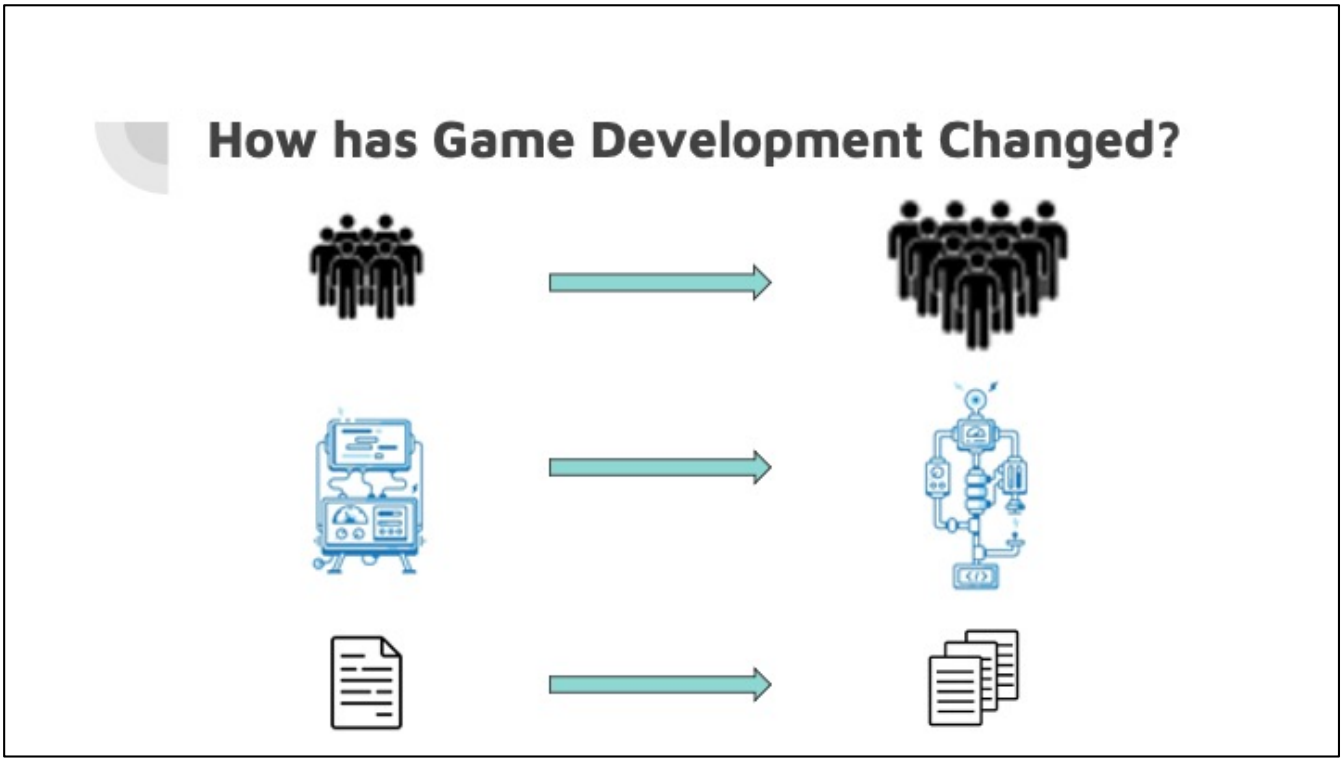
Vi skal presentere artikkelen “Software Architectures and the Creative Process in Game Development”

Artikkelen tar for seg spørsmål som:

- Hvilken rolle software arkitektur har i spillutvikling
- Hvordan spillutviklere håndterer endringer av software arkitekturen
- Hvordan den kreative utviklingen støttes og spiller sammen med arkitekturen
- Hvordan spillutvikling har endret seg gjennom de siste årene

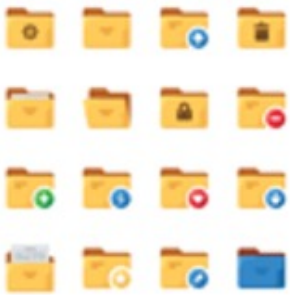


Spillutvikling er ganske forskjellig nå, fra hva det var i begynnelsen.
I begynnelsen var det ofte små team, hvor software ofte bestod av noen få essensielle moduler. Fokuset lå i mindre grad på software arkitekturen og mer på hvordan man kunne lage interessante spill med den begrensede hardwaren man hadde.
Spill evolusjonen har resultert i større og mer komplekse arkitekturer. Arkitekturene skal kunne være mer fleksible og støtte den kreative prosessen. Nyere spill krever også flere mennesker med ferdigheter innenfor flere forskjellige disipliner: datakunnskap, digital kunst (3d modellering, teksturer og sprites), musikk og lyddesign, og spilldesign. Det oppstår fort et skille mellom de tekniske og kreative teamene som kan skape utfordringer.

