Design

# Visie

## Variaties

De mogelijke variaties in Jabberpoint die ik zag zijn:

* Jabberpoint kan naast Load en Save andere acties krijgen later zoals Export of Import
* Jabberpoint kan mogelijk naast XML en Demo, nieuwe Formats krijgen
* Presentatie doet naast navigeren en Slides aansturen om te tekenen ook handelingen zoals het luisteren naar wat de controllers binnenkrijgen. Ik denk dat de presentatie in dit geval als het programma groter wordt, nog meer verantwoordelijkheden krijgt.
* Nieuwe soort SlideItems
* Verantwoordelijkheden en complexiteit van controllers die groter worden OF van Presentatie met de afhandeling van ‘luister-acties’.
* Meer controllers, zoals een MouseController bijvoorbeeld.

## Problemen oorspronkelijke design

Dit zijn de problemen in het oorspronkelijke design volgens mij:

* Presentatie heeft teveel verantwoordelijkheden en zou alleen informatie moeten bevatten en de slides aansturen om zijn slide items te tekenen.
* Controllers zijn qua uitbreidbaarheid niet goed en de communicatie naar Presentatie is met uitbreiden ook misschien een probleem.
* Load en save naast de formats is niet goed uitbreidbaar
* Draw wordt niet apart rekening mee gehouden.
* Geen factories. Andere objecten maken andere aan.

# Ontwerp

## Factories

Jabberpoint is mijn geval verantwoordelijk voor het aanmaken van alle objecten ongeveer. Omdat ik veel met aggregatie werk neem ik even aan dat als je bijvoorbeeld Slide aanmaakt, je in mijn geval, ook Annotation aanmaakt.

Als Presentatie aangemaakt wordt, is daarna de Factory verantwoordelijk voor het aanroepen van Slide en dat gaat zo naar beneden tot Slide alle SlideItems aanmaakt en etc. Of dit allemaal onder 1 factory Presentatie kan weet ik niet maar dat kan een 2de optie zijn als dit heel dom blijkt te zijn maar dat weet ik niet.

## Save/Load Save

Jabberpoint regelt alle basis functies van het programma, dus het stuurt dingen zoals save en load aan.

Load zou ervoor moeten zorgen dat alle factories worden aangeroepen omdat de presentatie geladen moet worden en ingevuld moet worden. Ik weet even niet of dit werkt zo maar het is snel genoeg recht te zetten denk ik, zou dit niet helemaal logisch zijn.

## MVC en Observer

MVC en observer geeft heel wat structuur en omdat er al controllers zijn, past het perfect.  
Het model geeft veranderingen in het model door naar de view, en deze gebruikt dan update(). Update verwijst eigenlijk door naar de PresentatieHandler weer in mijn geval om alle objecten zichzelf te laten tekenen.

Ik heb 2 views, waar de 2de SlideViewerAnnotation is. Mijn idee was dan om het model te laten bepalen met een status welke van de 2 views subscribed moet zijn zodat deze luistert en aangepast kan worden. Omdat er toch iets van een status nodig is voor de mouseController of het iets moet doen met de muisbewegingen. Deze status houdt het model dan zelf bij, waar de controllers op reageren.  
2 views hoeft niet perse nodig te zijn als het te lastig wordt, maar omdat we een soort van overlay annotatie hebben kan het ook gezien worden als 2 slides in 1, waar 1 slide exclusief is voor annotatie.

De view luistert naar user interaction acties en geeft deze dan door aan de controller. Dit kan nog worden uitgebreid met een Event klasse of iets dergelijks dat gebruikt kan worden door de view om te kijken naar welke controller het moet en of dit moet..

De controllers geven alleen zaken door in principe maar kunnen waarschijnlijk wel even controleren of het een actie vereist.

## PresentatieHandler

Om het voorgaande verhaal op te vangen en daar tussen alle zaken te regelen heb ik een PresentatieHandler. Deze vangt vanuit de controllers op wat moet gebeuren, wat de controllers uitkiezen. Daarna is het een tussenpersoon tussen waar het model uit bestaat, de presentatie, en het echte PresentatieModel dat ik heb dat weer doorgeeft aan de view wat veranderd is.

PresentatieHandler is een aanspreekpunt voor alle elementen van de applicatie zowat en het idee is om deze op te breken in meerdere klassen die verschillende dingen doen, misschien een navigatie klasse of iets dergelijks, zodat vanuit deze PresentatieHandler acties gedelegeert kunnen worden.

## Slide uitbreiding en Annotatie

Ik heb annotatie vastgemaakt aan Slide omdat ik Annotatie zie als een leeg plaatje die je kan opbouwen met de uiteindelijke functie die over de slide gaat. Dan kan de info van de pen, de kleur en de dikte waarschijnlijk in de Annotatie opgeslagen worden en dan als Model gebruikt worden door de View en Controllers.

Annotatie zit vast aan een Slide omdat het een soort van slideItem is in mijn geval wat wel altijd aanwezig moet zijn in een slide, alleen is het leeg als niemand wat annoteert.

## Draw en Strategy

Elk object tekent zichzelf nog altijd, alleen is het schrijf gedrag los gemaakt van het object met de strategy pattern.