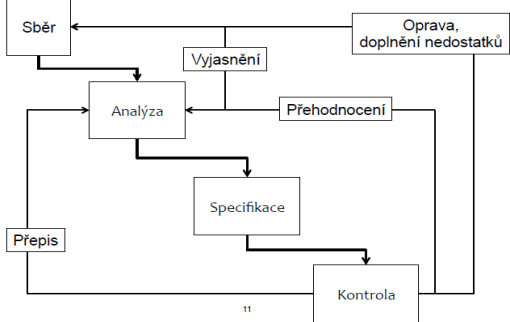


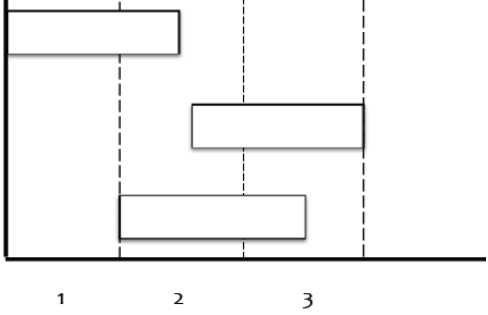
Projekt	„Časově ohraničené úsilí, směřující k vytvoření unikátního produktu nebo služby“. Čas a jedinečnost jeho výstupů, ho odlišují od procesu.
Projektový trojimperativ	Definice projektu je dána třemi veličinami, časem, kvalitou výstupu a zdroji. Změna jakékoliv z nich automaticky znamená, že musí dojít k odpovídající změně obou ostatních.
Charakteristiky projektu	<i>rozsah, čas, náklady, kvalita, zdroje, rizika projektu</i>
POS (Project Overview Statement)	Dokument, který by měl obsahovat: Problém a příležitosti Cíl projektu Obsah projektu Kritéria úspěchu Překážky rizika a předpoklady
FURPS	F (<i>functionality</i>) – funkčnost U (<i>usability</i>) – užitečnost R (<i>reliability</i>) – spolehlivost P (<i>performace</i>) – výkon S (<i>supportability</i>) – rozšiřitelnost

<p>F (functionality) – funkčnost</p>	<p>Zaměřuje se na hlavní funkcionality a schopnosti programu, zda software podporuje byznys proces a bezpečnost systému.</p>
<p>U (usability) – užitečnost</p>	<p>Pohled lidského faktoru; jakým celkovým dojmem působí aplikace, dokumentace a školicí materiály.</p>
<p>R (reliability) – spolehlivost</p>	<p>Jedná se o hodnocení četnosti a závažnost chyb, přesnosti zpracování vstupů a výstupů. Pro vyjádření spolehlivosti se často používá pojem MTFB (mean time between failures), což je střední doba mezi chybou nebo selháním. V této oblasti se také sledují možnosti obnovení provozu a zotavení výpadku.</p>
<p>P (performace) – výkon</p>	<p>Hodnocení celkové rychlosti odezev systému a zpracování klíčových byznys aktivit. Zároveň se sledují i technické parametry testovaného systému, např. vytížení zdrojů OS, zatížení síťového provozu, vytížení jednotlivých komponent systému.</p>
<p>S (supportability) – rozšiřitelnost</p>	<p>Rozšiřitelnosti aplikace v případě potřeby, možnosti údržby aplikace, její testovatelnosti. V této oblasti se taktéž hodnotí i přizpůsobitelnost a možnosti konfigurování.</p>

Kick off meeting	Úvodní schůzka, která zaručuje, že projekt začíná v kontrolované a organizované podobě.
Analýza požadavků - Sběr požadavků Koho se ptát?	Uživatel vs. Zákazník Všichni vs. Produktový "šampion" (expert na produkt, má zodpovědnost za produkt ale nerozhoduje o zásadních věcech)
Analýza požadavků - Sběr požadavků Jak se ptát?	Workshop/Interview Sledování uživatelů Stávající dokumentace Dotazníky Šampion přímo v týmu (agilní přístup)
Analýza požadavků - Požadavky	<i>Funkční požadavky:</i> podnikatelské požadavky, podnikatelská pravidla, uživatelská funkcionalita. <i>Nefunkční (obecné) požadavky:</i> kvalitativní parametry, požadavky na rozhraní, omezení.
Analýza požadavků - Obecné požadavky	<i>Zákaznické:</i> dostupnost, efektivita, flexibilita, integrita, kompatibilita, spolehlivost, odolnost, použitelnost (usability) <i>Programátorské:</i> udržovatelnost, přenositelnost, znovupoužitelnost, testovatelnost