En annen ting er at spillutvikling skiller seg også fra annen software-utvikling, ved at det ikke i like stor grad er påvirket av funksjonelle krav. Det eneste kravet er at spillet skal være engasjerende og morsomt.
Ellers er det fortsatt mye ikke-funksjonelle krav fordi spill ofte har real-time requirements. Disse kravene kan være med på å gjøre utviklingen utfordrende.

Related work

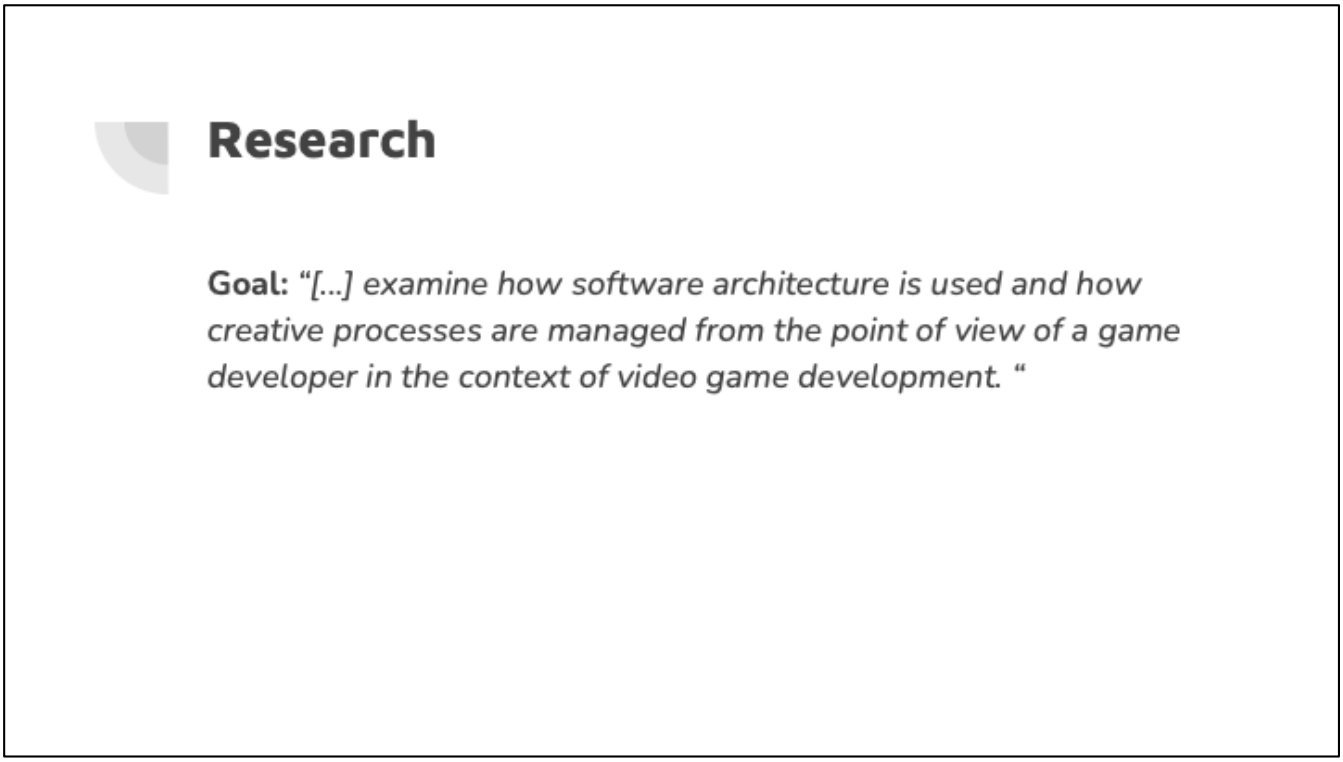
Unrealistic scope
Feature creep
Crunch time



Asset handling

Poor communication
between disciplines

Tidligere arbeid viser at spillutvikling skiller seg fra vanlig software utvikling på andre måter.
Spillutvikling har ofte problemer med urealistisk scope, og dårlig tid med mye crunching.
Spill utvikling må også i mye større grad håndtere forskjellig assets, som bilder, modeller og kode.
Til slutt ser vi også at kommunikasjonen mellom det tekniske teamet, management og det kreative teamet kan være utfordrene.



Dette nye studiet bruker et sett med 20 påstander hvor deltakerne skal angi i hvilken grad de er enige. Deltakerne hadde også mulighet til å kommentere med ekstra informasjon.
Målet med undersøkelsen var å undersøke hvordan software arkitekturen ble brukt og hvordan den kreative prosessen blir håndtert av spillutviklerne.

What Role does Software Architecture Play in Game Development?

