:

- *Pozorování uživatelů* a jimi prováděné úkony je jeden ze základních postupů.
- Dalším, známým, způsobem jsou *scénáře* neboli zaznamenávání ucelených postupů uživatele při práci;
- *Myšlení nahlas* je technika, při níž uživatel za přítomnosti autora systému nebo testera nahlas říká, co právě dělá a proč to dělá;
- *Heuristická evaluace* je metoda, kde se neformálně hodnotí předložené UI podle sepsaných doporučení (viz. např. Jakob Nielsen – konzistence, jasné a stručné instrukce a chybové hlášky, prevence chyb, ...)

Testování UI je potřeba provádět s budoucími uživateli. Test se provádí za přítomnosti vývojáře. Optimální počet je 3-6. Testování má následující etapy:

- Příprava
- Zahájení - je třeba zdůraznit že se testuje systém, nikoliv uživatel. Čím víc chyb se najde tím lépe. Zdůraznit, že vše je anonymní a dobrovolné. Požádat o *myšlení nahlas*.
- Provedení testů - atmosféra by měla být uvolněná. Nedávat najevo že uživatel je pomalý či že dělá chyby. Není dobré, aby na podřízeného dohlížel jeho šéf. Na začátku dát lehčí úkol, uživatel jej splní a bude se cítit dobře. Pokud toho má tester dost je třeba ukončit testování.
- Vyhodnocení testů - Požádat uživatele o zhodnocení, zaznamenat výsledky testů.

Při vyhodnocení se bere v úvahu např.:

- doba provedení určitého úkolu
- počet správně splněných úkolů, počet chyb (celkově, za určitou dobu)
- počet použitých příkazů
- kolikrát a na jak dlouho se uživatel zasekl
- frekvence použití nápovědy, manuálu, hovorů na support, ...
- počet frustrací/potěšení uživatele

2. Problém testování

Nově napsané programy vždy obsahují chyby. Je proto nezbytné programy otestovat s cílem nalezení chyb nebo ověření, že systém funguje tak, jak má. Pro každý případ chybné práce (selhání, failure) je třeba nalézt příčinu. Testování programů je tedy proces, ve kterém se zjišťují selhání, příčiny selhání se lokalizují, čímž se zjistí místa v programech a zprostředkovaně v dokumentech (defekty), které je třeba změnit. Opravený program se testuje dokud frekvence selhávání systému neklesne pod určitou mez. Etapa ladění je velmi pracná, pracnější než psaní programů a je považována za nejobtížnější ze všech etap vývoje SW.

Problémem testování je to, že většina úkolů, které si při testování klademe, patří mezi algoritmicky nerozhodnutelné problémy. Příkladem je požadavek aby se provedly všechny funkce systému. Dá se ukázat, že neexistuje program, který by pro jiný program ověřil zda neexistují instrukce které nemohou být nikdy provedeny (tzv. mrtvý kód). Takových problémů existuje mnoho a proto je testování tak náročné. Jakkoli náročným testováním nelze dokázat, že testovaná aplikace je zcela bez chyb. Cíle formálně dokázat, že v programu nejsou určité třídy chyb, se snaží dosáhnout disciplína, která se nazývá formální verifikace. Zjednodušeně řečeno je princip formální verifikace takový, že matematicky popíšeme vlastnosti, které chceme v softwaru ověřit (například živost, dosažitelnost stavů, apod.) a pomocí počítače se snažíme dokázat, zda daná vlastnost platí či ne.

[vsuvka]

Verifikace představuje kontrolu vůči specifikaci (dokument, podle kterého vývojáři programují) - **vyhovuje SW specifikaci**? **Validace** je kontrola vůči zadání od zadavatele (požadavku od klienta) – **splňuje SW požadavky uživatele**?

Vstupy pro testování začínají vznikat už v návrhu aplikace. Výsledkem návrhu je mimo jiné i vypracování testových případů. Testování využívá i výsledků oponentur všech etap vývoje a inspekcí kódu. Je potřeba pamatovat také na testování extrémních hodnot (nic, nepovolené znaky, prázdný soubor, …).

