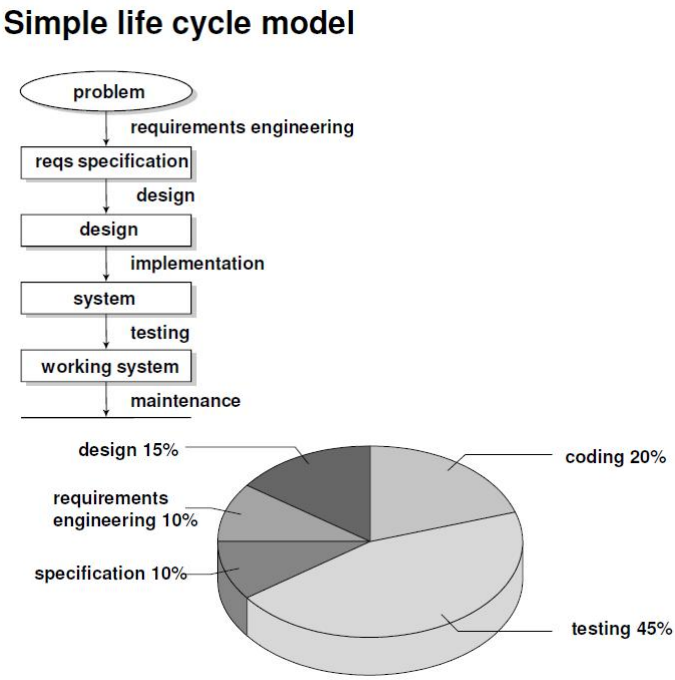
3 Proces vývoje software, jeho struktura a fáze vývoje, klasické a moderní agilní metodiky vývoje, řízení rizika. (A4B33SI)

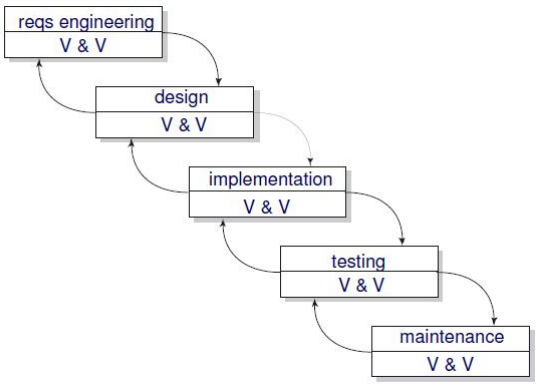
3.1 Proces vývoje software

1. **Requirement engeneering** (Stanovení požadavků) => Dokument 10%
   - jaké funkce, možnost rozšíření,...
   - získání nutné dokumentace, co musí umět (funkční/nefunkční požadavky, akceptační podmínky, náročnost na výkon...
2. **Design** (Návrh) 10%(spec) + 15%(design)
   - klade se důraz na to, co musí umět, ne jak
   - výsledkem je přesná specifikace
3. **Implementation (**Implementace) 20%
   - zaměření na jednotlivé komponenty
   - cílem je funkční SW
4. **Testing** (Testování) 45%
   - dělá systém to, co má?
   - mělo by se testovat v průběhu celého vývoje (až 45% času vývoje!)
   - validace: Děláme správný systém? (dle požadavků)
   - verifikace: Děláme systém správně? (bez chyb)
5. **Maintenance** (Údržba)
   - oprava chyb po předání zadavateli (neměly by být, ale jsou...) 21% + 4%(prevence)
   - úprava SW (25%), přidávání/úprava funkčnosti (25%)



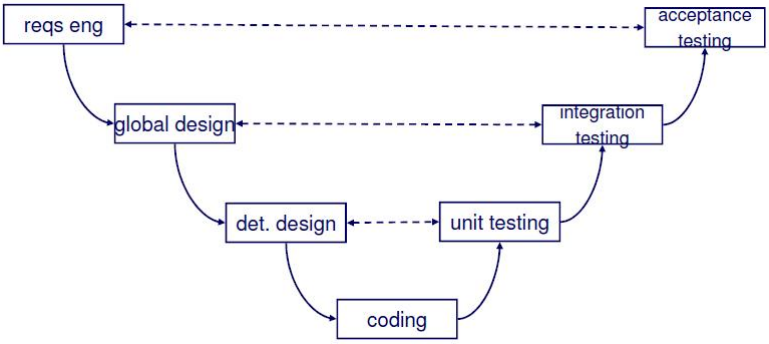
3.2 Metodiky vývoje

1. **Tradiční model**
   - vhodný na velké projekty
   - problémy: špatná (žádná) zpětná vazba, údržba nezahrnuje vývoj
2. **Waterfall model**
   - mezi každým krokem se provádí verifikace a validace a případně se vrací k dodělání
   - stále příliš nepružný (stále pevná dokumentace)



