Pokayoka - samen werken aan kwaliteit in de bouw

**1: Omschrijving**

Geef een omschrijving van het project: wat ga je ontwikkelen en wat is het doel. Dit mag nog een functionele omschrijving zijn.

Hoe stuur je in een bouwproces op kwaliteit? Tot nu toe bestonden er alleen tools als Ed Controls en Snagstream, die door het doen van opleverinspecties lieten zien dat er nogal wat schort aan de opgeleverde kwaliteit. Er kunnen fouten worden geconstateerd, die in de praktijk niet worden omgezet naar acties voor praktijkmensen, zodat fouten gemaakt blijven worden. Racido ontwikkelt programmatuur om in de bouwsector alle processtappen (van calculatie tot werkvoorbereiding tot aan oplevering) te beheersen en te administreren op diverse detailniveau’s van diverse actoren (opdrachtgever, (onder)aannemer, individuele werknemers) en verbeteringsfouten te verwerken. De ontwikkeling van het bouwwerk zal op elk moment in de tijdsplanning (in 2D en 3D) te tonen zijn inclusief de hieraan te koppelen milestones voor elke (onder)aannemer en taken met deadlines voor elke betrokken werkman. Het wordt mogelijk om tijdens het project opgedane kennis te registreren, zodat ook andere partijen waarvoor deze kennis van belang kan zijn, zich hiervan kunnen voorzien. Doelstelling van de te ontwikkelen programmatuur is om het bouwproces zodanig effectief en efficiënt in te richten dat de te ontwikkelen planning per bouwcasus gegarandeerd compleet is, onbedoelde fouten worden voorkomen (poka yoke) en de faalkosten tot een minimum worden beperkt. De basis voor de te realiseren functionaliteit is het Bouw Informatie Model (BIM) dat geïmporteerd zal worden in de te ontwikkelen architectuur. BIM levert op functioneel niveau de structuur en de fasering van het bouwproces. De architectuur moet het mogelijk maken dat een ieder wijzigingen vanaf zijn mobiel kan doorvoeren, waarbij zowel on- als offline synchronisatie wordt geborgd. De architectuur zal webbased worden aangeboden, waarbij de reactietijd van de pagina’s gegarandeerd instant moet zijn.

**2: Technische problemen bij de ontwikkeling van de programmatuur**

* De wens ook offline te kunnen werken in de architectuur, wijzigingen op te slaan en informatie samen te voegen, leidt tot een datasynchronisatieprobleem;
* Snelheid van dataconversie: binnen twee seconden moet elke betrokken actor bij zijn gegevens over het bouwproces kunnen komen vanaf een link op zijn mobiel, inclusief de 4D data waarbij op elk gewenst planningsmoment de status van de bouw op detailniveau wordt gevisualiseerd;
* Versleuteld opslaan van de gevoelige data icm het ontwikkelen van een conflictresolutie als oplossing voor de snelheid van dataconversie is technisch complex. Zeker omdat binnen de eigen te ontwikkelen database bestaande technische oplossingen niet gebruikt kunnen worden.
* Hoe kan snelle kwalitatief hoogwaardige fout/conflictherkenning worden gegarandeerd?
* Hoe projectie maken van 3D naar 2D tbv slicing van 3D IFC modellen voor verwerking op 2D iPad en van 3D naar 4D.
* Hoe kan programmatuur zo generiek mogelijk worden ontwikkeld zodat deze geschikt is voor diverse exportplatforms en voor importeren diverse centraal gedeelde verwerkersvoorschriften (van in dit geval leveranciers van bouwmaterialen)?

**3: Mogelijke oplossingsrichtingen**

Wat is voor elk genoemd technisch knelpunt een mogelijke technische oplossing? Beschrijf deze oplossing in IT jargon. Benoem ook deliverables

Racido ontwikkelt een architectuur die het mogelijk maakt informatie naar data te converteren en vice versa en gewenste informatiedata te genereren op op elk niveau en elk moment on- of offline. Er wordt daarbij generieke code ontwikkeld om verwerkingsinstructies te genereren op alle gewenste actorenniveaus.

Er wordt een methode ontwikkeld voor datasynchronisatie op de database ook wanneer offline wordt gewerkt. Hiertoe worden bij meerdere schrijfacties prioritering van de client en conflictresolutie ontwikkeld. Het encrypten en decrypten van data vindt plaats op basis van te ontwikkelen cryptografische versleuteling in de vorm van een security first design. Technisch risico: cryptografische versleuteling icm conflictresolutie.

De ontleding van data over twee assen (structuur en fasering) is de basis voor het te ontwikkelen fractal planning algoritme dat snelle datageneratie op voldoende detailniveau moet garanderen voor elke unit of work in de gehele keten. Om snelheid te borgen wordt a-synchroon geprogrammeerd.

