Architektúra softvérových systémov

- prednášateľ: Ivan Polášek, D110
- kontakt: polasek@fiit.stuba.sk, ipo@gratex.com
- štúdium: inžinierske
- študijný odbor: softvérové inžinierstvo
- autor koncepcie: Prof. Ing. Pavol Návrat, PhD.

Zdroje prednášok

- Materiál a poznámky prof. Návrata - Úvod
- M.Shaw a D.Garlan: Software architecture (Carnegie Mellon University) – Architektonické štýly
- Buschmann (Siemens, Germany), Henney (UK), Schmidt (Vanderbilt University, USA): POSA 4
- Ostatní kolegovia..

Ostatní kolegovia..

Ľubor Šešera (Softec, STU): Internetové a intranetové systémy, aplikačná vrstva

Ľubor Šešera (Softec, STU): Internetové a intranetové systémy, vrstva dátových služieb.

Valentino Vranić (STU): Konfigurovateľnosť v modelovaní softvéru

Daniel Masar (Gratex): Skúsenosti s tvorbou frameworkov a aplikácií pre veľké J2EE aplikácie

Andrej Danko (SAP): Formalization of Functional Aspects in Business Software Globalization. Architektúra biznis aplikácií (SAP), on-demand/cloud aplikácií

Peter Girovský (IBM): Technológie pre J2EE architektury-frameworky

Milan Marenčík (MS): Technológie pre .NET architektúry/frameworky

Veľká slovenská encyklopédia "Encyklopedia Beliana"

- **architektúra softvérových systémov**:
  - architektúra softvérových systémov poskytuje spôsoby abstraktného opisu štruktúry softvérových systémov a princípy a pravidlá, ktorými sa riadi ich návrh a vývoj v čase.
  - architektúra jednotlivého softvérového systému alebo špeciálnej triedy softvérových systémov je tiež schéma, ktorá vyjadruje jeho základnú štrukturálnu organizáciu.
  - disciplína informatiky, zaoberajúca sa navrhovaním štruktúry softvérového systému a jeho zložiek, ktoré majú stanovené funkcie a vzájomné vzťahy;
  - vznik architektúry softvérových systémov ako disciplíny si vynútila veľká a neustále narastajúca zložitosť používaných softvérových systémov.
  - umožnilo ho poznanie, že mnohokrát sa navrhujú softvérové systémy podobné už jestvujúcim.
  - Medzi významné architektonické vzory (štýly) patria vzory pre systémy dátovodov a filtrov, systémy s tabuľovou organizáciou, distribuované systémy, interaktívne systémy, vrstvené systémy, interpretačné systémy a pod.

IEEE Software Engineering Standards, 1987, 1994.

Softvér: zbierka programov, postupov, pravidiel,

a s nimi združenej dokumentácie a údajov.

Sotvérové inžinierstvo:

systematický prístup k vývoju, prevádzke, údržbe a vyradeniu softvéru.

aplikovanie vedy a matematiky, pri ktorom sa prostredníctvom programov, postupov a s nimi združenej dokumentácie možnosti počítačového vybavenia stávajú užitočnými človeku.

Čo je to softvér?

- Počítačové programy a prislúchajúca dokumentácia ako požiadavky, návrhy a používateľské príručky
- Softvérové produkty môžu byť vyvíjané pre konkrétneho zákazníka alebo môžu byť vyvíjané pre trh.
- Softvérové produkty môžu byť
  - Generické – vyvíjané na to, aby sa predávali širokým skupinám – PC softvér, Excel, Word,...
  - Na mieru – vyvíjané pre zákazníka podľa špecifikácie
- Nový softvér môže byť vytvorený vyvíjaním nových programov, konfigurovaním generických systémov alebo znovupoužitím existujúceho softvéru