Organizační zabezpečení testů u velkých projektů obvykle zastřešuje speciální tým testerů. Tento tým může pracovat nezávisle, to je ale velmi drahé. Proto se doporučuje, aby testéři spolupracovali s vývojovým týmem.

Cílem testování je tedy dokázat, že je program nesprávný.

Typy testů

- částí (unit testy) – testují se samostatné kusy programu, často dělají programátoři
- integrační – shora/zdola, zapojeno více tříd/modulů, testuje se taky komunikace mezi jednotlivými částmi
- regresní – zopakování většiny testů, může být příliš náročné na uživatele a na provoz
- funkcí – testují se ucelené akce systému, neznáme vnitřní strukturu

- systému – v simulovaném provozu jako celek
- předávací – podle smlouvy
- test užíváním – zkušební provoz
- test simulací nebo prototypem

Vlastnosti testů:

- Testy by měly být reprodukovatelné.
- Testy by měly být deterministické, tj. měly by mít na začátku vždy stejné vstupní podmínky.
- Testy by měly být nezávislé, tj. nebýt ovlivněny ostatními testy.
- Testy by měly být levně opakovatelné.

Typy testerů

- programátor je současně tester – populární, málo účinné (vadí u kritických aplikací), lepší je varianta když si testují programátoři navzájem (nikdy netestuju to co jsem programoval)
- ,,samostatný tester“ white box
- „samostatný tester“ black box – nejlepší, nejúčinější, nejdražší

Fáze

- testování částí/modulů → testování při integraci → testování přo předání

Integrace

Po dokončení testů jednotlivých modulů IS dochází k propojení (integraci) jednotlivých částí do celku. IS by se neměl integrovat naráz, ale postupně. Vycházet by se mělo ze závislostního grafu (modul B je závislý na modulu A) a pomocí tohoto grafu postupně integrovat. Integraci můžeme rozdělit na integraci shora a integraci zdola. Integrace zdola začíná u modulů které nepotřebují jiné moduly a postupně integruje až se integrují již integrované množiny modulů. Je to poměrně dobrá metoda, vyžaduje však mnoho pomocných programů, které generují data pro prověřené moduly. Také se pozdě zkontrolují funkce systému.

Integrace shora začíná u modulů, které nejsou potřeba v jiných modulech a závislosti tohoto modulu jsou simulovány. Tyto simulované podřízené moduly se naprogramují později a připojí do vznikajícího systému. Výhodou je, že se testují konečné funkce (moduly na vysoké úrovni abstrakce) a testuje se dříve rozhraní. Nevýhodou je, že simulace nižších úrovní je často velmi obtížná.

Existují některé další integrační (kombinované metody). Metoda sendviče funguje tak, že se systém rozdělí a některé funkcionality se integrují shora a některé zdola. Například u OS se uživatelské moduly integrují zdola a společné služby shora.

Činnosti při testování

Na základě plánu testů se pro jednotlivé moduly SW specifikuje:

- Jaké testové případy se uvažují (vstupy, příkazy, očekávané reakce systému,

výstupy)

- Jaké testové procedury se předpokládají
- Popis testu (cíl testu, podmínky a způsob provedení)

Po provedení testů se vypracovávají zprávy o zjištěných nedostatcích (ty je dobré zaznamenávat do žurnálu testů). Na závěr se vypracuje souhrná zpráva o testech. Jedno selhání/nedostatek se popisuje následovně:

- Identifikátor testu
- Zkrácený popis problému (abstrakt)
- Podrobný popis problému
- Důsledky selhání pro plán testů (co je třeba opravit)