<p>Analýza požadavků - Vlastnosti dobrého požadavku</p>	<p>Úplnost Jednoznačnost Správnost Priorita (must/should have, nice to have) Proveditelnost Ověřitelnost (akceptační test)</p>
<p>Analýza požadavků - Životní cyklus požadavku</p>	 <pre> graph TD Sběr --> Vyjasnění Vyjasnění --> Analýza Analýza --> Specifikace Specifikace --> Kontrola Kontrola --> Přehodnocení Přehodnocení --> Vyjasnění Přehodnocení --> Oprava["Oprava, doplnění nedostatků"] Oprava --> Vyjasnění Kontrola --> Přepis Přepis --> Analýza </pre>
<p>Model užití (Use case) - Typy zobrazení</p>	<p>Model: Aktér, Příklad užití, Hranice systému</p> <p>Textová reprezentace: Úspěšný a alternativní scénář(e), Počáteční a koncový stav</p>
<p>Model užití (Use case) - Co to je?</p>	<p>Use case je technika pro zdokumentování požadavku na nový systém, nebo změny na stávající systém.</p> <p>Poskytuje jeden nebo více scénářů, které zaznamenávají, jak by systém měl spolupracovat s koncovým uživatelem, nebo jiným systémem k dosažení konkrétních cílů.</p>
<p>Model užití (Use case) - Textový zápis</p>	<p>Podmínky (Preconditions) : podmínky nutné před UC.</p> <p>Průběh: průběh tohoto UC.</p> <p>Alternativní průběh: Co se stane, když něco není, jak by mělo.</p> <p>Podmínky (Postcondition): Co nového se po dokončení této UC může.</p>

Model procesů	Konceptuální (implementačně nezávislý) model všeho, co bude výsledek projektu dělat.
Model procesů - Značení	<p><i>Start/cíl:</i> černí puntík/bílý puntík s černou tečkou</p> <p><i>Aktivita:</i> obdelník</p> <p><i>Rozhodování:</i> kosočtverec</p> <p><i>Parelelní průběh:</i> tlustá svislá čára</p>
Model analitických tříd	<p>*..* = Registrovaný uživatel si může půjčit více aut. Použití vztahu *..* často značí neznalost problematiky – viz další slide.</p> <p>1..1 = Auto má pevně určené parkovací místo. Vždy se dá jednoznačně určit, které auto patří na které místo a obráceně.</p>
Plán aktivit - WBS	Jedná se o jednoduchou analytickou techniku, jejímž cílem je rozložit projekt na jednotlivé činnosti až do takové úrovně podrobnosti, aby k nim bylo možné přiřadit odpovědnosti, pracnost a časový horizont.
Milníky	Milník má nulovou dobu trvání a identifikuje kritické místo v plánu. Dosažení tohoto stavu musí být naplánováno a také musí být stanoven čas, ve kterém ho dosáhneme. Etapy projektu na sebe vzájemně navazují, některé mohou běžet současně a vzniká tak síťový graf, ve kterém jsou zobrazeny souběžnosti a závislosti.

<p>Odhad trvání aktivit/projektu</p>	<p>Podobnost s jinými aktivitami Historická data Rada experta Delphi technika (časový poker) Tříbodová technika</p>
<p>Harmonogram projektu</p>	<p>Je to časový plán projektu, který obsahuje posloupnost provedení jednotlivých činností, plánovaná data plnění těchto činností a klíčové milníky projektu. V praxi většinou vyjádřen formou Ganttova diagramu.</p>
<p>Gunttův model</p>	<div> <div>Stanovit cíle projektu</div> <div>Zajistit finance</div> <div>Napsat POS</div> </div> 
<p>RACI (matice zodpovědnosti)</p>	<p>Používá se pro přiřazení a zobrazení odpovědností jednotlivých osob v nějakém úkolu.</p> <p>R - <i>Responsible</i> (Odpovědný za vyhotovení) A - <i>Accountable</i> (Odpovědný za úkol) C - <i>Consulted</i> (Možný konzultat k úkolu) I - <i>Informed</i> (Kdo má být informován)</p>
<p>Techniky odhadu rozpočtu</p>	<p>Analogie Podle WBS Tří fázový odhad</p>

