Tworzenie gier na urządzenia mobilne

dr Przemysław Juszczuk

Katedra Inżynierii Wiedzy

Wykład 3

O czym dzisiaj?

- Etapy pracy;
- Od pomysłu;
- Przez tworzenie;
- Do realizacji.

Projekt w punktach

- Pomysł i wstępne założenia projektu;
- Prototyp;
- Dopracowanie założeń;
- Tworzenie elementów gry;
- Import elementów do silnika gry;
- Projektowanie poziomów;
- Testowanie;
- Przygotowanie wersji pod konkretną platformę.

Projektant gier

Osoba odpowiedzialna za opracowanie projektu gry, przygotowanie założeń, decyzje na etapie projektu. Jego najważniejszym zadaniem jest zadbanie o to, by produkt był grywalny.

Projektant gier cz.2

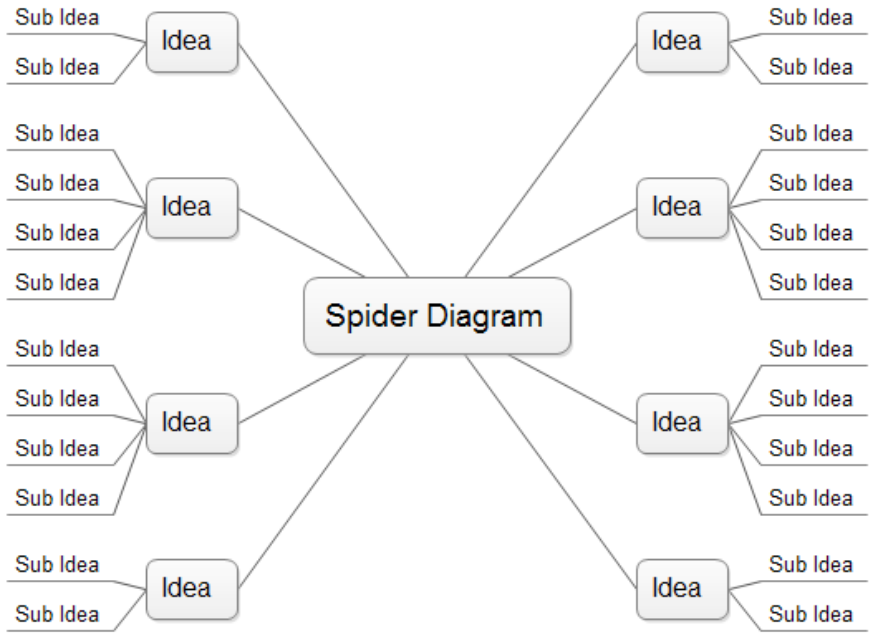
- To nie musi być informatyk/programista;
- "kreatywność, umiejętność pracy w zespole, komunikatywność, zarządzanie czasem, odporność na stres";
- umiejętność prototypowania (minimum to kartka papieru i ołówek);
- znajomość pakietów biurowych (+coś do grafiki);
- narzędzia dodatkowe, takie jak Tiled?

Wszystkiego nauczyć się nie da

- Lead Designer;
- Level Designer;
- Writer;
- UI Designer;
- Character Designer;
- Audio Designer;
- World Designer.

Lead Designer

- odpowiedzialność za projekt/projekty;
- trzeba wyłapać, na jakim etapie jest projekt - co się zmieniło, gdzie są problemy;
- tworzenie dokumentacji;
- oczywiście potrzebna kreatywność;
- spider diagrams?



Rysunek: Przykład Spider diagram (źródło: edrawsoft.com)

Writer

- co powinien robić? - kreatywne pisanie, opracowanie tzw. main plot, historii świata, fabuły, przygotowanie dialogów;
- co często robi - ogólny szkic historii, która schodzi na dalszy plan;
- czasami nawet nie ma takiej osoby w zespole - gry FPP ze szczątkową fabułą, albo zupełnie bez fabuły.
- na drugim biegunie mamy rozbudowane linie dialogowe i szczegółowo opisane uniwersum (Planescape, Baldur's gate);
- czasami wystarczy rozwinąć uniwersum o nowe elementy, ale wtedy uważamy, żeby wszystko było zgodne;
- przykłady bez spojlerów? - wiedźmin książka a wiedźmin gra; Śródziemie a gry.

