# Timeline

- Konferencja BUILD 2011 na której pokazano światu Win8 Developers Preview.

- Październik 2011 – kolejne releas’y elementów związanych z .NET 4.5

- Q1 2012 – Pogłoska - w tym czasie podobno Dell rozpoczyna tworzenie tabletu specjalnie pod win8.

- Czerwiec 2012 - Release Win8 – pogłoska, bazująca na modelu releasowym dla Win7.

Bardzo dużo niewiadomych, same informacje są pilnie strzeżone (do tego stopnia, że nawet np. są usuwane wpisy z LinkedIn).

# Async

Coraz więcej rdzeni, duża ilość usług dostępnych w chmurze - języki imperatywne nie nadążają za tymi wymaganiami. Dodatkowo opóźnienia przeszkadzają i interfejsy użytkownika robią się ciężkie, mało responsywne. Dodatkowym problemem jest jakość działania aplikacji silnie zależna od jakości połączenia internetowego (w krajach naszego pokroju jest to bardzo widoczny problem).

Opowiastka o dzwonieniu do konsultanta - UPC :P - synchroniczne blokujące oczekiwanie (czekanie to też akcja synchroniczna) jak automat mówi, że prosi o jeszcze trochę cierpliwości - blokada UI do czasu jak odezwie się konsultantka. Można tego uniknąć dwojako - włączam tryb głośnomówiący i idę robić co innego, jednocześnie czekając na to jak się ktoś odezwie i wtedy muszę przerwać dotychczasową pracę i zsynchronizować się z konsultantem (tworzę nowy wątek i synchronizuje się - MT). Niestety to ja zarządzam tym wątkiem i jestem odpowiedzialny za synchronizację, poświęcam swój czas i zasoby na to. Drugi sposób jest inny - na linii odzywa się automat, który mówi mi aby zostawił numer i zatwierdził go hashem - odezwą się do mnie za 20 minut. Mogę się w pełni zająć czymś innym, po 20 minutach telefon z konsultantem zadzwoni do mnie - to osoby do których dzwoniłem odpowiadają mi gdy jestem wolny, nie muszę się synchronizować dostarczam im tylko tzw. continuation (lub inaczej callback) - w naszym przypadku numer telefonu

**CZEKANIE JEST AKCJĄ SYNCHRONICZNĄ – innymi słowy: BLOKUJĄCĄ**

Drugie rozwiązanie ma następujące zalety - skalowalność (co jeśli będę chciał wykonać 15 telefonów?) i wydajność (nie poświęcam własnych zasobów na synchronizację). Jest też łatwiejsze do zarządzania z punktu widzenia architektury. Ktoś kto pisał wielowątkowy kod wie co to znaczy ból synchronizacji i sekcji krytycznych - tutaj za pomocą prostego patternu jesteśmy w stanie pozbyć się sporej części takich problemów - są one powiązane z następnym problemem przedstawionym na następnym slajdzie…

# Async part 2:

Responsywności oczekuje się od wszystkich aplikacji (to nie jest luksus tylko wymaganie). Taki trend można było najszybciej zauważyć w aplikacjach webowych, ponieważ tam praktycznie wszystko zależy od połączenia internetowego. Jednak coraz więcej aplikacji desktopowych komunikuje się z usługami dostępnymi w chmurze więc i tu ten problem staje się coraz wyraźniejszy. Dodatkowo Microsoft postanawia w związku z nadchodzącą premierą Windowsa 8 i udoskonalaniu swoich rozwiązań opartych o windows.live i Azure walczyć z tym problemem - maksymalnie ułatwia tworzenie aplikacji w których można unikać tych problemów poprzez ...

Każde z kolejnych rozszerzeń CLR ma jakiś motyw przewodni (LINQ, Generics) ostatnie dwa update'y są ze sobą ściśle powiązane - chodzi o programowanie równoległe. Ostatni wprowadzone rozszerzenie dla obsługi asynchronicznych żądań jest dostępny już w .NET 4.5 - To właśnie wspominany przez Nas Async, który ułatwia tworzenie responsywnych i odpornych na zakłócenia aplikacji.

**Rx - programowanie funkcyjne**

Framework który dostarcza nowych rozwiązań - obserwowalne kolekcje, pewne podstawowe elementy funkcyjne (zip, map, reduce) do języków imperatywnych. Obecnie bez problemu dostępny w C#, VB i JavaScript - sam korzystam z tego ostatniego już ponad rok chwalę sobie ten framework, bardzo dobry i stabilny.

**Async - programowanie imperatywne**

Główny bohater, rozszerzenie języka i łata na kompilator a nie na CLR - przejrzyste dla języka pośredniego, może być zaimplementowane wszędzie tak naprawdę.

Sposób implementacji może być różny dla różnych języków np. C# i VB i dla JavaScriptu mamy Promises (pl. obietnice) - ale tak naprawdę zasada jest identyczna, patterny (wzorce) są spójne obiekty za pomocą których możemy obsługiwać asynchroniczne wywołania podawać continuation callback, error handlery i callbacki dla postępu.

Dodatkowe słowa kluczowe które dodane są do kompilatorów (nie do jezyka pośredniego) - co znacznie ułatwia używanie tego feature w innych środowiskach, bo można pisać aplikacje kompatybilne z .NET 4 a nadal korzystać z async i await.

Dzięki temu - implementowalność tego typu rozwiązań zależy bezpośrednio od tego czy mamy dostępny ITask i pochodne z tej grupy. Oczywiście na platformie Windows 8, zmieniamy profil na .NET 4.5 i z automatu bez żadnych dodatkowych DLLek i patchy mamy dostęp do nowych rozwiązań.

# Przejście:

Zaraz po wypuszczeniu Windows 8 Developers Preview w październiku pojawiły się kolejne rozszerzenia związane z .NET 4.5 – jednym z nich jest ASP.NET MVC 4 który niesie ze sobą wsparcie dla mobilnych stron internetowych tak nieodłącznie związanych z usługami zawartymi w chmurze. Ale zanim opowiem o nowych elementach, warto przypomnieć sobie różnice i czym tak naprawdę jest ASP.NET MVC.