SWOT analýza	SWOT-analýza	Interní analýza	
		S: Silné stránky	W: Slabé stránky
	O: Příležitosti	S-O-Strategie: Vývoj nových metod, které jsou vhodné pro rozvoj silných stránek společnosti (projektu).	W-O-Strategie: Odstranění slabin pro vznik nových příležitostí.
	T: Hrozby	S-T-Strategie: Použití silných stránek pro zamezení hrozeb.	W-T-Strategie: Vývoj strategií, díky nimž je možné omezit hrozby, ohrožující naše slabé stránky.
ROI	<p>Označuje poměr vydělaných peněz k penězům investovaným. ROI tedy udává výnos v procentech z utracené částky.</p> <p>$ROI (\%) = \text{výnosy} / \text{investice} * 100$</p>		
Tabulka rizik	<p>Musí obsahovat:</p> <p><i>Popis rizika</i> <i>Pravděpodobnost rizika</i> <i>Dopad rizika na trojúhelník</i> <i>Závažnost rizika</i> <i>Protipatření</i></p> <p>Rizika jsou: <i>Technická, Spojená s řízením, Organizační, Externí</i></p>		
Jak udržet projekt ITOB?	<p>Pravidelné denní meetingy Práce ASAP Reportování ASAP Nebojte se Nehádejte – ptejte se Splňte, ale nepřekračujte požadavky S lidmi v týmu jednejte na rovinu</p> <p><i>ASAP (As Soon As Possible) = co nejdříve</i></p>		
EVA	<p>Je nástroj pro zjišťování a dodržování kvality při realizaci projektu. Díky ní lze dokázat, zda projekt po realizaci přináší či nepřináší peněžní hodnotu.</p>		

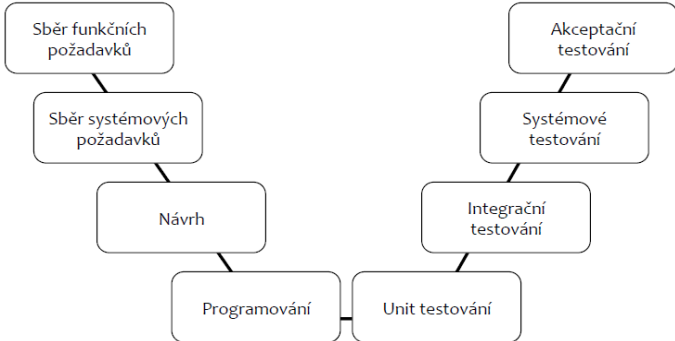
Reakce na skluz/prodražování	<p>Změna plánu Změnou zdrojů Domluvou se zákazníkem</p>
SCM	<p>Jedná se o celý balík programových prostředků, který umožňuje propojení jednotlivých článků dodavatelského řetězce (dodavatel - výrobce - distributor - prodejce - zákazník), a tím podstatně zlepšuje jeho schopnost reagovat na požadavky zákazníka.</p>
High cohesion – Vysoká soudržnost	<p>Tato zásada říká, že každý objekt, každá metoda musejí mít jediný (a navíc jednoduchý) úkol, za nějž jsou zodpovědní.</p>
Low coupling – Nízká provázanost	<p>Říká že každá entita si má při plnění úkolů vystačit sama a minimálně se obracet na jiné entity. Tuto zásadu není možné dodržet, ale lze se k ní přiblížit, jinými slovy – počet vazeb mezi třídami by měl být minimální.</p>
Demeterova pravidla	<p>Ta říkají, že uvnitř metody bychom měli pracovat pouze s:</p> <p><i>Sebou samým</i> <i>Parametrem metody</i> <i>Atributem this</i> <i>Elementem kolekce, které je v atributu this</i> <i>Objektem vytvořeným uvnitř metody</i></p> <p>Volání metod s nízkým provázáním může vypadat takto: <i>sales.GetPayment().GetAmount()</i></p>