Level Designer

Zaczynamy od koncept artów, szkicy, modeli, a dalej;

- ogólna budowa terenu - pokoje, wzgórza, teren działań gracza;
- planowanie rozmieszczenia zasobów na mapie (RTS - surowce), (RPG - questy);
- podział na mniejsze lokacje, określenie wyjścia z danego terenu;
- warunki pogodowe;
- kluczowe punkty: rozmieszczenie gracza, rozmieszczenie przeciwnika, interaktywne obszary.

Level design toolkit

- Infinity Engine - 2D RPG + AD & D, czyli Baldur's Gate, Icewind Dale, Planescape: Torment;
- Aurora Engine - stosowane przez CD Projekt RED przy wiedźminie.
- GemRB - czyli Inifinity Engine w wersji Open-sorce.

Rysunek: Baldur's Gate 2 (źródło: gemrb.org official site)

Etap 1 - pomysł

- określenie gatunku - RPG, RTS, FPP, TPP, logiczna, symulator, zręcznościowa, przygodowa;
- podgatunek? hack'n'slash, symulator samolotowy, platformówka 2D, clicker, idle?
- grupa docelowa - single/multi/MMO?
- platforma docelowa - Android, Windows Phone OS, BlackBerry? - np. łatwiej wybić się na BlackBerry, ale większe wsparcie zapewnia Android;
- tytuł - na początek wystarczy roboczy - ułatwi komunikację w zespole (zwłaszcza tym większym), a także przyda się w przypadku możliwej wczesnej bety i promocji w mediach społecznościowych ułatwi (tutaj już bardziej tytuł ostateczny);
- ad. tytuł - musi przyciągnąć gracza - krótki, łatwy do wyszukania tytuł - a może kierować się google trends? Sprawdzamy, jakie frazy są aktualnie popularne itp.

Etap 1 - pomysł (otoczka fabularna)

- ogólny zarys fabuły (nawet w przypadku idle, czy clickera);
- krótka charakterystyka bohaterów, przeciwnika/przeciwników;
- sceneria - lokacje, zarys i ich opis.

Jeszcze na etapie pomysłu należy określić grupę docelową:

- w grach dla dzieci unikamy zombie i lochów;
- w grach dla starszego odbiorcy unikamy jednorożców itp. (no chyba, że jest to związane z promocją - np. diablo 3);
- niższa grupa docelowa = prostsza, łatwiejsza w odbiorze fabuła.

Model biznesowy

- gra darmowa z reklamami;
- gra darmowa z płatnymi dodatkami;
- płatna gra z darmowymi patchami i dodatkami;
- płatna gra z płatnymi patchami i dodatkami (źle);
- płatna gra z płatnymi teksturami w HD (bardzo źle);
- popularne w ostatnim czasie mikropłatności?
- gra w pełni darmowa bez reklam - sposób na wyrobienie marki;
- nawet świetny pomysł i sensowny model ale duża liczba bugów pogrąży nie tylko grę ale zapewne też developera.

Etap 2 - Game Design Document GDD - po co?

- zapisujemy pomysł;
- mamy porządek w dokumentach;
- mamy listę TODO;
- wiemy, jakie zasoby posiadamy (czas, wiedza, zasoby ludzkie);
- funkcjonalności (must be oraz opcjonalne);
- jasny wykaz luk, które należy wypełnić/dopracować pomysły i funkcjonalności;
- pod koniec tworzenia projektu mamy coś na kształt dokumentacji;
- mamy wykaz założeń początkowych, które możemy porównać z projektem końcowym.

GDD

- Koncepcja - główny pomysł, grupa docelowa, platforma docelowa, szacowane koszty, szacowane zasoby, kluczowe idee projektu;
- Dokument (dokumenty) projektowe - GDD
- Dokumenty produkcyjne - koszty, lista zadań, specyfikacja techniczna, testy.