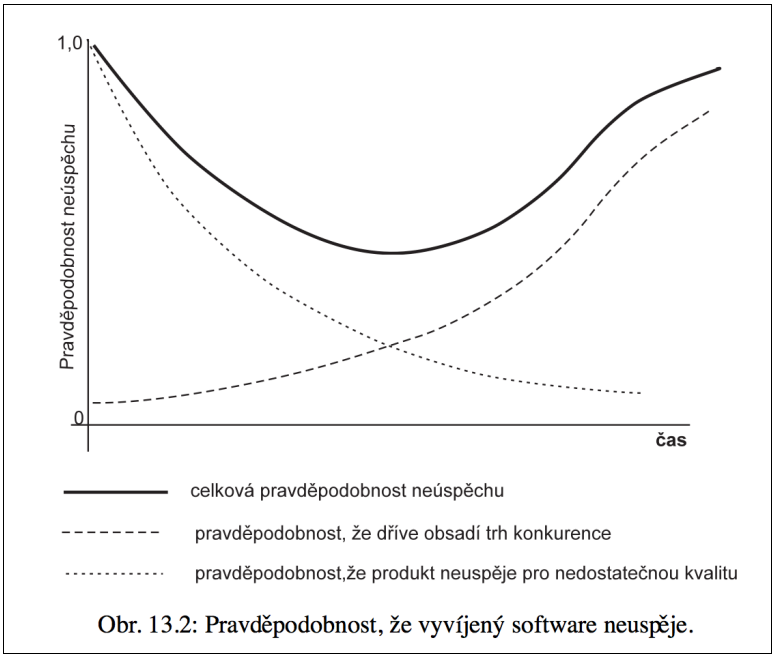
Souhrná zpráva obsahuje:

- Zprávy o předání položek k testování
- Žurnál testů
- Zprávy o chybách nebo jejich shrnutí
- Souhrná zpráva - co vše se testovalo, jaké jsou výsledky, zda je produkt přijat či ne

Ukončení testování

Nikdy neodhalíme všechny chyby. Některé bugy/defekty se budou opravovat za provozu (např. v Kenticu se reportované defekty opraví a každý týden se výdává hotfix). Při rozhodování o předání systému a ukončení testování exustují dvě rizika:

1. neodladěný systém → moc chyb (uděláme šmejd který nikdo nebude chtít používat) → neúspěch
2. „přeladěný systém“ → velké zpoždění (konkurence bude rychlější, nebo pokuta od zákazníka) → neúspěch



Obr. 13.2: Pravděpodobnost, že vyvíjený software neuspěje.

3. Zásady tvorby dokumentace

Dokumentace SW je velmi důležitá, platí to jak o technické, tak o administrativní a ekonomické dokumentaci. Je nezbytná pro velké projekty. Může vyžadovat více práce než kódování. Do jisté míry představuje paměť firmy.

Dokumentace má být:

- aktuální
- přehledná a srozumitelná
- umožňuje opravovat či modifikovat programy bez rizika zavlékání dalších chyb
- umožňuje snadno ověřit správnost implementace
- umožňuje sledovat průběh prací

SW dokumenty mají obsahovat informace o tom:

- jak systém používat
- jak systém instalovat a obsluhovat
- jak systém udržovat
- popis realizace
- testové procedury, data a hodnocení testů
- ostatní dokumenty

Dále existuje uživatelská dokumentace (obsahuje: návod k instalaci, úvod do systému, popis fcí, návod k použití, …) a dokumentace pro údržbu.

Kvalita dokumentace je stejně důležitá jako kvalita programu, přes nejrůznější doporučení a normy bývá ale na dosti nízké úrovni. Dokumentace bývá neúplná, zastaralá a především nepřehledná. Psaní dokumentace vyžaduje hodně úsilí. Napsaný text by měl být pečlivě čten autorem i jeho spolupracovníky, oponován a upravován. Některé zásady které pomůžou ke kvalitnější dokumentaci:

- psát kratší věty, snažíme se o maximální stručnost
- používáme-li často odkazy je vhodné nepoužívat pouze čísla kapitol ale i nadpis kapitoly v textu odkazu
- u obtížných míst naopak nešetřeme slovy i s příkladem
- je třeba zajistit jednoznačnost používaných termínů. Například pojem proces může mít v různém kontextu různý význam. Pak je vhodné doplnit slovník
- dokument by měl být dekomponován do kapitol nejvýše několik stránek dlouhých, odstavce ne delší než 10 vět
- jazykově správně
- dobře graficky upravený
- dobře hierarchicky rozdělen

Pro vedení dokumentace existují normy. Obecné zásady jsou v ISO 9000-3.