<p>DRY (Don't repeat yourself)</p>	<p>Kód je jen na jednom místě Není jen problém týmů</p> <p><i>"Každý kus znalostí musí mít jediné, jednoznačné, zastoupení v autoritativní systému."</i></p>
<p>Princip objektového návrhu GRASP</p>	<p>Je sada doporučení, principů a vodítek, sloužících k vytvoření kvalitnějšího objektového návrhu. Princip GRASP se zaměřuje se na rozdělení zodpovědností jednotlivým třídám a objektům. Zodpovědností se rozumí podúloha, kterou má třída nebo skupina tříd řešit.</p>
<p>GRASP - Creator</p>	<p>B by měla vytvářet A pokud: <i>Instance B se skládají z A</i> <i>Instance B zaznamenávají A</i> <i>Instance B úzce využívají A</i> <i>Instance B mají informace nezbytné pro vznik A</i></p>
<p>GRASP - Polymorphism</p>	<p>Chování se mění na základě typu objektu.</p>
<p>GRASP - Controller</p>	<p>Třída označovaná jako controller (ovladač) je taková třída, která jako první zpracovává systémovou událost a zajišťuje její provedení.</p>

Class model UML	<p>Úrovně – <i>konceptuální, designová a implementační.</i></p> <p><i>Konceptuální</i> (analytický) model je vytvářen za účelem analýzy požadavků na software.</p> <p><i>Designový model</i> (model návrhu) rozšiřuje a zpřesňuje např. o viditelnosti atributů a metod, datové typy ap.</p>
Sekvenční diagram UML	Zobrazuje chování a spolupráci jednotlivých objektů v rámci jednoho případu užití.
Testování zahrnuje	<p>Plánování testu</p> <p>Provedení testu</p> <p>Porozumění výsledkům testu</p> <p>Nápravná opatření</p>
Testování provádí	<p>Uživatel</p> <p>Tvůrce</p> <p>Tester (<i>Technická způsobilost, Tvůrčí myšlení, Kritické myšlení, Praktické myšlení</i>)</p>
Proč testovat?	Čím dříve se chyby odhalí, tím nižší jsou náklady na jejich odstranění.

Cíl testování	<p>Ověřit, že SW dělá přesně to, co je uvedeno ve specifikaci a dále jak je schopen se vyrovnat s nestandardními stavy, jak reaguje na chybu uživatele nebo chybu v datech, selhání jiné SW nebo HW komponenty, jak se vypořádá se zátěží a nedostatkem systémových zdrojů, zda se dokáže zotavit po havárii, zda je odolný vůči útokům, jak funguje na různých HW a SW konfiguracích, atd.</p>
Black-box testing	<p>Je realizováno bez znalosti vnitřní datové a programové struktury.</p> <p>To znamená, že tester nemá k dispozici žádnou dokumentaci, binární ani zdrojové kódy. Tento způsob testování vyžaduje testovací scénáře, které jsou buď poskytnuty testerovi, nebo si je tester u některých typů testů sám vytváří.</p>
Black-box testing - Klady	<p>Tester nemusí být technik.</p> <p>Ověří rozpory mezi systémem a specifikací</p> <p>Test cases mohou být vytvořeny hned jak je hotová funkční specifikace</p>
Black-box testing - Zápory	<p>Jsou třeba rozsáhlá testovací data (vstupy)</p> <p>Je těžké odhalit všechny důležité vstupy v omezeném čase</p> <p>Vysoká pravděpodobnost změny testovaného blackboxu během testování</p>
Testování zátěže (Performance/Load)	<p>Hlavní myšlenka:</p> <ol style="list-style-type: none"> 1. Ověřit, že SW obstojí v běžném provozu 2. Zjistit, kdy to spadne <p><i>Má smysl ho provádět až po dokončení SW</i></p> <p><i>Začít testovat na prázdné DB – co je pomalé na prázdné, určitě nebude rychlé na plné</i></p>