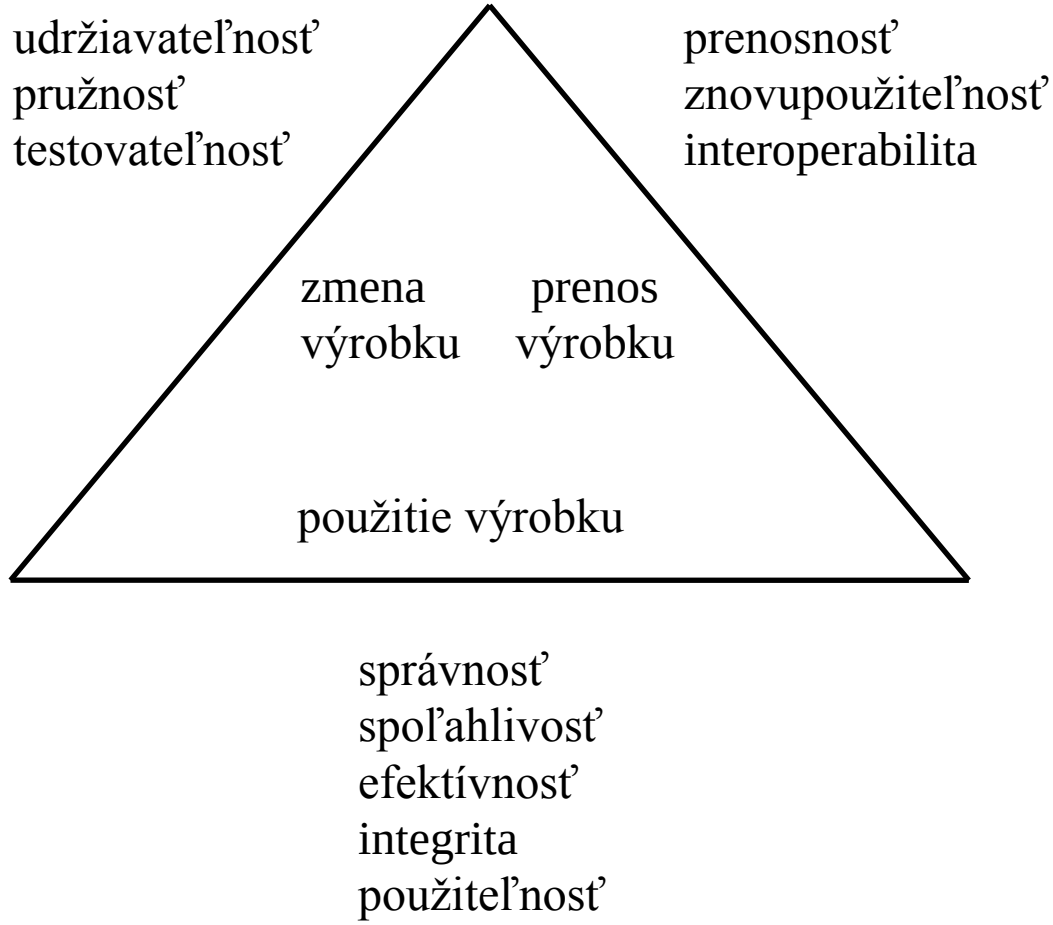
Softvérové súčiastky a súdržnosť

- **Softvérová súčiastka** je entita/prvok, ktorý implementuje jedinú logickú entitu alebo funkciu.
- **Miera súdržnosti** je miera, do akej súčiastka vnútorne spolu súvisí.
- **Viazanosť súčiastok**:
  - **Voľne** viazané - zmena jednej prevdepodobne neovplyvní zmenu ďalších. Decentralizácia stavu, vzájomná komunikácia pomocou parametrov alebo správami.
  - **Silne** viazané - spoločné premenné, odovzdávanie riadiacej informácie

Úrovne súdržnosti

náhodná	časti sú jednoducho dané dokopy
logické združenie	súčiastky vykonávajúce podobné funkcie sa združia
časové združenie	súčiastky volané súčasne sa združia
procedurálne združenie	súčiastky sa postupne navzájom volajú
komunikačná súdržnosť	súčiastky spracúvajúce ten istý vstup a vytvárajúce ten istý výstup
postupná súdržnosť	výstup jednej časti je vstupom ďalšej časti
funkcionálna súdržnosť	každá časť je nutná pre vykonanie jedinej funkcie
objektová súdržnosť	každá operácia poskytuje spôsob zmeny alebo prečítania atribútov objektu

vlastnosti softvéru



Švajčiarska kvalita (spoločnosť Laufen)

- Dodržiavanie švajčarskej inžinierskej kvality
- Dokonalá funkčnosť a dizajn všetkých výrobkov
- Najvyššie štandardy precíznosti, spoľahlivosti a dôkladnosti
- Prísne kontroly kvality
- Chybné výrobky sú v ktoromkoľvek štádiu vyradené, aj keď ide iba o drobné kazy
- Nielen praktickosť, ale aj niečo navyše vo výrobku aj v službách

Cena softvéru

- Cena softvéru často dominuje nad cenou počítačového systému. Ceny softvéru na PC sú často väčšie ako cena hardvéru
- Ceny za udržiavanie softvéru sú vyššie ako tie, ktoré dáme za vývoj. Pre systémy s dlhou životnosťou táto cena môže byť aj niekoľkokrát vyššia
- Softvérové inžinierstvo sa zaoberá vývojom, ktorý cenovo výhodný

cena, termíny, akosť

- cena - cena použitých zdrojov: ľudskej práce, softvéru, hardvéru, ďalších zdrojov
  - dominantný zdroj: ľudská práca (osobomesiace)
- termíny - skracovanie termínu uvedenia výrobku na trh
- akosť -
  - prevádzka (použitie) výrobku
  - prenos výrobku
  - zmena výrobku

-Softvérové inžinierstvo

- Ekonomiky všetkých národov sú závislé na softvéri
- Viac a viac systémov sa riadi/ovláda pomocou softvéru (bankový sektor)
- Softvérové inžinierstvo sa týka téorií, metód a nástrojov pre profesionálny vývoj softvéru

-Čo je softvérové inžinierstvo?

