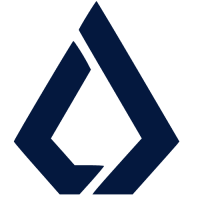
Case study: Messaging

Seven step approach



Project: Lisk Delegates

Project team: BBB-BC

Team: Joeri Berman (2834499)

Luc Urlings (3071081)

Stijn Baltessen (3064565)

Wouter Vandenboorn (3099571)

Floris Feddema (3188256)

Davy de Haas (3099695)

Opdrachtgever: Study case, Bartosz

Versie: 0.1

Versie datum: 31 maart 2020

Status: Concept

# Document revisies

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Wijzigen** | **Auteur** | **Datum** |
| 0.1 | Opzet Document | Stijn Baltessen | 31-03-2020 |

# Inhoudsopgave

[**Document revisies**](#_bu2l4jqkimrr) **1**

[**Inhoudsopgave**](#_5pg2mng0n2po) **2**

[**Introductie**](#_8anxnj4hrgfu) **4**

[**Beschrijving case**](#_sdkb7p1qxmgj) **5**

[**Seven step approach**](#_djfvx4apkl6h) **6**

[**Discussie met de groep**](#_y9a4y0h6wipy) **7**

[Interpretatie groep](#_hh5majw2stu5) 7

[**Identificeer de vragen**](#_49pivsptx4th) **8**

[Wat is een system of systems?](#_4pe4fskiqjy9) 8

[Wat is middleware tech stack?](#_wi6s8xd7cpyd) 8

[Wat zijn enterprise integration patterns?](#_9un3jofcnpjz) 8

[Wat is de selectiecriteria van de middleware tech stack voor deze case study?](#_bc5cuxvf9t5y) 8

[**Brainstorm**](#_z6hvd6w1uz8j) **9**

[**Analyseer en structureer**](#_xew0oaz6sqtn) **9**

[**Formuleer leerdoelen**](#_35z77npcvh3l) **11**

[**Individuele studie**](#_10a2mcppftd0) **12**

[Enterprise integration patterns](#_7knpacyoael1) 13

[Wat is een enterprise integration pattern?](#_hnm51f83pbw7) 13

[Application integration](#_far9qdx9sc75) 13

[Enterprise integration pattern](#_8cfy4su0i25f) 13

[Waar moet je aan denken tijdens het maken van een applicatie integratie?](#_h418xetl1l4x) 14

[Application coupling](#_56vs01z3rmer) 14

[Integration simplicity](#_1msigo9prxyb) 14

[Integration technology](#_r2mub08qu6m0) 14

[Data format](#_d8j0npsp4em4) 14

[Data timeliness](#_ninehsrid1n) 14

[Data of functionaliteit](#_nx3w9p4lpdzt) 14

[Asynchroon](#_b5kygnjvjes0) 15

[Welke verschillende enterprise integration pattern zijn er?](#_mjppnyxn8nph) 15

[File transfer](#_1u2gapn6ychv) 15

[Shared database](#_zidaa6d9a1d0) 15

[Remote procedure invocation](#_7llp0cwqoryu) 16

[Messaging](#_76cnyaa4kgo5) 16

[Beoordelingsmatrix](#_zhrugu1plnvr) 17

[Waarom zou je een enterprise integration pattern gebruiken?](#_iccmnsoj2v2f) 17

[Op welke manier heeft een enterprise integration pattern toepassing op onze case?](#_cn8j1f623r03) 18

[Middleware techstack](#_4sdgasot85j8) 19

[Wat is een tech stack?](#_tu85tkr79m4h) 19

[Wat is middleware?](#_9sdykp24ueb0) 19

[Wat is middleware techstack?](#_ml61y4tdygje) 20

[Welke soorten middleware zijn er?](#_pw1r9tasyco5) 20

[Wat is de selectiecriteria van de middleware tech stack voor deze case study?](#_i4waq3nc55b9) 21

