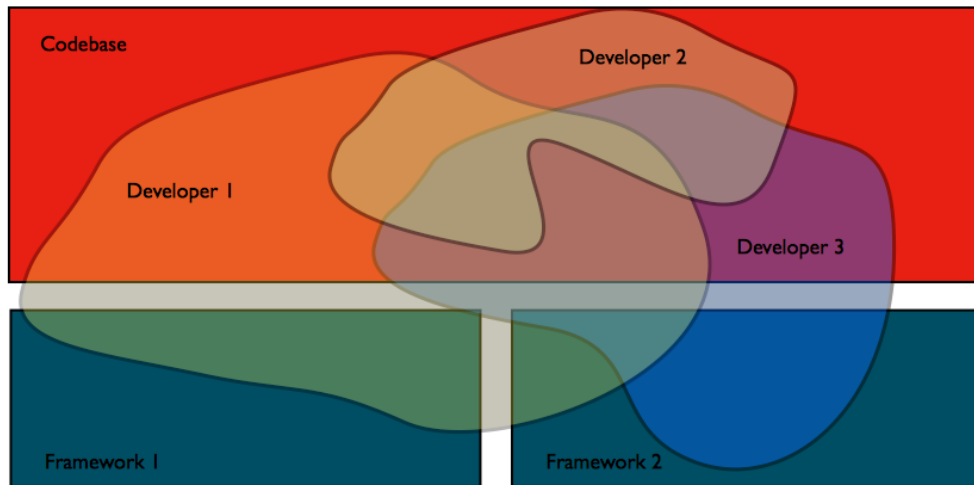


Competence Debt

A *Technical Debt* fogalom már rég befészkelte magát a (vezet) fejlesztők és az архитеktek fejébe, ugyanis ez a "technikai adósság" egy darab szám, ami többé-kevésbé jól jellemzi a forráskód állapotát.

A forráskód ugyanis olyan, hogy a fejlesztés során folyamatosan változik. Ha rendszeresen jönnek üzleti igények, amelyeket az üzlet minél hamarabb éles üzembe szeretne látni, hogy pénzt tudjon keresni, akkor a gyors megoldások mentén beszivárognak rossz megoldások is. Ezeket a tákolásokat elbb-utóbb kell oldani, mivel hibákat rejtenek magukban a le nem kezelt használati esetekre, illetve a fejlesztők képesek precedens értéknek tekinteni ezeket a megoldásokat és egyszerűen lemásolják, mint gyors megoldás.

A forráskód teremtette adósság növekedésnek másik ága az, hogy a világ is változik a meglévő forráskód körül, ami jó megoldás volt pár hónapja vagy éve, az mára már elavult. A használt keretrendszerek is változtak, esetleg új keretrendszerek jöttek ki, amelyekkel megoldhatóak az eddigi kerül megoldások...



The biggest difference between the two types of debt is that while technical debt grows faster the more you change a codebase, competence debt grows faster if you stop changing it!

Szóval itt az új fogalom, a **Competence Debt**, amely sokkal rejtettebb, mint a technikai adósság, ugyanis a fejlesztők kompetenciájáról van szó: arról, hogy a meglévő kódbázist és a használt keretrendszereket mennyire ismerik, mennyire járatosak ezeken a területeken. A nagy különbség pedig az, hogy ha nem nyúlunk a forráskódhoz, akkor a kompetencia adósság jobban növekszik, mint a technikai, amely általában akkor növekszik, ha üzleti nyomásra módosítjuk a kódot és nem állunk meg rendet tenni.

Nálatok mekkora a *Competence Debt*? 😊