- Softvérové inžinierstvo je inžinierska disciplína, ktorá sa zaoberá všetkými aspektmi tvorby/výroby/produkcie softvéru
- Softvéroví inžinieri by mali používať systematický a organizovaný prístup k ich práci a používať prislúchajúce nástroje a techniky, ktoré závisia od riešeného problému, vývojových ohraničení a dostupných zdrojov

Aký je rozdiel medzi softvérovým inžinierstvom a informatikou?

- Informatika sa zaoberá teóriou a základmi; softvérové inžinierstvo sa zaoberá praktickými postupmi vývoja poskytujúceho použiteľný softvér
- Teórie informatiky sú stále nedostačujúce, aby mohli pôsobiť ako úplný fundament pre softvérové inžinierstvo – tak, ako je fyzika pre elektroinžinierstvo

Aký je rozdiel medzi softvérovým inžinierstvom a systémovým inžinierstvom?

- Systémové inžinierstvo sa zaoberá všetkými aspektmi vývoja počítačových systémov vrátane hardvéru, softvéru a procesným inžinierstvom.
- Softvérové inžinierstvo je časť procesu týkajúceho sa vývojom softvérovej infraštruktúry, riadenia, aplikácií a databáz v systéme
- Systémoví inžinieri sa zaoberajú systémovou špecifikáciou, architektonickým návrhom, integráciou a nasadením

Čo je softvérový proces?

- Skupina činností, ktorých cieľom je vývoj alebo evolúcia softvéru
- Generické činnosti vo všetkých softvérových procesoch sú
  - Špecifikácia – čo systém by mal robiť a jeho vývojové ohraničenia
  - Vývoj – produkcia softvérového systému
  - Validácia – skontrolovanie, či softvér je ten, čo zákazník potrebuje
  - Evolúcia – zmena softvéru podľa neskorších potrieb

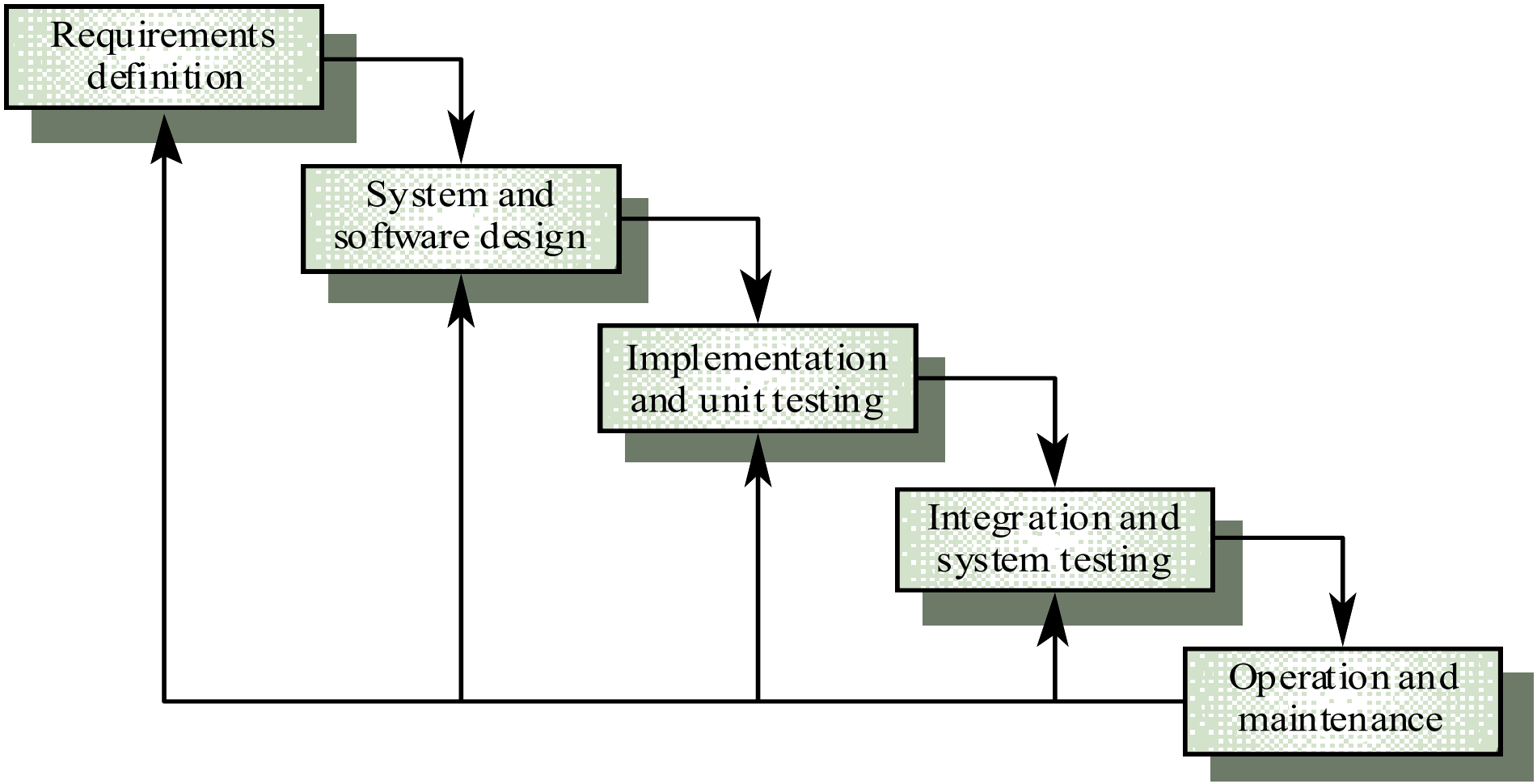
Čo je softvérový procesný model?