Existuje taky administrativní a ekonomická dokumentace (obsahují: podklady pro uzavření hospodářské smlouvy, podklady pro fakturaci a sledování nákladů).

Faktory kvality dokumentace:

- srozumitelnost – srozumitelná pro svou cílovou skupinu, která ji bude používat
- úplnost – obsahuje všechny potřebné informace
- testovatelnost – požadavky a cíle mají být formulovány tak, aby se daly ověřit (pouze 5% odpovědí má dobu odezvy větší než 5s)
- modifikovatelnost – lze dělat snadné změny a uchovávat jejich historii
- vystopovatelnost – u důležitých lze najít jejich důvod, proč bylo rozhodnotu tak či onak
- jednotná struktura – jednotné termíny apod.
- jednoznačnost – žádné nejasnosti či dvojí výklad
- použitelnost
- dostupnost – pro všechny, kteří ji potřebují
- aktuálnost – odpovídá aktuálnímu stavu

4. Počítačová ergonomie

IT systémy využívá stále větší část populace. To ovšem vytváří i negativní vlivy, které si musíme uvědomit a jako tvůrci softwaru se je pokusit odstranit a sami se jim vyvarovat.

Počítačové nemoci z povolání

Poškození zdraví v důsledku pracovní činnosti. Práce s PC je mentálně namáhavá a může ohrozit zdraví psychické i fyzické.

a) Objektivně zjistilené nemoci z práce na počítači

Dají se zjistit přístroji (např. otoky nebo záněty). Mezi objektivně zjistitelné nemoci z práce s počítači patří různé otoky a záněty. Zasažena bývá krční a bederní páteř a paže.

- RSI (repetitive strain injuries – nejčastější skupina objektivně zjistitelných nemocí způsobená opakovanou monotonní zátěží určitých pratií)
  - záněty v loketním kloubu - projevuje se bolestí lokte v určitých pozicích
  - otoky nervových pouzder v ruce a v předloktí - projevuje se brněním
  - otoky a záněty pouzder šlach a šlachových úponů - bolesti v paži vedoucím až k znehybnění
  - poškození kloubů a šlach v zápěstí
- Poškození krční páteře - při nevhodné poloze předloh a nevhodné poloze obrazovky dochází k přetěžování krční páteře a šíje.
- Poškození bederní páteře - bolesti v kříži. Nejčastěji je způsobeno nevhodným sezením a nekvalitní židlí.
- Nemoci očí, poškození zraku
- Problémy s dolními končetinami – bolesti, záněty, křečové žíly, riziko trombóz

Obrana proti RSI jsou přestávky, uvolňovací cviky. Proti poškození páteře se bráníme kvalitní sedačkou, dobře umístěným displejem a přestávkami.

b) Subjektivní nemoci

Vycházejí z neformálních vyšetření a z pocitů pacienta. Patří sem naříklad:

- zažívací potíže, nevolnost
- nespavost, noční úzkosti, noční děsy
- vyčerpanost, poruchy soustředění
- pocit psychické nepohody, špatná nálada

Tyto subjektivní problémy mohou být vyvolány nevhodným uspořádáním pracoviště - neodstíněným vysokofrekvenčním polem a vyzařováním obrazovky, nevhodná místnost, …).

c) Další potíže

- závislosti (hraní her, on-line gambling, …)
- psychosomatické problémy (spíše subjektivní, z nadměrné psychické zátěže – ubíjející rutina, nespokojenost, obecně stres)
  - syndrom vyhoření