- Handle complexity of engines, libraries, APIs
- Manage changing requirements
- Performance
- Game concept <-> Game engine
- Enable the creative team

Fra undersøkelsen ser vi at software arkitekturen er viktig for å håndtere kompleksitet, og at en robust arkitektur er viktig for å kunne håndtere endringer, i tillegg til å oppnå høy ytelse.
Likevel viser det seg at bare 50% mener at hovedmålet med arkitekturen er ytelse, siden funksjonalitet, fleksibilitet og brukbarhet også er viktig. Med brukbarhet mener vi innebærer testing, mulighet for automatisering etc.

Et viktig punkt er hvordan spill konseptet og arkitekturen påvirker hverandre. For eksempel kan man ha en generelle game engine, som er fleksibel og kan brukes til mye. Samtidig kan en generell engine skape mye overhead og redusere ytelse. Alternativt man kan spesialisere arkitekturen for spillet, men miste mye fleksibilitet. Eksisterende software har også en liten påvirkning på spill konseptene. Det kan hende at konseptet må endres for passe med arkitekturen, men det skjer stort sett bare dersom endringene i arkitekturen vil koste for mye. Konseptet påvirker nok software i større grad enn software påvirker konseptet.

Undersøkelsen viser også at software is stor grad sikter mot å hjelpe det kreative teamet mer å realisere deres tanker og idéer, det kan for eksempel være gjennom høynivå GUI editor eller scripting

Changes during development

- Creative freedom vs technical limitations
- Adding and changing features
- Creative team vs technical team vs management

Q7 - trade-off mellom kreativ frihet og tekniske begrensninger. Hender at designerenes ideer ikke er mulige på grunn av tekniske problemer, men som regel skal design settes først, og programmererne har som rolle å realisere det.
Q8/9 - Det kreative teamet får som regel lov til å be om endringer og ny funksjonalitet i løpet av utviklingen. Omfanget og gjeldende utviklingsfase påvirker om endringene blir implementert.
Q10 Hvor enkelt det er å legge til ny spillfunksjonalitet kan avhenge. Store endringer kan fort bli dyrt, og jo lengre ut i utvklinging desto vanskeligere blir det. Selv om det er mulig/enklet, blir det også frarådet ettersom det kan bli for mye. Enklere for emergence(systembaserte) spill enn scripted(hardkodede) spill.
Q11 Det avhenger også hvor vidt designerene må finne seg tilstrekkelige med de verktøyene de allerede har.
Q12 Splitt mellom de kreative og management om hvem som bestemmer endringene. En dialog. Ide fra kreative, tekniske sier om the er realistisk, management om de har råd til det

How are Creative Development Processes Managed and Supported?

- Flexible engines
  - Dynamic loading
  - Scripting
- Rapid prototyping
- Creative team included in development feedback
  - Defining requirements

Som sagt tidligere er game engines bygget for å støtte den kreative prosessen. Den inkluderer ofte verktøy som gjør den fleksibel, som dynamisk lasting av modeller, og høynivå scripting. Disse verktøyene er med på å muliggjøre rask prototyping og testing. Det åpner for eksempel opp for å raskt designe og teste nye baner, eller endre NPC oppførsel. Siden det kreative teamet ofte ikke har full forståelse for implikasjonen på ytelsen, er det viktig å kunne samarbeide med det tekniske teamet. Det er også viktig at det kreative teamet er med på selve utviklingen for å kunne skape den funksjonaliteten som trengs.

Evolution of game development

- Third party & middleware
- Game development easier?
- Closer to ordinary software development?
  - Differences

Q17 - Flere som sier de bruker mer tredjepart enn tidligere, og flesteparten mener også at det er viktig for utviklingen (Q18).
Q19 - På et teknisk hvis, pga bedre og mer tilgjengelige verktøy og spillmotorer, er utvikling enklere. Men kravene og kompleksiteten har også øket, så spillutvkling generelt er ikke utfordrene.
Q20 - Det er uenighet om spillutvklinging er blitt mer som ordinær utvkling. Kreativ problemløsning, dagens verktøy
Endringer definert i tidligere litteratur:

- Livssyklus: Ofte mindre arbeid etter lansering.
- ikke-funksjonelle krav
- brukervennlighet

Conclusion

Til konklusjon, spillutvkling hjar utviklet seg til store teams som jobber på komplekse systemer innenfor mange forskjellige disipliner, som sammen jobber mot ikke-funksjonelle krav.
Programvarearkitekturen i spillutviklinging har i rolle å håndtere noe av probleme som oppstår pga dette. Spillmotorene bør støtte kreativiteten ved å være dynamiske og effektive.
Arkitekturen er også viktig for å håndtere kompleksitet. Det er en balanse mellom en spesialisert effekt spillmotor of en fleksibel generel spillmotor.
Vi ser at konseptene og det tekniske har påvirkning på hverande, men konseptet priorieters oftest.
Til slutt så vi at tredjeparts software er mer viktig enn tidligere og at det er med på at den tekniske delen av spillutvkling er enklere.