- Zjednodušená reprezentácia softvérového procesu, prezentovaného zo špecifickej perspektívy
- Príklady procesnej perspektívy sú:
  - Workflow perspektíva pracovných tokov – sekvencia činností
  - Data-Flow perspektíva toku údajov – tok informácií
  - Perspektíva rolí – kto robí, čo
- Generické procesné modely:
  - Vodopád
  - Iteratívny vývoj
  - Komponentový softvérový vývoj

Generické softvérové modely

- Vodopádový model
  - Oddelené fázy špecifikácie a vývoja
- Evolučný vývoj
  - Špecifikácia a vývoj sa prekrývajú
- Formálna transformácia
  - Matematický model sa implementuje pomocou formálnych transformácií

Vodopádový model



Vodopádový model

- Vodopádový model odráža inžinierske postupy, avšak fázy sa občas prekrývajú
- Proces sa stáva viditeľným odkedy sú vyrobené dokumenty po každej fáze
- Nevýhoda vodopádového modelu tkvie v ťažkosti niečo zmeniť po tom, ako proces začal

Evolučný vývoj

- Prieskumné prototypovanie
  - Cieľ je pracovať so zákazníkmi a vyvinúť finálny systém z počiatočnej vonkajšej špecifikácie
  - Malo by začať s dobre pochopenými požiadavkami
- Prototypovanie na zahodenie
  - Cieľ je pochopiť systémové požiadavky
  - Začína so zle pochopenými požiadavkami

Evolučný vývoj

- Problémy
  - Chýba viditeľnosť procesu
  - Systémy sú často ťažko viditeľné
  - Treba špeciálne schopnosti ako jazyky na rýchle prototypovanie a prostredia
- Využiteľnosť
  - Pre malé alebo stredne veľké interaktívne systémy
  - Pre malé časti veľkých systémov
  - Pre systémy s krátkou životnosťou

Manažment rizík

- Jedna zo základných úloh manažéra je minimalizovať riziko
- Riziko sa týka množstva a kvality dostupných informácií. Menej informácií, vyššie riziko
- Vysoko-rizikové činnosti viac-menej zapríčiňujú predlžovanie projektu a jeho predražovanie