GDD - położenie

- Najlepiej - zarządzanie projektem i dokument w chmurze (Asana + google drive, Trello);
- Dobrze - dokument w chmurze - google drive (opcjonalnie coś z szyfrowaniem - tak dla bezpieczeństwa);
- Źle - dokument na dysku;
- Najgorzej - drukowany dokument podczas spotkań.

GDD - co zawiera

- historia;
- postacie;
- poziomy;
- sposób rozgrywki;
- assety;
- dźwięk i muzyka
- GUI.

Tworzenie prototypu gry

Prototyp umożliwia rozgrywkę, pokazuje podstawowe cechy produktu, pozwala na krótką rozgrywkę. Na tym etapie nie martwimy się optymalizacją, ładnymi assetami, dźwiękami, muzyką, strukturą poziomu, płynnością animacji itp.

- błędy? - mamy czas na sprawdzenie, czy są jakieś istotne błędy i możemy od razu reagować;
- OSZACOWANIE nakładu pracy - sprawdzenie, czy mamy wystarczające zasoby, czy nie przyda nam się dodatkowy grafik/programista;
- wizualizacja rozgrywki - możliwość przetestowania jej i oceny przez osoby spoza zespołu.

Po prototypie

- mamy listę sugestii i proponowanych zmian od "testerów";
- dodajemy swoją listę zmian, modyfikacji, nowych funkcjonalności;
- etap, na którym możemy przemyśleć projekt, dodać nowe elementy i ewentualnie porzucić projekt bez ponoszenia dużych kosztów (lepiej teraz, niż dużo później);
- teoretycznie w tym momencie nasza dokumentacja projektu powinna być kompletna - mamy listę wszystkich funkcjonalności, jakie mają się pojawić, wszystkie zmiany zostały zaakceptowane przez zespół itp.

Podział obowiązków i przyszłe zyski

- kto jest w zespole (kto jest liderem?) i jaką ma funkcję (patrz: projekt wykonaliśmy dokładnie po połowie);
- jakim sprzętem dysponujemy? - przydzielamy zadania w zależności od umiejętności (kluczowe) oraz zasobów (istotne);
- pracujemy w jednym miejscu? Czy raczej zdalnie? - w przypadku pierwszym zawsze łatwiej o metodykę scrum;
- oprogramowanie i dostępne biblioteki - darmowe licencje itp.;
- ustalenie deadline;
- realne oszacowanie budżetu - jakimi środkami dysponujemy.

Właściwy projekt

- prefabrykaty, ponowne wykorzystanie elementów, ale w miarę możliwości z wykluczeniem assetów dostępnych za darmo - możemy je wykorzystać jako bazę do własnych assetów (w zależności od licencji);
- dla dowolnej gry warto korzystać z materiałów referencyjnych: zdjęcia, filmy, szkice, mapy, istniejące już grafiki;
- narzędzia do automatycznej budowy modeli (w szczególności w przypadku postaci humanoidalnych);
- przy odpowiednio dużym budżecie - outsourcing, czyli zlecenie wykonania wybranych elementów, np. grafik osobom z zewnątrz bez konieczności zatrudniania ich na stałe.

Budowa poziomów

- budowa modułowa - wsparcie dla rozwiązań losowych, przy czym dużo zależy od rozgrywki. W przypadku gry labiryntowej z fabułą w tle jest to rozwiązanie sensowne;
- w przypadku gry fabularnej poziomy i świat gry muszą być przemyślane;
- gry typu clicker/idle - nacisk może być połączony na fabułę (rzadziej), lub na mechanikę (zdecydowanie częściej), a więc w jaki sposób ma przebiegać rozgrywka, żeby przyciągnąć gracza, następnie zaś, aby gracz wracał do produkcji;
- testowanie poziomów - najlepiej, aby poszczególne etapy, poziomy, fragmenty rozgrywki były testowane jednocześnie przez osoby zaangażowane w projekt, jak i przez osoby z zewnątrz. Przy czym na tym etapie ewentualna krytyka powinna dotyczyć już bezpośrednio samej struktury poziomów, a nie mechaniki rozgrywki.