[**System of Systems**](#_yfnbz6knenx) **22**

[Wat is System of Systems?](#_vrj4fvkxkqj9) 22

[Elementen](#_f0xkr8lpy7w3) 22

[Wat voor types SoS zijn er?](#_dekefen13pgn) 22

[Directed](#_3usxgc7pftzx) 23

[Acknowledged](#_9uuaes6gwt29) 23

[Collaborative](#_vst8eric894g) 23

[Virtual](#_gxhc6lbrl3bs) 23

[Wat zijn de best practices?](#_pcmcbblmykdz) 24

[**Conclusie**](#_a5160cf0bvz2) **25**

[**Literatuurlijst**](#_y4hx91mbmk6z) **26**

# Introductie

Om de study case messaging op te lossen gaan we de seven step approach toepassen, in dit document zal elke stap worden uitgelegd. Uiteindelijke conclusie wordt op het einde opgeschreven.

Eerst wordt de study case uitgelegd en hierna de zeven stappen die genomen worden.

# Beschrijving case

Een innovatief groot AI project waar meerdere partijen aan werken. Het probleem van dit project is dat elke partij hun eigen geprefereerde technologie gebruikt.

Nu is de taak aan ons om te kijken hoe wij een systeem integratie patroon kunnen toepassen om een “system of systems” te maken. Hierbij wordt gekeken naar “middleware tech stack”

We willen in stap 2 een middleware oplossing dat bijna perfect is voor (delen) van het project vinden. We willen hiervoor een duidelijke selectiecriteria opstellen en de verschillende alternatieven hiermee vergelijken.

# Seven step approach

Hier staan de zeven stappen van de [seven step approach](https://portal.fhict.nl/Studentenplein/LMC/1920vj/Software%20engineering/Afstudeerfase/EnterpriseSoftware/CaseStudies/SevenStepsApproach.docx) methode die bij problem based learning voorkomt.

1. Discuss the case

2. Identificeer de vragen

3. brainstorm over potentiële bekende oplossingen

4. analyseer resultaat van brainstorm sessie

5. formulate learning objectives, wat moet er nog geleerd worden

6. individuele study om de kennis van 5 te krijgen

7. Discussieer de bevindingen van 6 hoe dit 5 en 2 beantwoord.

# Discussie met de groep

Hier wordt gediscussieerd wat de case study van ons vraag en of iedereen de opdracht snapt

## Interpretatie groep

Een tech stack moeten verzinnen en designen met de context die we hebben in de case study.

Wat de architect heeft beslist wat in het project moet komen:

|  |
| --- |
| • React in almost a real-time to the inputs coming from audio, video and wearable sensors analysis services  • Process the sensory data from environment sensing IoT devices and supply it for further analysis to a dedicated service  • An AI component will decide on how the system needs to change  • A mixed reality component will need to react to change requests send by the AI component  • Data hungry AI component needs to learn from all the interactions of all the participants in all the training. Data need to be stored across the system for later retraining and simulations.  • There will be no budget for extensive testing, instead it was agreed to record as much as possible of the system behaviour so it can be analysed by eternal tools later. |

Hieruit kunnen we de belangrijkste punten halen om te weten wat de criteria is voor de keuzes die we moeten maken.

Luc: “Je wilt het systeem analyseren en een system of systems kan je hier mee helpen.”

De probleemstelling waar wij op zijn gekomen is:

**Één middleware tech stack vinden dat het beste bij het project past dat wordt beschrijving in de case study.**

# Identificeer de vragen

We hebben nu de probleemstelling, we gaan nu in dit hoofdstuk kijken welke vragen hierbij komen om het probleem op te lossen.

Per vraag zullen we onderbouwen waarom wij denken dat dit van belang is voor het oplossen van het probleem.

## Wat is een system of systems?

We moeten met een voorbeeld komen hiervan, dus is het van het belang dat we het snappen. En de benodigde kennis hierover hebben.

## Wat is middleware tech stack?

We moeten een een middleware tech stack kiezen, hoe doen we dit als we niet weten wat dit is.