Problémy

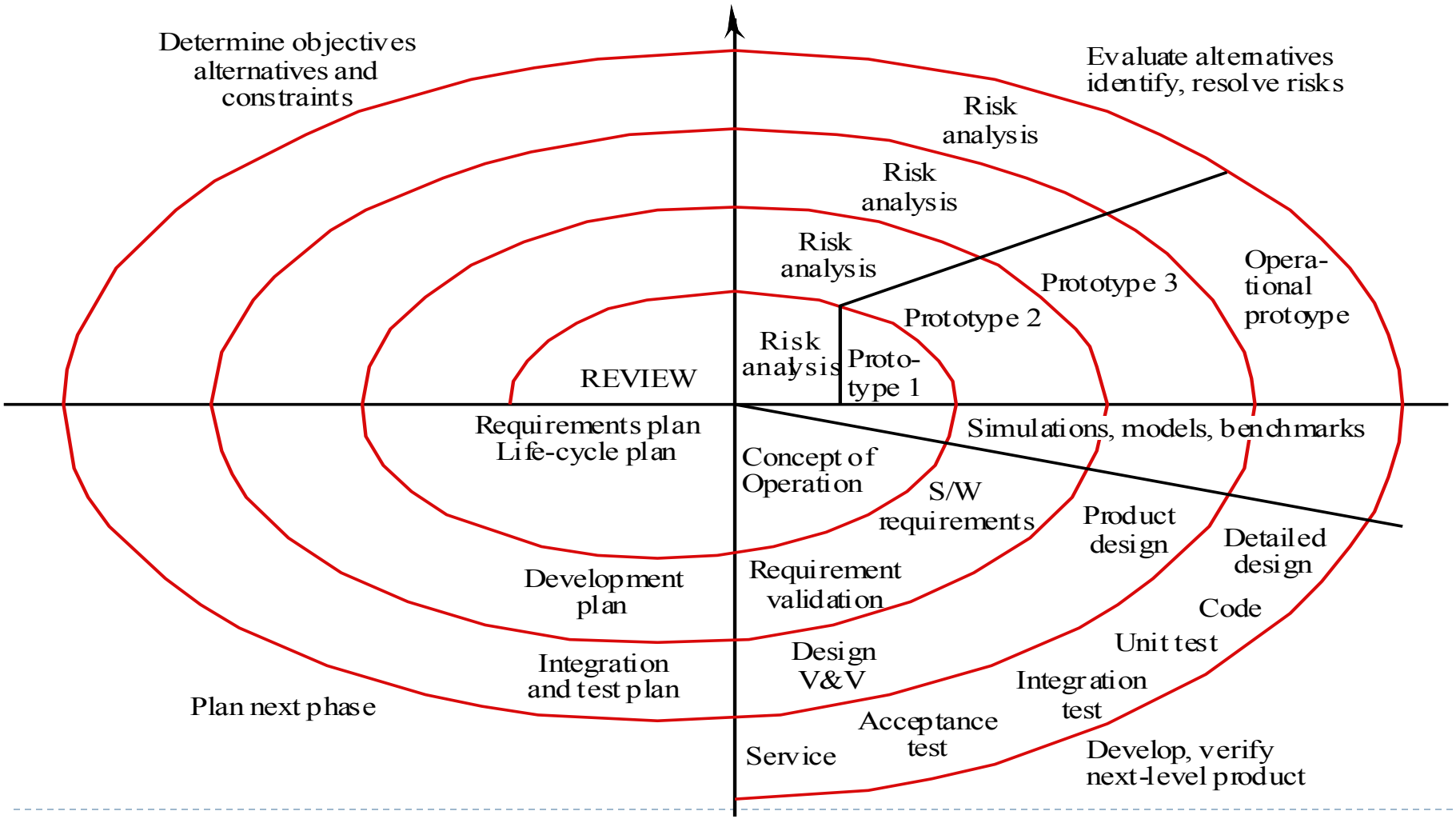
- Vodopádový
  - Vysoko rizikové pre nové systémy kvôli špecifikačným a návrhovým problémom. Malé riziko pre dobre pochopený vývoj použitím známej technológie
- Prototypovanie
  - Malé riziko pre nové aplikácie, pretože špecifikácia a program spolu držia krok. Vysoké riziko kvôli zlej viditeľnosti procesu
- Transformačný
  - Vysoko rizikové, pretože si vyžadujú veľmi pokročilé metódy

Hybridné procesné modely

Veľké systémy sa často robia z viacerých podsystémov, ktoré majú často rôzne charakteristiky. Rovnaký procesný model by sa nemal použiť pre všetky podsystémy, napr. môžeme použiť:

- Prototypovanie pre vysoko-rizikové aplikácie
- Vodopádový model pre dobre pochopený vývoj

-Boehmov špirálový model



Šablóna pre špirálovú otáčku

- *Ciele – ciele analýzy*
- *Ohraničenia – faktory, ktoré limitujú možnosti*
- *Alternatívy – alternatívne cesty dosiahnutia cieľa*
- *Riziká – možné riziká s identifikovanými alternatívami*
- *Vyriešenie rizík – stratégie, ktoré redukujú riziká*
- *Výsledky – Výsledky z týchto stratégií*
- *Plány – Ako naplánovať ďalšiu fázu*
- *Potvrdenie – rozhodnutia ako pokračovať*

-Príklad: katalóg súčiastok

- Ciele
  - Zabezpečiť softvérový katalóg súčiastok
- Ohraničenia
  - V rámci roka
  - Musí podporovať existujúce typy súčiastok
  - Cena menej ako 100 000 EUR
- Alternatívy
  - Kúpiť existujúci softvér na vyťahovanie informácií
  - Kúpiť databázu a vyvíjať katalóg použitím databázy
  - Vyvinúť katalóg na špeciálne použitie