Kodowanie i Testowanie

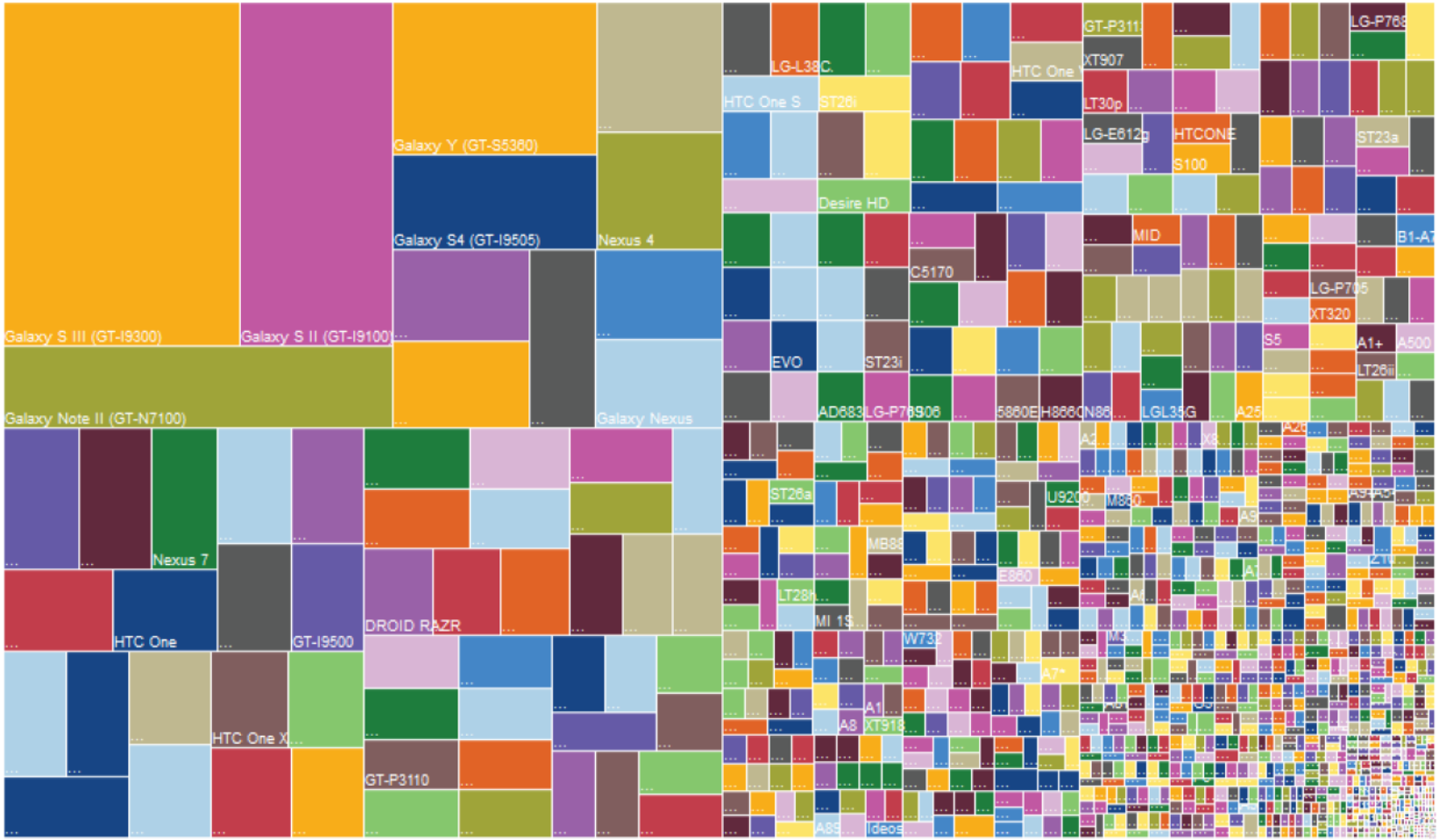
- programuj na kartce - wstępny zarys skryptu, kawałka kodu, klasy rozpisany na kartce w pseudokodzie pozwala od razu zweryfikować dany pomysł - na tym etapie określoną funkcjonalność rozbijamy na mniejsze zadania, które z kolei sa implementowane iteracyjnie;
- komentowanie kodu - przede wszystkim komentarze blokowe opisujące metody w skryptach - komentarze liniowe schodzą na dalszy plan;
- w przypadku tworzenia dokumentacji na bieżąco - każdy większy element opisujemy od razu;
- etykiedy TODO oraz FIXME.