## Wat zijn enterprise integration patterns?

Niemand weet wat dit is. We denken dat dit wel relevant is want iemand die hier ervaring mee heeft moet het probleem aanpakken in de case study.

Dus we denken dat dit betrekking kan hebben met de vorige vragen.

Het antwoord op deze vraag zal bepalen of dit bruikbaar is voor het antwoord van de case study.

## Wat zijn de selectiecriteria van de middleware tech stack voor deze case study?

We willen verschillende middleware tech stack met elkaar vergelijken en hierna eentje kiezen die het beste past bij de study case.

Om dit proces te onderbouwen kunnen we selectiecriteria opstellen waar de case study zegt dat de middleware aan moet voldoen. Hiermee kunnen we dan uiteindelijk de juiste keuze maken.

# Brainstorm

Hier bespreken we wat de groep al weet en identificeren we de mogelijke oplossingen. Per mogelijke oplossing zeggen we ook waarom we denken dat een mogelijke oplossing is.

* Er is geen voorkennis binnen de groep over system of systems.
* Er is geen voorkennis binnen de groep over middleware tech stacks.
* Er is geen voorkennis binnen de groep over enterprise integration patterns.
* Er is geen voorkennis binnen de groep over de selectiecriteria van de middleware tech stack.

## 

# Analyseer en structureer

Momenteel hebben we geen voorkennis over ieder onderwerp dus kunnen we dit ook niet analyseren.

# Formuleer leerdoelen

Aan het einde van deze study case heeft de student veel kennis opgedaan, met deze kennis beheerst hij de volgende leerdoelen:

* De student weet wat een system of systems is.
* De student weet wat een middleware tech stack inhoud.Hij weet ook hoe hij aan de hand van een context, **selectiecriteria moet opstellen zodat hij de beste middleware tech stack weet te kiezen.**
* De student weet wat enterprise integration patterns zijn. En het verband dat dit heeft met “Middleware tech stack” en “System of systems”.
* De student heeft zich verdiept en toegepast in de de seven step approach.

# Individuele studie

Floris & Stijn Enterprise integration

Luc & Wouter middleware tech stack

Davy & Joeri system of systems

## Enterprise integration patterns

De persoon in de case study die het probleem wilt aanpakken en oplossen. Heeft als ervaring als eerste puntje enterprise integration patterns. Willen kijken wat voor betrekking dit heeft met de opdracht.

Deelvragen:

* Wat is een enterprise integration pattern?
* Welke verschillende enterprise integration patterns zijn er?
* Waarom zou je een enterprise integration pattern gebruiken?
* Op welke manier heeft een enterprise integration pattern toepassing op onze case?

### Wat is een enterprise integration pattern?

De vraag is van belang om te beantwoorden omdat we dan kun inzien of dit betrekking heeft op de opdracht.

#### Application integration

Hoe zorg je ervoor dat verschillende applicaties met elkaar kunnen werken (integreren). Het verschil wat zich voordoet met enterprise integration in vergelijking tot application integration is: “Op basis op van de principes die komen bij enterprise development een application integration maakt.”

#### Enterprise integration pattern

Is dus een pattern die je kan gebruiken om een application integration te maken op een enterprise niveau.

### 

### Waar moet je aan denken tijdens het maken van een applicatie integratie?

#### Application coupling

De minimalisering van de afhankelijkheden die applicaties op elkaar hebben. Wanneer een afhankelijkheid verandert dan verbreekt de integratie van 2 applicaties zolang degene die op de afhankelijk waren niet zich aanpassen. Het interface die ontwikkelt wordt voor het koppelen van applicaties moet specifiek genoeg zijn om functioneel te worden maar generiek genoeg zijn dat het verschillende implementatie veranderingen kan doorvoeren.

#### Integration simplicity

De hoeveelheid dat je een applicatie moet veranderen om deze te integreren moet minimaal zijn.

#### Integration technology

Integraties kan vaak leiden tot speciale hardware of software. Dit kost projecten veel tijd en geld. Dit minimaliseren is een streven.