-Príklad: katalóg súčiastok

- Riziká
  - Môže byť nemožné to zabezpečiť v rámci ohraničení
  - Špecifikácia katalógu je nejasná
- Vyriešenie rizík
  - Optimalizovať časové ohraničenia
  - Vyvinúť prototyp na pochopenie požiadaviek

-Príklad: katalóg súčiastok

- Výsledky
  - Systémy na sprístupňovania informácií (information retrieval) sú nepružné. Identifikované požiadavky sú nerealizovateľné.
  - Prototyp možno rozšíriť, aby mohol poskytovať úplnú funkcionalitu
  - Katalóg na špeciálne použitie je cenovo nevýhodný
- Plány
  - Vyvinúť katalóg obohatením prototypu a zlepšením používateľského rozhrania
- Rozhodnutie
  - Financovať ďalší 12-mesačný vývoj

-Pružnosť špirálového modelu

- Dobre pochopené systémy (nízke technické riziko) – vodopádový model. Fáza analýzy rizík je relatívne lacná
- Stabilné požiadavky a formálna špecifikácia. Kritická bezpečnosť - model formálnych transformácií
- Vysoké riziko – napr. neúplná špecifikácia – prototypovanie
- Hybridné modely sa prispôsobujú rôznym častiam projektu

-Viditeľnosť procesu

- Softvérové systémy sú nehmatateľné, čiže manažéri potrebujú dokumenty na odhadnutie pokroku
- Avšak toto môže spôsobovať problémy
  - Načasovanie termínov odovzdania priebežných výstupov sa nemusí zhodovať s časmi potrebnými na ukončenie príslušnej činnosti. Avšak vedenie projektu potrebuje pravidelné výstupy.
  - Potreba produkovať dokumenty ohraničuje iteráciu procesu
  - Na zrecenzovanie a potvrdenie dokumentov treba významne veľa času
- Vodopádový model zostáva naďalej najpoužívanejším modelom, založeným na výstupoch

Dokumenty vodopádového modelu

Activity	**Output documents**
Requirements analysis	Feasibility study, Outline requirements
Requirements definition	Requirements document
System specification	Functional specification, Acceptance test plan Draft user manual
Architectural design	Architectural specification, System test plan
Interface design	Interface specification, Integration test plan
Detailed design	Design specification, Unit test plan
Coding	Program code
Unit testing	Unit test report
Module testing	Module test report
Integration testing	Integration test report, Final user manual
System testing	System test report
Acceptance testing	Final system plus documentation

Viditeľnosť procesného modelu

Process model	Process visibility
Waterfall model	Good visibility, each activity produces some deliverable
Evolutionary development	Poor visibility, uneconomic to produce documents during rapid iteration
Formal transformations	Good visibility, documents must be produced from each phase for the process to continue
Reuse-oriented development	Moderate visibility, it may be artificial to produce documents describing reuse and reusable components.
Spiral model	Good visibility, each segment and each ring of the spiral should produce some document.