Kodowanie i Testowanie

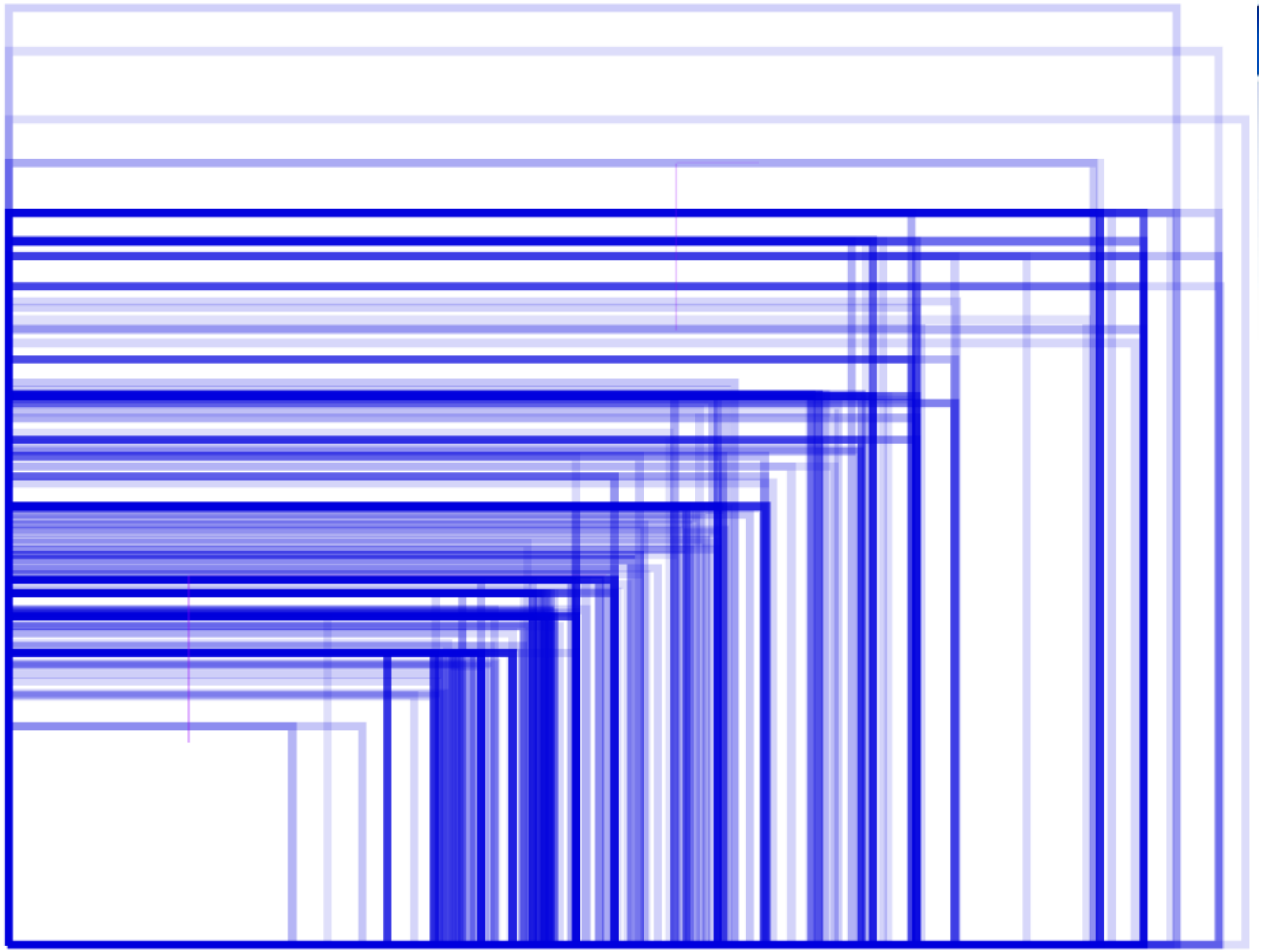
- testowanie to proces ciągły - nowe funkcjonalności sprawdzamy w miarę ich pojawiania się w grze;
- jeżeli mamy taką możliwość, to do testów włączamy osoby z zewnątrz;
- wersjonowanie - cokolwiek do kontroli wersji jest bardzo pożądane;
- testowanie = rejestrowanie błędów, czyli informacja o tym, w jakich warunkach pojawił się błąd oraz w miarę możliwości sprawdzenie, czy błąd jest powtarzalny. W drugim przypadku rozwiązanie problemu staje się zdecydowanie łatwiejsze.