#### Data format

Tijdens een integratie moet je ervoor zorgen dat er een mogelijkheid is om de format gelijk te hebben of een vertaler er tussen te zetten. Dit zorgt ervoor dat de integratie soepeler verloopt.

#### Data timeliness

In de integratie van verschillende applicaties is het van belang dat de overdracht van data snel is. Het streven is dus om met een hoge frequentie kleinere data pakketten te sturen dan met een lage frequentie zware pakketten.

Ook is het van belang dat de latency tussen data sharing wordt meegenomen in een integratie design.

#### Data of functionaliteit

Tijdens het maken van applicaties die functies gebruiken van een geïntegreerde applicatie moet je nadenken hoe dit verschilt met het aanspreken van een lokale functie. Ook is het van belang om na te denken of een verzoek data is, of dat het een functionaliteit is, die wordt uitgevoerd.

#### Asynchroon

In een lokale applicatie wachten processen vaak op elkaar, dit zorgt ervoor dat functies altijd moeten wachten op antwoorden van andere functies. In een integratie kan dit ook niet het geval zijn.

### Welke verschillende enterprise integration pattern zijn er?

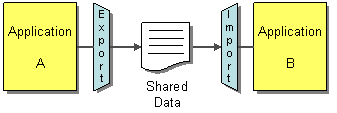
[Introduction to Integration Styles](https://www.enterpriseintegrationpatterns.com/patterns/messaging/IntegrationStylesIntro.html)

Deze vraag wordt gesteld omdat we hiermee, meer inzicht krijgt over de verschillende toepassingen van enterprise integration patterns.

de vier:

* File transfer
* Shared database
* Remote procedure invocation (example: REST)
* messaging

#### File transfer

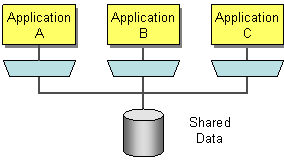


A Slaat een bestand op een locatie op, applicatie B leest bestanden uit deze locatie en verwijdert ze, nadat ze verwerkt zijn. Vaak gebruikt file extensie is XML.

Dit gebeurt niet real time, dit gebeurt met ingesteld tijd intervallen, waardoor je geen “busy waiting time” hebt.

* Het nadeel hiervan is dat het foutgevoelig is en de applicatie lang bezig is met het processen van de bestanden.

#### Shared database



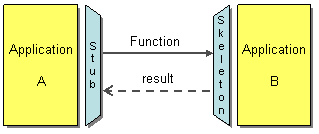
Er is een database, waar de data van verschillende applicaties in wordt verwerkt.

Wanneer applicatie A iets zet in de database wat betrekking heeft op applicatie C dan krijgt C hier niet meteen een melding van. Dit moet applicatie C zelf inzien.

Dit gebeurt niet real time, dit gebeurt met ingesteld tijd intervallen, waardoor je geen “busy waiting time” hebt.

* De een nadeel is dat de data niet real-time is.
* Een voordeel is dat dit minder foutgevoelig dan file transfer, door instellingen die je kan doen om potentiële fouten op te vangen in de database.
* Een voordeel is dat er meer inzicht is op de data die wordt opgeslagen door meerdere applicaties.

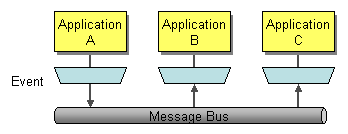
#### Remote procedure invocation



Applicatie B heeft een publiek aanspreekpunt, dit kan aangesproken worden door applicatie A. Wanneer applicatie B wordt aangesproken voert hij aan de hand van waar en waarmee hij aangesproken is een functie uit. Dit resultaat wordt teruggestuurd naar degene die applicatie B aanspreekt.

* Een voordeel is dat het real-time is en niet niet tijdsinterval afhankelijk.
* Een voordeel is dat elke applicatie kan de integriteit van zijn data bewaken.
* Een nadeel is dat het synchroon is. Dus applicatie A wacht op het antwoord van applicatie B.

#### Messaging



Het is Asynchroon berichten sturen van applicatie A naar B. Het is niet benodigd dat applicatie B beschikbaar is op het moment dat applicatie A een bericht zou sturen.