3. **V-model**

   - provádí se verifikace a validace vůči fázím
   - stále nepružný



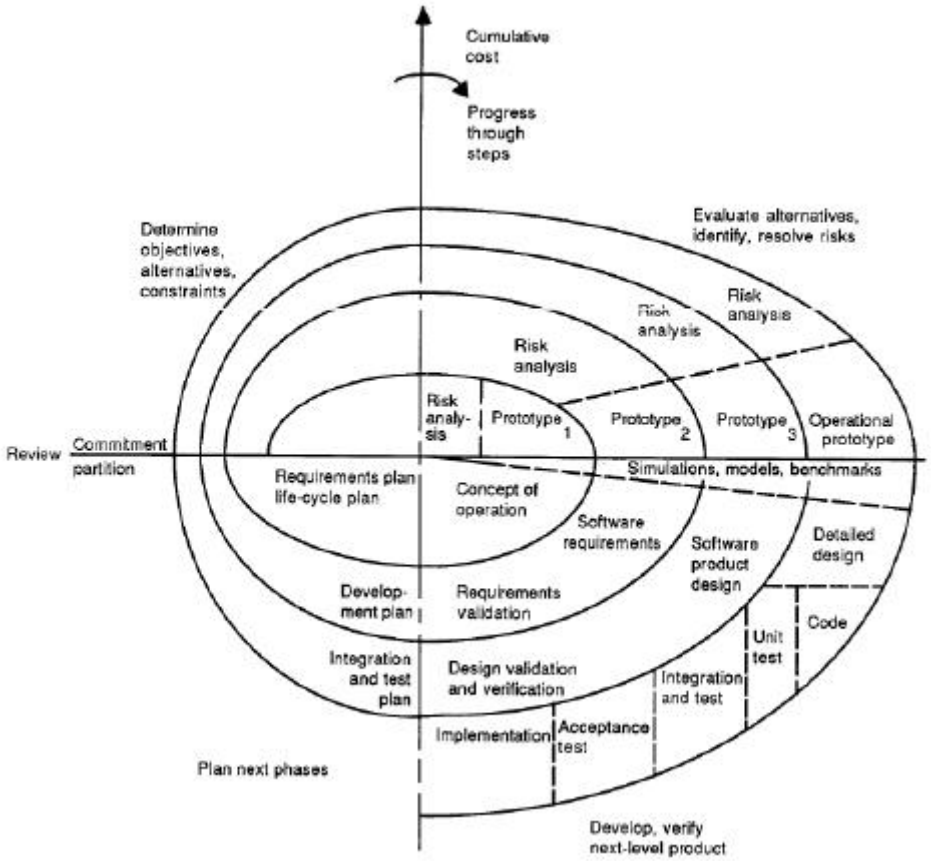
4. **Prototypy**

   - celkem levné, nemusí umět vše -> je ale nutné ukáza, jak bude fungovat
   - 2 typy:
     - *throwaway* - na jedno použití -> jako *waterfall*, až N-tý bude finální, jinak se zahodí a začne se znovu
     - *evolutionary* - vývojový -> až několikátý bude dodán
   - klady: jednodušší a rychlejší vývoj, rychleji se objeví chyby, jednodušší údržba (někdy)
   - zápory: více funkcí než je potřeba, méně výkonný, horší design, těžší údržba (někdy), nutná větší zkušenost vývojářů

5. **Incremental development**

   - dodáván po kouskách, v každém kousku waterfall model
   - uživatel více zatažen do vývoje (pro každý kousek)
   - nebude mít více funkcí než je nutné

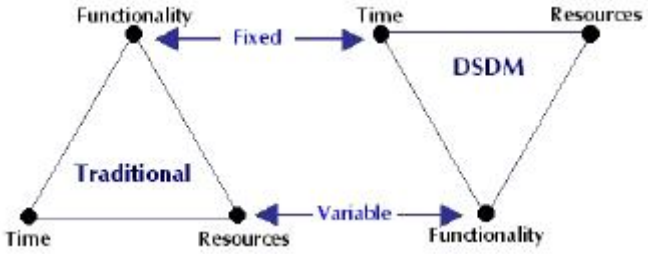
6. **Spiral model**



7. **RAD** (Rapid Application Development)
   - evoluční vývoj s časovými rámci (do kdy co musí být hotovo)
   - nejprve časový rámec -> snaha v něm udělat co nejvíce dle priorit
   - SWAT teams - skilled workers with advanced tools
8. **DSDM** (Dynamic System Development Method) - nejvíce v UK
   - pevně daný čas a prostředky -> odvíjí se funkčnost (u tradičních naopak)
   - nutná spolupráce uživatele
   - testování během vývoje
   - inkrementální vývoj, možnost vrátit změny
   - fáze:
     a) report a osnova o proveditelnosti, rychlý prototyp
     b) bussiness study - analýza, dodání architektury
     c) funkční model - časové rámce, inkrementační fáze

d) design a build iterace

e) implementace



9. **RUP** (Rational Unified Process)
   - doplňuje UML, používán na objektové systémy
   - fáze:
     a) Inception (začátek) - určení cílů, kritické use-cases, časový plán, odhad ceny
     b) Elaborate - založení architektury, včechny use-cases
     c) Construction - manufactory process (= prokládání dohromady)
     d) Transition - uvolnění pro uživatele, často několik releases
10. **MDA** (Model Driven Architecture)
    - nezávislý na architektuře -> výsledkem je PSM = platform specific model
    - 2 typy:
      – MODEL ———> CODE↻ (*údržba*)
      – (*údržba*) ↻MODEL ———> CODE

3.3 Řízení a rizika

- řízení:
  – *času* - počet člověko-hodin a plánování; těžko měřitelné; více lidí != méně času
  – *informací* - především dokumentace (Agilní méně, ale má lepší lidi); aktuální stav
  – organizace - týmy, lidé,...; organizace práce
  – kvality - žádná funkčnost navíc, přesně to, co má mít; nutná komunikace se stakeholdery
  – peněz - různé, hodně o osobách
- řízení pomocí CCB (Configuration Control Board)

  - drží aktuální stav

- méně lepších lidí, vyvážený tým