Ergonomie

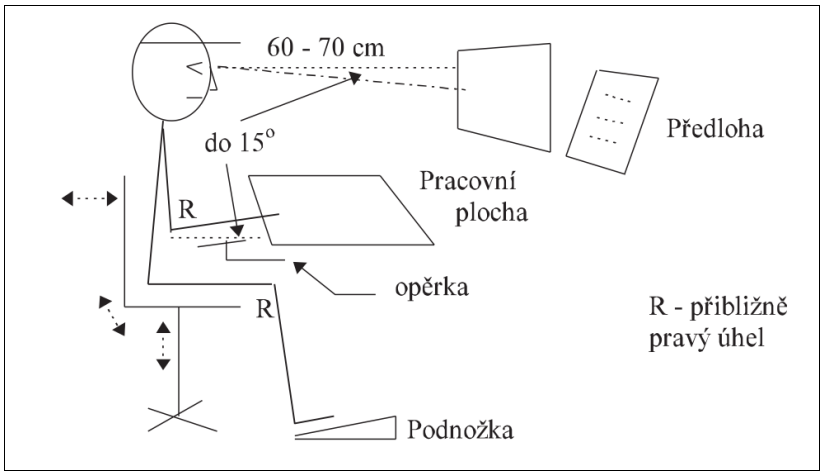
Je třeba chránit své zdraví a zdraví osob, které budou pracovat s námi navrženým softwarem před negativními vlivy.

a) Ergonomie práce s počítači

Přestávky (protáhnout se), nepracovat příliš dlouho (6h/den), střídat druhy zátěže (myš, klávesnice, touchpad, …), ...

b) Ergonomie pracovního místa

Správné vybavení (monitor, židle, stůl, …), správné sezení a vzdálenost monitoru, uklizený stůl, osvětlení, okna z boku, výhled (nejlépe do přírody), ...



c) Ergonomie pracoviště

Soukromí (není na to jednotný názor, důležitejší pro kvalifikovanou a kreativní práci), osvětlení, výhled (nejlépe do přírody), relax místnosti a aktivity (sleeping room, pinec, stolní fotbal, …), hezké a estetické prostředí (kytky, obrazy, …), dobré mikroklima, ...

d) Ergonomie softwaru

Pracovní zátěž lze snížit používáním kvalitního softwaru. Ten by měl být user friendly → nestresovat, příjemné barvy, nezáludný a srozumitelný, intuitivní, dobré je taky střídání činností (myš, klávesnice, vstát a jít k tiskárně, …), ...

5. Práce v týmu

Je zjevné, že větší úkoly nemůže realizovat jednotlivec ani malá skupina, proto je potřeba tým.

Výhody: sdílení znalostí, synergie (vzájemné pozitivní ovlivňování), dělba práce a rolí → lepší využití talentů, menší závislost na jednotlivcích, lépe se využije špičkových pracovínků (kterých je málo a jsou potřeba).

Nevýhody: náklady na budování a údržbu týmu (jsou k tomu potřeba specifické znalosti a dovednosti), administrativa, někdo může mít problém přijmout svou roli nebo se podřídit týmu, špatně organizovaný tým může mít horší efektivitu než jeho nejslabší člen.

Z průzkumů vyplývá, že je větší spokojenost s prací a vyšší produktivita u členů menších týmů s neformální organizací (agilní přístup). Centralizace je výhodná tam, kde je nutné rychlé rozhodování a u relativně jednoduchých a pracných činností, jako je shromažďování a předávání informací.

Optimální tedy je malá neformállní skupina se členy obou pohlaví a neautokratickým vedoucím. Bohužel existují úkoly, na které malý tým nestačí.