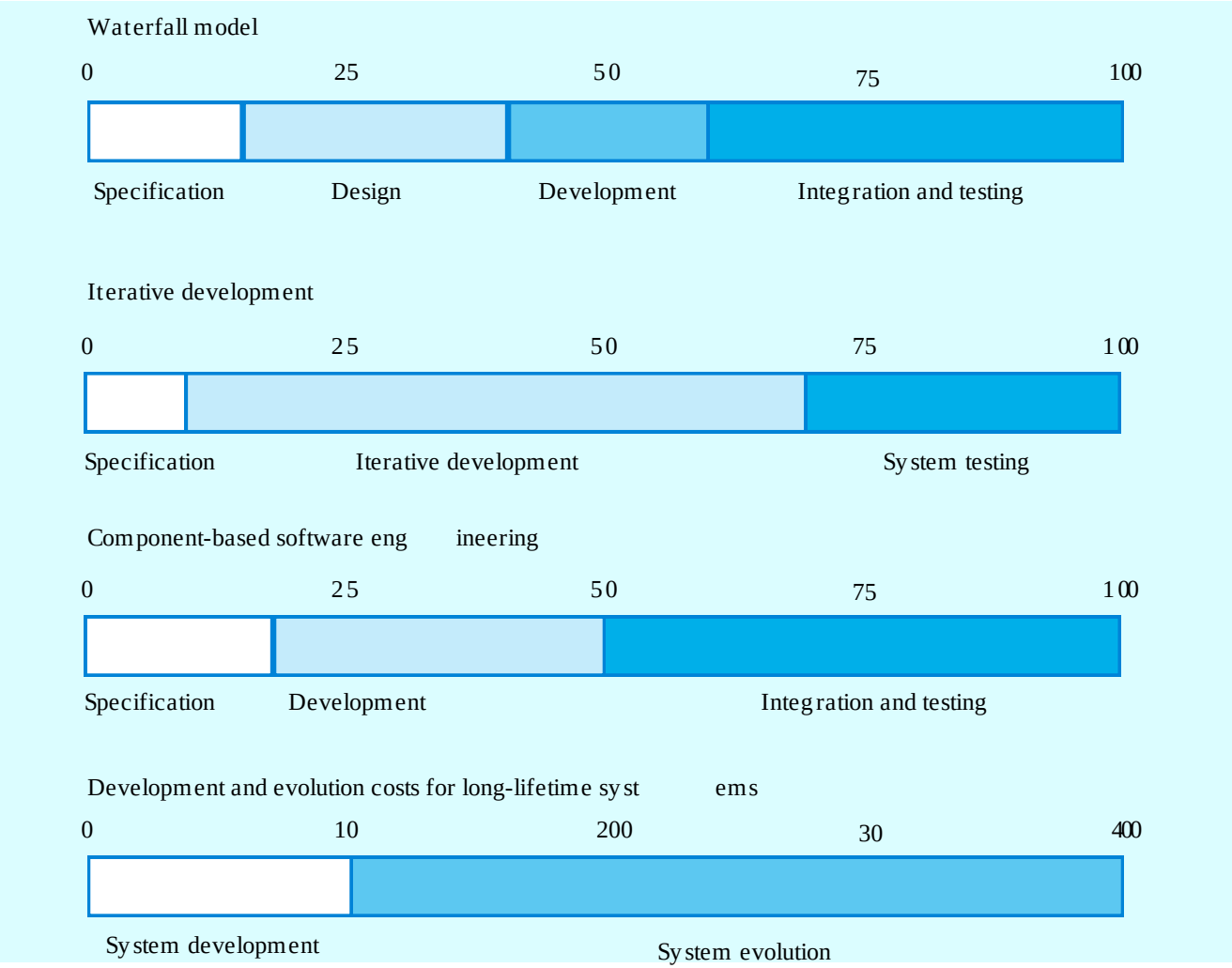
Vlastnosti návrhu softvéru

- Prispôsobivosť návrhu
  - Súčiastky sú voľne viazané
  - dobrá a aktuálna dokumentácia
  - zrejmá korešpondencia medzi úrovňami návrhu (napr. úroveň interakcií súčiastok, úroveň dekompozície súčiastok)
  - súčiastky sú veľmi súdržné

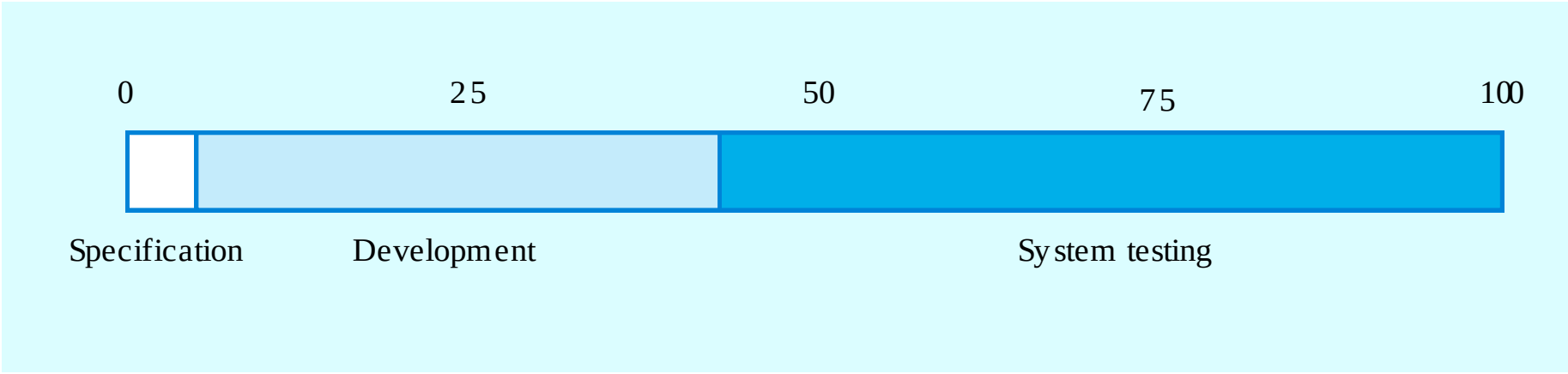
Aké sú ceny v softvérovom inžinierstve?

- Približne 60% z ceny je cena za vývoj, 40% sú výdavky za testovanie. Pre softvér na mieru, evolučné výdavky často prevyšujú vývojové výdavky
- Výdavky veľmi závisia od typu systému a požiadavky od systémových vlastností ako výkonnosť a spoľahlivosť
- Distribúcia výdavkov závisí od vývojového modelu, ktorý je použitý

Activity cost distribution



Výdavky za vývoj produktu



Aké metódy sú v softvérovom inžinierstve?