<p>Reakce na výsledky zátěžových testů</p>	<p>Koupit lepší železo</p> <p>Úprava kódu</p> <p>Varovná zpráva</p> <p>Omezení počtu transakcí</p>
<p>White-box testing</p>	<p>Základní myšlenka: <i>To co je implementováno je implementováno správně?</i></p> <p>Je dostupná aplikace i kód</p>
<p>Unit testing</p>	<p>Testujeme nejmenší části programu</p> <p>Postupně postupujeme k větším celkům <i>metoda ---> třída</i></p> <p>Každá testována unit by neměla být závislá na ostatních</p>
<p>Nástroje Unit testing</p>	<p>JUnit - <i>NetBeans</i></p> <p>TestNG - <i>inspirace z Junit, vylepšení</i></p> <p>xUnit frameworks - <i>C++, C#, PHP, Ruby, Python</i></p> <p>dbUnit</p> <p>Code Coverage</p>
<p>Statické testování</p>	<p>Nevyžaduje běh softwaru, proto je možné s ním začít ještě před vytvořením prvního prototypu: I dokumentace je předmětem testování.</p>

Dinamické testování	<p>Vyžaduje existenci spustitelné verze softwaru a probíhá hlavně na základě poskytování různých vstupů a posuzování výstupů testovaného programu.</p>							
V - model								
Řízení projektů používající agilní metodiky	<p>Vize</p> <p>Týmová práce & spolupráce</p> <p>Jednoduchá pravidla</p> <p>Sdílení informací</p> <p>Lehké řízení</p> <p>Učení se</p> <p><i>Spokojený zákazník i programátor</i></p> <p><i>Za krátkou dobu, dodat kvalitní SW</i></p>							
Co Agilní přístup potřebuje?	<p>Zkušené programátory</p> <p>Někoho s předchozí zkušeností s Agile</p> <p>Komunikační schopnosti</p> <p><i>Doporučeno v malých týmech, Zkušení programátoři, nekritické systémy</i></p>							
Žádost o změnu	<table><tr><td>Název změny/ID</td></tr><tr><td>Žadatel</td></tr><tr><td>Stručný popis změny a její zdůvodnění</td></tr><tr><td>Podrobný popis změny</td></tr><tr><td>Dopady do stávajícího prostředí</td></tr><tr><td>Cena</td></tr><tr><td>Časová náročnost</td></tr></table>	Název změny/ID	Žadatel	Stručný popis změny a její zdůvodnění	Podrobný popis změny	Dopady do stávajícího prostředí	Cena	Časová náročnost
Název změny/ID								
Žadatel								
Stručný popis změny a její zdůvodnění								
Podrobný popis změny								
Dopady do stávajícího prostředí								
Cena								
Časová náročnost								

SCRUM	<p>Agilní metodika řízení IT pro 4 až 15 osob v jedné místnosti. Figurují v dvě skupiny tzv. Pigs a Chickens.</p> <p><i>PIGS</i>: osoby, které přímo souvisejí s vývojem aplikace.</p> <p><i>CHICKENS</i>: uživatelé produktu, manažeři, kteří přímo nezodpovídají za vývoj</p>
Dělení PIGS	<p><i>Product Owner</i> - osoba, která zodpovídá za priority, za to, co se bude v příštím sprintu implementovat a určuje implementační detaily.</p> <p><i>Scrum Master</i> - ten, kdo má programátory odstínit od okolního světa. Řídí vývojáře, ale zároveň se stará o to, aby jim fungovaly počítače, měli dostupný software, řeší spory apod.</p>
Dělení CHICKENS	<p><i>Stakeholders</i> - lidé od zákazníka, testeři, připomínky zvenčí.</p> <p><i>Managers</i> - osoby, které pomáhají nastavit prostředí, ale nejsou nic zbývajcího</p>
Fáze celého projektu v agilních metodikách	<p><i>Nultá iterace</i> - první krátká analýza a naprogramování nějaké základní činnosti, který se dá předvést </p> <p><i>Analýza změny</i> <i>implementace požadované vlastnosti</i> </p> <p><i>Předvedení klientovi</i> <i>Pokud není produkt hotov, zpět k analýze</i> <i>Pokud ano - údržba, rozvoj</i></p>
Stadia testování v projektu	<p><i>Zahajovací fáze</i>: testování dokumentace, bez funkčního kódu, ověřování případů užití.</p> <p><i>Produkční fáze</i>: testování nových funkcionalit, testování celého řešení k odevzdání.</p> <p><i>Dokončovací fáze a předávání</i>: testování celého produktu, testování instalačního balíku produktu a upgradů.</p>