Dělení pracovníků dle jejich motivací

1. pracovníci orientovaní na úkol (*workoholici*, motivováni prací)
2. pracovníci orientovaní na spolupráci (*kamarádi*, motivováni interakcí a prací v týmu)
3. pracovníci orientovaní především na sebe sama (*sobci*, kariéristé – motivováni svou leností nebo kariérou)

(1) mohou být dobří vedoucí. Ale pokud jich je v týmu mnoho, tak mají tendenci k organizační nekázni, neradi se podřizují. (3) jsou dobří vedoucí, ale musí být motivováni prací. Mívají dostatek vůle a zároveň se nespokojují s nižším postavením. Z pohledu pohlaví jsou muži častěji orientovaní na práci, ženy pak na spolupráci. Snadno nahraditelní jsou manažeři a dělníci. Obtížně nahraditelní jsou vývojáři, vizionáři a specialisté.

Týmová loajalita je pojem vyjadřující stav, kdy členové týmu pojmou cíle projektu za své vlastní a brání ho před vnějším světem. Jedinci se identifikují s cíli a postupy týmu, jsou ochotni pro tým něco udělat a bojovat za něj. Přátelství týmu a hrdost na dosažené výsledky. Mezi nevýhody patří např. týmový šovinismus (nekritická obhajoba zájm týmu a jeho činností) a ponorková nemoc.

Při práci v týmu mohou nastávat různé problémy:

- Ve skupinách začnou vznikat (třeba na poradách) různé rozpory. Bývá to tehdy, pokud je mezi členy týmu příliš mnoho vůdců (1), (3). Pak je vhodné tým reorganizovat - rozdělit na více týmů

- Egoistické postoje. Je důležité, aby se chyby v programech a dokumentaci brali jako nutné zlo a nikdo se neobviňoval. Kód a celý software je dílem týmu, nikoliv spojení částí patřících jednotlivcům.

Při řešení nějakého rozhodnutí existují následující metody:

- postavení před hotovou věc
- rozhodnutí v klice (klika je malá skupina uvnitř týmu)
- dohoda menšiny
- dohoda – konsenzus - většiny
- všeobecný konsenzus
- zdánlivý konsenzus

Nejlepší rozhodnutí je všeobecný konsenzus a nejhorší zdánlivá dohoda.

Vedoucí týmu

Úspěch týmu silně závisí na vedoucím týmu. Vedoucí bývá přirozeně osoba, jehož rozhodnutí nebo doporučení bývají často respektována - nazývá se *vedoucí de facto*. Je výhodné aby byl zároveň *vedoucím de jure* neboli zvolen oficiálně organizací. Existují i týmy, kde mohou být tyto role odděleny a tito dva vedoucí musí respektovat své vzájemné role. Vedoucí de facto musí chápat potřebnost administrativy a vedoucí de jure musí ustoupit v technických otázkách. Ve větších týmech nemusí být vedoucí nutně odborně nejlepší (vedení/řízení vyžaduje specifické autonomní dovednosti).

Vedoucí by měl budit pocit, že je platným členem týmu. Má být schopen najít svého zástupce a spolupracovat s ním. Za hlavní psychologické rysy osobnosti patří:

- odborná a organizační kompetentnost
- zdravá sebedůvěra a tvrdohlavost, ne však arogance, psychologicky silná osobnost
- schopnost najít správné lidi, stanovit jim adekvátní úkoly a motivovat je
- být příkladem (pracovním i morálním), dovést uznat a napravit svoje chyby
- předvídavost, odhad rizik
- nepanikařit
- podržet tým při neúspěchu
- formulace cílů i způsobů řešení
- diplomatické schopnosti
- loajalita k podniku

Organizace SW týmů

Ke struktuře týmu je třeba zmínit princip minimaxu. Při tomto manažerském přístupu se rozhodnutí volí tak, aby maximální možná ztráta byla minimální. Pro příklad si představme špičkového programátora, který sám pracuje na důležitém projektu. Takový programátor může zastat mnoho průměrných programátorů (poměr 1:20 není neobvyklý), nicméně maximální možná ztráta právě tohoto zaměstnance by znamenala velký problém. Z

principu minimaxu tedy vyvodíme, že je lepší více průměrných programátorů. Ovšem i přístup minimaxu má své nevýhody - několik průměrných programátorů pravděpodobně stvoří pouze průměrný produkt, využití kvalitního programátora se dá parafrázovat větou „risk je zisk“.