Build, release and deploy

- wskazanie platformy docelowej - na początku jednej, w przyszłości, przy duzej liczbie odsłon można przenieść grę na inne platformy, lub rozwinąć system patchy i kolejnych wersji, dodatków;
- sprawdzenie wymagań stawianych grze przez platformę, na której ma nastąpić publikacja - w tym kosztów, jakie są wymagane przy publikacji gry.

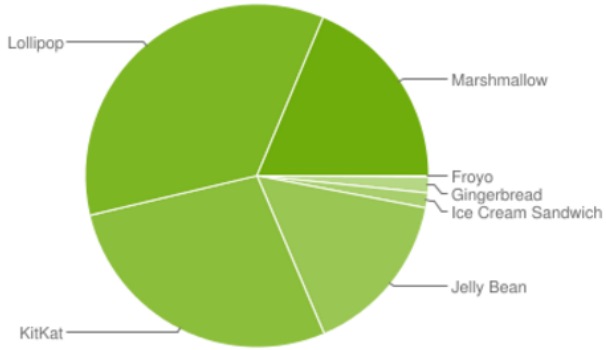


Rysunek: Urządzenia na rynku. Źródło: opensignal.com/reports/



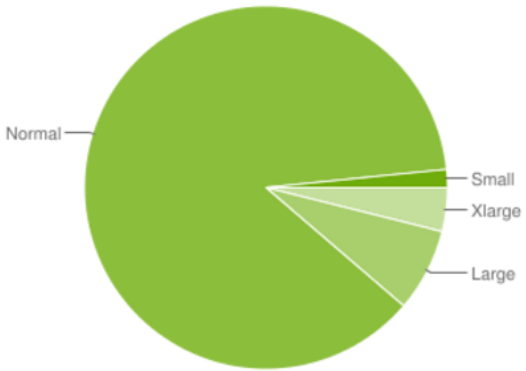
Rysunek: Wielkość ekranu dostępnych urządzeń. Źródło: opensignal.com/reports/

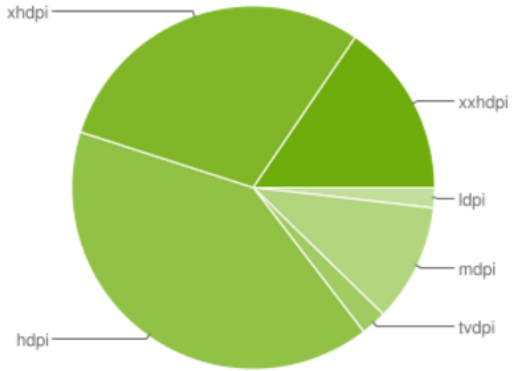
Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.4%
4.1.x	Jelly Bean	16	5.6%
4.2.x		17	7.7%
4.3		18	2.3%
4.4	KitKat	19	27.7%
5.0	Lollipop	21	13.1%
5.1		22	21.9%
6.0	Marshmallow	23	18.7%



Rysunek: Wersje systemu. Źródło: https://developer.android.com

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	1.6%						1.6%
Normal		3.5%	0.2%	39.5%	28.4%	15.5%	87.1%
Large	0.2%	4.1%	2.1%	0.5%	0.5%		7.4%
Xlarge		2.9%		0.3%	0.7%		3.9%
Total	1.8%	10.5%	2.3%	40.3%	29.6%	15.5%	





Rysunek: Wielkość ekranu. Źródło: https://developer.android.com

Gdzie możemy zarobić na aplikacji?