Het is asynchroon, data gaat niet verloren als een applicatie niet beschikbaar is.

* Een nadeel is dat het asynchroon is. En dus niet realtime.

#### Beoordelingsmatrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | File transfer | Shared database | Remote procedure invocation | Messaging |
| Application coupling | HEAVY | HEAVY | LIGHT | LIGHT |
| Integration simplicity | LIGHT | LIGHT | HEAVY | HEAVY |
| Integration technology, External or internal data persistence | EXTERNAL | EXTERNAL | INTERNAL | EXTERNAL |
| Data format | LOOSE | TIGHT | LOOSE | LOOSE |
| Data timeliness | LARGE | SMALL | SMALL | SMALL |
| Data of functionality | DATA | DATA | DATA &  FUNCTION | DATA & FUNCTION |
| Async of sync | ASYNC | ASYNC | SYNC | ASYNC |
| Two way data | ONE WAY | ONE WAY | TWO WAY | ONE WAY |

Uiteindelijke gaan we kijken bij de conclusie wat wij willen uit deze matrix per criteria en dan kijken welke integratie methode het meest van toepassing is.

### Waarom zou je een enterprise integration pattern gebruiken?

Nadat we hebben onderzocht wat en welke enterprise integration patterns er zijn. Kunnen we ook toelichten waarom je dit zou gebruiken.

Door het toepassen van enterprise integration patterns zorg je ervoor dat de integratie van applicaties met eisen die hierbij komen juist wordt gedaan.

Het is van belang om te zeggen dat de integratie van verschillende applicaties binnen een systeem niet altijd op dezelfde manier gaan.

De conclusie is dat je integration patterns toepast om de koppeling van applicaties juist toe te passen.

### Op welke manier heeft een enterprise integration pattern toepassing op onze case?

Het is van belang om te weten waarom het gebruikt wordt, dit is in de vorige vraag beantwoord. Nu onderzoeken we of dit ook toepassing heeft op de case study. En in wat voor mate dit de probleemstelling kan helpen oplossen.

In de study case wordt gezegd dat er 15 partijen zijn met allemaal verschillende systemen zitten te maken, ieder systeem per partij kan een ander techniek gebruiken. Nu is het van belang als deskundige om het volgende te kunnen vast stellen.

Hoe koppel je deze verschillende systemen op een juiste wijze met elkaar. De redenatie achter elke koppeling wordt onderbouwd door een integratie pattern.

## Middleware techstack

### Wat is een tech stack?

*A* ***technology stack****, also called a solutions stack or a data ecosystem, is a list of all the technology services used to build and run one single application. The social site Facebook, for example, is composed of a combination of coding frameworks and languages including JavaScript, HTML, CSS, PHP, and ReactJS. This is Facebook’s ‘tech stack.’*

Een tech stack oftewel een Technology stack is een combinatie van verschillende tools en technieken die gebruikt worden voor een enkele applicatie. Bijv. Facebook maakt gebruik van een techstack met de codeertalen JavaScript, HTML, CSS, PHP en ReactJS. De combinatie van deze talen binnen de context zorgt er dan voor dat dit een techstack is voor de facebook applicatie.

### Wat is middleware?

Middleware is software die tussen de toepassingen die op een server draaien worden uitgevoerd. Middleware is een verborgen vertaallaag en maakt verschillende extra functionaliteiten mogelijk die nodig zijn voor *Enterprise Software*. Het wordt ook wel 'plumbing' (leidingen) genoemd omdat er twee toepassingen met elkaar worden verbonden, zodat gegevens en databases makkelijk door de 'pijp' kunnen worden gestuurd. Dankzij middleware kunnen aanvragen worden uitgevoerd, zoals het indienen van formulieren via een webbrowser of de webserver dynamische webpagina's laten retourneren op basis van een gebruikersprofiel.