Existuje několik základních typů organizace SW týmu:

- **horda** historicky nejstarší a práce se v ní rozprostře rovnoměrně mezi několik nebo mnoho programátorů a každý z nich řeší svůj díl od počáteční analýzy, přes algoritmizaci až po odladění. Ač není nutné tuto organizaci týmu zavrhovat, tak sebou nese velké úskalí, že z členů týmu se stanou osamělí vlci, pracující na vlastní pěst. S prací pak končí ve chvíli, kdy je přestane bavit - typicky dlouho před dokončením projektu.
- **demokratická skupina** funguje dobře pouze pokud je tvořena schopnými pracovníky. Nesmí být příliš velká a nesmí pracovat na ostrých termínech
- **tým šéfprogramátora** je výhodný, pokud je k dispozici několik vynikajících odborníků a poměrně mnoho relativně nezkušených pracovníků. V takovém týmu se dají řešit středně velké problémy
- **supertým** - opravdu velké problémy. Je to spojení více týmů, kde každý tým má určité kompetence.

Neformální role v týmu

Člen týmu by měl splňovat řadu podmínek. Při hodnocení čenů týmu je vhodné hodnotit nejen pracovní schopnosti a nadání, ale také zlozvyky. V psychologii práce se rozeznávají následující pracovní role:

Některé využitelné role:

- Hledač informací, inovátor: Hledá stále nové informace, snaží se o přesnost, dovede najít rozpory v návrzích. Vhodný jako oponent. Bývá slabší při realizaci úkolů do konce.
- Encyklopedista: Databáze zkušeností, hodnotí problém ve vztahu ke známému a podobnému: ”když jsme dělali x, museli jsme . . . “.
- Harmonizátor: Dovede uklidňovat napětí, dovede podporovat rozvoj dobrých vztahů. Dovede uklidňovat spory a hledat vhodné kompromisy.
- Koordinátor: Dává věci do širších souvislostí, objasňuje vztahy, shrnuje znalosti, dovede koordinovat aktivity, které spolu souvisejí, a také dovede takové aktivity nalézt.

Některé nežádoucí role:

- Agresor: Závidí, destruktivně nesouhlasí, bezohledně útočí ve věcech pracovních i osobních.
- Negativista: Cílem je zápor za každou cenu, zpochybňuje dohodnuté.
- Exhibicioista: Předvádí se, chlubí se, prosazuje se.

Pří řízení týmu je důležité detekovat potřebu rolí a zjistit, kteří členové týmu mohou hrát rozhodující role. Naopak je třeba eliminovat prostor pro rozvoj záporných rolí.

Ostatní aspekty práce v týmu

V týmech se mohou objevit podvědomá schémata chování, tzv. p*sychohry*, narušující práci v týmu. Nejčastěji se vyskytuje hra alkoholik. Nevýkonný pracovník je alkoholik kterého se snaží napravit vedoucí. Ve špatných zvyklostech ho udržuje utěšovatelka tím že mu říká „oni pro tebe nemají pochopení“ a kamarádi, kteří alkoholika přesvědčují, že o nic nejde. Další zajímavá hra je „beru vše“ - člen týmu, co přijímá stále nové úkoly a pak se vymlouvá na přetížení. Vedoucí týmu musí sledovat, zda členové týmu nejednají podle některého z výše uvedených schémat jednání.

Závěr

http://statnice.dqd.cz/mgr-szz:in-ins:2-ins

Vypracované otázky PA102 a PA105 (TIS I a TIS II)

Kniha Informační Systémy, prof. RNDr. Jaroslav Král, DrSc., 1998

- UI str. 215, resp. 217
- Ergonomie str. 49, resp. 47
- Testování str. 203, resp. 201
- Dokumentace str. 277, resp. 275
- Práce v týmu str. 121, resp. 119