- Štruktúrované prístupy k sofvérovému vývoju, ktoré zahŕňajú systémové modely, notácie, pravidlá, rady k návrhu a procesne postupy
- Opisy modelov
  - Opisy grafických modelov, ktoré by mali byť vyprodukované
- Pravidlá
  - Ohraničenia aplikované na systémové modely
- Odporúčania
  - Rady na dobré návrhové postupy
- Procesné postupy
  - Aké činnosti sledovať

Čo je CASE (Computer-Aided Software Engineering)

- Softvérové systémy, ktoré majú zámer poskytovať automatizovanú podporu pre činnosti softvérového procesu
- CASE systémy sú často používané na podporu metód
- vrchný-CASE
  - Nástroje na podporu skorých procesných činností pri požiadavkach a návrhu
- dolný-CASE
  - Nástroje na podporu neskorých činností ako programovanie, ladenie a testovanie

Aké sú vlastnosti dobrého softvéru ?

- Softvér by mal poskytovať požadovanú funkcionalitu a výkon pre používateľa mal by byť udržiavateľný, spoľahlivý a akceptovateľný
- Udržiavateľnosť
  - Softvér sa musí vyvíjať, aby sa prispôsoboval meniacim sa potrebám
- Spoľahlivosť
  - Softvér musí byť dôveryhodný
- Efektívnosť
  - Softvér by nemal plytvať systémovými prostriedkami
- Prijateľnosť
  - Softvér musí prijateľný pre používateľov, pre ktorých bol vyvinutý. T,j, musí byť zrozumiteľný, použiteľný a zlučiteľný s inými systémami

Aké výzvy sú pri softvérovom inžinierstve?

- Rôznorodosť
  - Vyvinúť postupy na tvorbu softvéru, ktorý sa dokáže vyrovnať s rôznymi platformami a prostrediami, v ktorých sa vykonáva Dodanie
  - Vyvinúť postupy, vedúce k rýchlejšej dodávke softvéru
- Dôvera
  - Vyvinúť postupy, ktoré demonštrujú, že softvér je dôveryhodný pre používateľov.

Profesijné a etické zodpovednosti

- Softvérové inžinierstvo zahŕňa širšie zodpovednosti ako jednoduché aplikačné a technické zručnosti
- Softvéroví inžinieri sa musia správať čestným a zodpovedným spôsobom, ak majú byť rešpektovaní ako profesionáli
- Etické správanie je viac ako jednoduché dodržiavanie zákonov

Problémy profesijnej zodpovednosti

- Dôvera
  - Inžinieri by mali rešpektovať dôvernosť ich zamestnávateľov alebo klientov odhliadnuc od toho či bola ale nebola podpísaná akákoľvek formálna dohoda o dodržaní dôvernosti.
- Zodpovednosť
  - Inžinieri by nemali chybne prezentovať svoju úroveň kompetencie. Nemali by prijímať vedome prácu, ktorá je mimo ich kompetencie

Problémy profesijnej zodpovednosti

- Práva duševného vlastníctva
  - Inžinieri by mali byť vedomí miestnych zákonov vládnucich nad používaním intelektuálneho vlastníctva ako patentov, autorských práv, atď. Mali by dodržiavať fakt, že intelektuálne vlastníctvo zamestnávateľov a klientov sú chránené.
- Zneužitie počítača
  - Softvéroví inžinieri by nemali používať ich technické zručnosti na zneužívanie iných počítačov. Zneužitie počítača siaha od relatívne triviálneho až po extrémne vážne.

ACM/IEEE etický kódex

- Profesijné spoločnosti v USA spolupracovali za účelom vytvorenia etického kódexu
- Členovia týchto organizácií podpisujú pri vstupe etický kódex
- Kódex obsahuje 8 princípov vzťahujúcich sa na správanie a rozhodnutia urobené profesionálnymi softvérovými inžiniermi, vrátane praktikov, manažérov, vedúcich a rovnako učňov a študentov

Code of ethics - preamble

- Preamble
  - The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.
  - Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

Code of ethics - principles

- PUBLIC
  - Software engineers shall act consistently with the public interest.
- CLIENT AND EMPLOYER
  - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
- PRODUCT
  - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

Code of ethics - principles

- JUDGMENT
  - Software engineers shall maintain integrity and independence in their professional judgment.
- MANAGEMENT
  - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- PROFESSION
  - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

Code of ethics - principles

- COLLEAGUES
  - Software engineers shall be fair to and supportive of their colleagues.
- SELF
  - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Ethical dilemmas

- Disagreement in principle with the policies of senior management.
- Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- Participation in the development of military weapons systems or nuclear systems.