*“Voorbeelden van veelgebruikte middleware zijn database-middleware, server-middleware, bericht-georiënteerde middleware, web-middleware en transacties verwerkende middleware. Elk programma omvat gewoonlijk berichtenservices, zodat verschillende toepassingen kunnen communiceren door middel van berichten frameworks, zoals SOAP (Simple Object Access Protocol), webservices, REST (Representational State Transfer) en JSON (JavaScript Object Notation). Hoewel alle middleware communicatiefuncties kan uitvoeren, is het type dat een bedrijf kiest afhankelijk van welke service er wordt gebruikt en welk type informatie er moet worden gecommuniceerd. Het kan hierbij gaan om beveiligingsverificatie, transactiebeheer, berichtenwachtrijen, toepassingsservers, webservers en adreslijsten. Middleware kan ook worden gebruikt voor gedistribueerde gegevensverwerking, waarbij de acties in realtime plaatsvinden in plaats van de gegevens heen en weer worden gestuurd.” [[1]](#footnote-0)*

Een voorbeeld hiervan is:

Stel je hebt 4 verschillende applicaties draaien en er is een klant die ook nog eens 3 verschillende applicaties heeft.

Er zijn plannen om al deze applicaties samen te voegen onder een groot systeem. Dit zal veel ontwikkelingstijd kosten en het wordt dan een zeer complexe applicatie.

Een andere optie om dit te doen wordt dan door alle bestaande systemen op een of andere manier met elkaar te verbinden zodat ze kunnen communiceren vanuit hun huidige omgeving.

Dit is waar Middleware komt kijken, door middleware te gebruiken hoeven we alleen te bestuderen wat de in en outputs en de van alle bestaande services en applicaties zijn. Gebaseerd op de uitkomst van deze studie kunnen we vervolgens een keuze maken voor het beste middleware framework om te gebruiken binnen deze context.

Dit middleware framework is dan uiteindelijk de verbinding voor de communicatie tussen de systemen. De systemen kunnen dezelfde outputs gebruiken en de middleware zorgt er dan voor dat deze data bruikbaar is voor de ontvangende systemen.

### Wat is middleware techstack?

Een combinatie van een tech stack en middleware is een lijst van gekozen middleware software. Denk aan Message Queue middleware, Authentication middleware, Api Gateway

### Welke soorten middleware zijn er?

Message Oriented Middleware (MoM) is een van de verschillende types middleware die er zijn.

voorbeelden hiervan zijn:

* Remote Procedure Call (RPC-based middleware) is een procedure die wordt uitgevoerd op een externe machine als of hij op de lokale machine wordt uitgevoerd.

Alles omtrent verbindingen wordt dan door de middleware voor de ontwikkelaar, hier hoeft de ontwikkelaar zelf dus niks aan aan te passen.

Als developer maak je een procedure ook zoals hij lokaal zou worden uitgevoerd, de benodigde netwerkverbindingen en onderdelen worden dan dus automatisch aangemaakt.

* Object Request Broker (ORB-based middleware) dient als een soort afspraak/contract/interface voor data en methodes die het programma gebruikt. Een client die een interface aanroept op de middleware hoeft niet te weten waar de achterliggende logica wordt uitgevoerd, dit wordt verzorgt door de middleware.
* Message oriented middleware (MoM-based middleware), MoM biedt mogelijkheden voor verschillende afzonderlijke applicaties en services om met elkaar te kunnen communiceren door middel van “messages”.

### Wat is de selectiecriteria van de middleware tech stack voor deze case study?

De volgende selectiecriteria die van toepassing zijn op de case study:

* *latency (moet zo laag mogelijk zijn)*
* *reliability(hier staat niks specifiek genoemd in de tekst, maar het is logisch dat dit zo hoog mogelijk moet zijn)*
* *High data throughput (er zal waarschijnlijk veel data door het netwerk moeten gaan).*
* *interoperability*
* *amount of work that is handled for you (belangrijk om de kosten zo laag mogelijk te houden)*

## System of Systems

### Wat is System of Systems?

Systems of systems is een collectie van systemen die samen hun capaciteiten samen poolen en zo een nieuwe of meer complexe functionaliteiten maken dan wanneer ze los van elkaar zouden werken.