Gekozen wordt voor een te ontwikkelen oplossing in Type Script ipv Javascript om eventueel aanwezige inconsistenties al tijdens het compileer te kunnen detecteren. Technisch risico: De IFC voor BIM kent geen implementatie in Type Script en dit zal ontwikkeld moeten worden. Om het coderen te simplificeren zal een eigen api worden ontwikkeld.

Projectie is technisch complex tgv “plat slaan” van informatie (bv maatvoering) en conversie naar bruikbare data (zoals symbolen). Tbv van 3D naar 4D projectie zal VR rendering worden ontwikkeld. Technisch risico: 3D ontwikkeling en -interactie binnen Browser. Complexiteit bij mobiel gebruik en toekomstige nu nog onbekend devices is extra groot.

Er wordt een separate api ontwikkeld voor integratie van externe data. De database zal gedistribueerd worden ontwikkeld om conflicteren met diverse databronnen te beperken en tijdsverlies bij dataconversie te beperken.

**4: Programmeertalen, ontwikkelomgevingen en tools**

Welke bestaande programmeertalen, ontwikkelomgevingen en tools ga je gebruiken bij deze ontwikkeling.

Datasynchronisatie zal plaatsvinden dmv levelDb ecosysteem. 2 papers worden geïmplementeerd en gekoppeld obv Level DB.

Hexastore.pdf wordt gebruikt voor indexering en querying. A Trustless Decentralised Bandwidth Incentive van Mark Nadal.

**5: Technische nieuwheid**

Waarom is het project voor jullie technisch uitdagend? Welke technische onzekerheden spelen hierbij een belangrijke rol? Wanneer het project kan worden gerealiseerd met gangbare technieken of algemeen bekende werkingsprincipes, dan wordt geen subsidie verstrekt. Uit het verhaal moet duidelijk blijken dat jullie je nek uitsteken op programmeringsgebied.

**SpotDB - een gedistribueerd graph database systeem - binnen de browser**

We ontwikkelen SpotDB, een nieuw graph database systeem, dat op basis van predicaten “subject - relatie - object” alle data op kan slaan. Op alle permutaties van deze relaties moeten indexes worden gegenereerd, om het doorzoeken van de data snel te kunnen laten gebeuren. De data moet snel op basis van verschillen tov de laatste synchronisatie kunnen worden gesynchroniseerd. Data moet kunnen worden gepersisteerd op de server (levelDB) en de client (indexedDB). Een gedistribueerde graph database bestaat nog niet, laat staan eentje die binnen de browser werkt. De meerwaarde van een dergelijk systeem is echter enorm, voor onze en ook voor andere use cases.

De benodigde stapjes om dit te realiseren staan beschreven in de genoemde 2 papers, maar zijn nog niet succesvol geïmplementeerd; we zullen dit dus zelf moeten ondernemen.

**Versleutelde opslag van data**

Grote aannemers gaan op basis van onze software ziekenhuizen en bankgebouwen bouwen. Alleen zij moeten dus toegang kunnen krijgen tot hun data. Zelfs wij als developers moeten geen toegang kunnen krijgen tot hun data. Door de data cryptografisch versleuteld op te slaan, kunnen alleen de eigenaren van de data deze data interpreteren. Een ander kan hier zonder de benodigde sleutels niets mee. Wij hebben nog geen ervaring met het versleuteld opslaan van data en we moeten nog goed in kaart brengen hoe we hiermee om kunnen gaan bijvoorbeeld wanneer de gebruiker een wachtwoord kwijt is.

**Database conflictresolutie**

Onze gedistribueerde databases zal master-master replicatie/synchronisatie gaan ondersteunen. Het kan voorkomen dat 2 personen dezelfde data hebben gewijzigd. Dan moeten er voorzieningen worden getroffen om de data op een betekenisvolle manier weer samen te kunnen voegen. Wij gaan dit op het laagste detailniveau doen; per record attribuut wordt bijgehouden wie het wanneer heeft gewijzigd.

**Bruikbaar maken van IFC model binnen browser**

IFC is een enorm uitgebreide standaard. Een gemiddeld IFC model is 300MB groot. Hier valt niet performant mee te werken binnen de context van een webbrowser. Dit biedt echter wel enorm veel voordelen voor onze gebruikers. Wij delen IFC op in informatie en geometrie en ontwerpen en ontwikkelen hiervoor onze eigen vertaling van IFC naar iets wat bruikbaar is binnen de browser.

**4D plannen**

Onze gebruikers gaan straks de individuele IFC elementen (funderingen, binnenmuren, etc) inplannen in onze planning module. Onze viewer moet projecties kunnen maken van de verwachte situatie op een datum/tijd gebaseerd op de huidige planning en de geometrie van de elementen. Dit moet bovendien voldoende performant gemaakt worden binnen de browser.

**6: Welke nieuwe principes op het gebied van IT worden verwerkt in de te ontwikkelen programmatuur?**

* **Graph databases**
* **Gedistribueerde databases**
* **Cryptografie**
* **3D rendering binnen de browser**