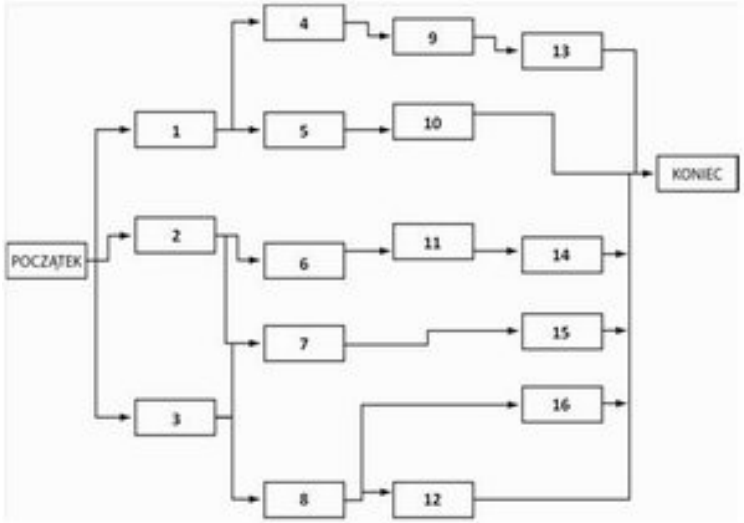
- Google Play (koszt wgrania 25 $) - łatwa instalacja, dużo potencjalnych odbiorców;
- Amazaon App Store (koszt 100 $);
- SlideMe - za darmo;
- GetJar - za darmo (w przypadku dwóch ostatnich społeczność, a co za tym idzie liczba pobrań jest mniejsza). Plusem jest mniejsza konkurencja niż w przypadku np. Google Play.

Diagram sieciowy / diagram PDM (ang. Precedence Diagram Method)

- ustalenie działań, które należy podjąć w celu zakończenia projektu;
- logiczna kolejność w czasie;
- podział na zadania konkretne, wobec których jesteśmy w stanie przewidzieć potencjalny czas zakończenia;
- jaka jest relacja pomiędzy poszczególnymi zadaniami;

Rysunek: Diagram sieciowy

- WBS – Work Breakdown Structure – kod w strukturze podziału prac (SPP);
- Czas – czas trwania zadania;
- ES – Earliest Start – Najwcześniejsze rozpoczęcie;
- TB – Total Buffer – Bufor całkowity;
- EF – Earliest Finish – Najwcześniejsze zakończenie;
- LS – Latest Start – Najpóźniejsze rozpoczęcie;
- FB – Free Buffer – Bufor swobodny;
- LF – Latest Finish – Najpóźniejsze zakończenie.



Rysunek: Diagram sieciowy – przykład. Young T.L. Skuteczne zarządzanie projektami. Helion, Gliwice 2010, 105-134

Bufor czasowy

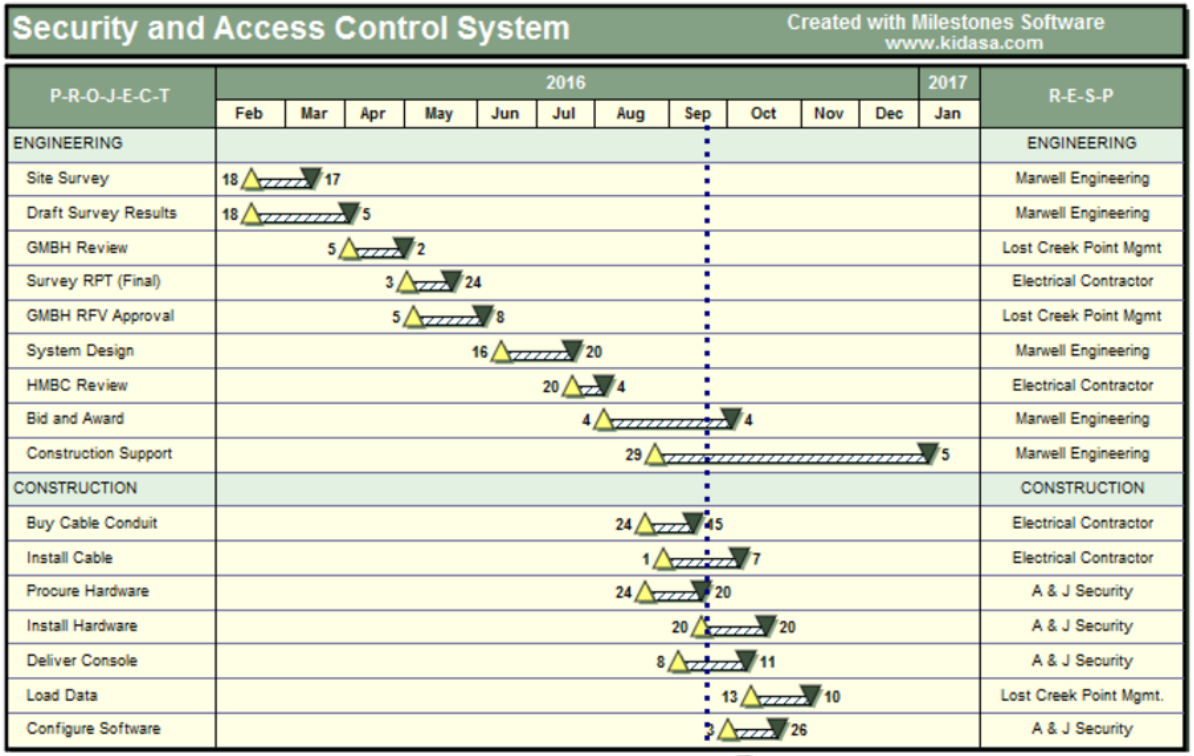
"(cytat, str. 85) Bufory czasowe to celowe rezerwy ustalone na potrzeby zabezpieczenia realizacji danej czynności. W praktyce na każdą czynność przeznaczony jest określony czas jej realizacji. Zwykle jest on nieco dłuższy niż wymagają tego rzeczywiste warunki techniczne. (...)" Marek Kasperek, Planowanie i organizacja projektów logistycznych. , Wydawnictwo Akademii Ekonomicznej, Katowice 2006

Diagram sieciowy 2

- każde zadanie powinno mieć chociaż minimalny bufor czasowy;
- w sytuacji kiedy istnieje zadanie o buforze równym 0 – jego opóźnienie automatycznie powoduje opóźnienie całego projektu;
- identyfikacja ścieżki krytycznej w diagramie sieciowym;
- ponadto każdy projekt posiada ścieżkę krytyczną biegnącą od pierwszego do ostatniego zadania;
- zadania w diagramie sieciowym mogą posłużyć do wygenerowania diagramu Gantta.

Diagram Gantta

- oś czasu – pozwalająca określić czas trwania całego projektu a także czas trwania poszczególnych zadań;
- oś pionowa – określa listę zadań, które są do zrealizowania w projekcie;
- kamienie milowe – zadania kluczowe dla projektu, których wykonanie pozwala na przejście do kolejnych etapów projektu;
- kamień milowy nie posiada kosztów – to wskazanie pewnego etapu projektu a nie konkretnego zadania;
- przykładem kamienia milowego jest dokument potwierdzający realizację pewnego etapu (np. zakończenie pierwszej z trzech iteracji tworzenia systemu);



Rysunek: Diagram Gantta – ganttchart.com

Trzy typy zadań

- kopanie rowu;
- kopanie studni;
- wycieczka górska.

Co na następnym wykładzie?

- model przyrostowy i programowanie zwinne;
- efektywność komunikacji;
- programowanie parami i programowanie ekstremalne;
- UML i scenariusze;
- nieco więcej o diagramie sieciowym i diagramie Gantta.

Dziękuję za uwagę