System of systems wordt vaak verschillend beschreven, er is geen 1 definitie. Veel voorkomende beschrijvingen van een SoS zijn:

* Het linken van systemen zorgt voor betere samenwerken van Command, Control, Computers, Communicatie en Informatie.
* SoS zijn op grote schaal gedistribueerde systemen. De componenten van de systemen zijn op zichzelf ook complexe systemen.
* SoS is een methode voor het ontwikkelen, integreren, samen laten werken en optimaliseren van systemen.
* Enterprise SoS heeft de focus van het koppelen van traditionele systeem engineering activiteiten met enterprise activiteiten.

De beschrijvingen verschillen het meeste afhankelijk vanuit welke bedrijfstak ze worden toegepast.

#### Elementen

Sos bestaat uit een aantal core elementen. Deze elementen vormen samen een framework voor het creëren een SoS platform. De elementen zijn als volgt:

* Begin met het begrijpen van de capaciteiten doelen. Dit zorgt ervoor dat alle wensen van de eindgebruiker worden voldaan.
* Bepaal tot in hoeverre deze doelen al voldaan worden door onderliggende systemen. Bijna alle SoS beginnen met onderliggende kwaliteiten en mogelijkheden. Het is belangrijk om rekening te houden met in hoeverre deze al de doelen bereiken.
* Creëer een architectuur voor de SoS dat kan groeien en evolueren. De wensen van de eindgebruiker veranderen continue.
* Monitor en integreer veranderingen binnen de SoS. In een SoS wordt nieuwe development gedaan door de systemen. Het is belangrijk om de veranderingen in de gaten te houden om te zien in hoeverre de verandering een effect heeft op de SoS.
* Anticipeer de impact van veranderingen van buiten de SoS. Een van de grootste risico’s is dat dat een verandering van buiten het systeem impact heeft op de SoS. Dit betekend dat het van belang is dat er een mechanisme moet zijn die plannen en veranderingen bijhoud en bepaald in hoeverre deze impact hebben op de SoS zodat hiernaar gehandeld kan worden.

### Wat voor types SoS zijn er?

De vier types van een SoS zijn gebaseerd op de autoriteit van een SoS op de individuele systemen zelf. In moderne SoS zijn deze vier types meestal allemaal deels aanwezig, met een type dat vaak domineert over de andere.

#### Directed

Bij een directed SoS hebben de systemen de mogelijkheid om individueel te werken, maar vallen nog steeds onder de autoriteit van de SoS. Het komt vaak voor dat deze systemen allemaal individueel met verschillende stakeholders zijn gebouwd zonder enig idee van een SoS.

#### Acknowledged

Bij een acknowledged SoS is er een SoS autoriteit met de verantwoordelijkheid voor het halen van bepaalde doelen, zonder dat de systemen hier ondergeschikt aan zijn. De doelen die de systemen moeten behalen zijn gebaseerd op coöperatieve afspraken tussen de SoS en het systeem.

#### Collaborative

Een collaborative SoS heeft geen bovengeschikte autoriteit of een overkoepelend systeem, maar is gebaseerd op het onderling onderhouden van afspraken en doelen op een vrijwillige basis tussen de systemen. Deze afspraken zijn zo bedoeld dat ze een centrale oplossing hebben die voor ieder systeem voordelig is.

#### Virtual

Een virtuele SoS is het tegenovergestelde van een directed SoS. Een virtual SoS heeft geen centraal en bovengeschikte autoriteit of een overkoepelend systeem, maar werkt op de aanwezigheid van onderlinge overeenkomsten tussen de systemen via protocollen en standaarden. Je kunt het daarom een beetje vergelijken met het internet.

### 

### Wat zijn de best practices?

* Zorg ervoor dat de deelnemers in het SoS snappen dat het een SoS is en geen enkel systeem.
* Het aanmoedigen van een service georiënteerde blik op het SoS. Dit helpt met het begrijpen van wat de mogelijkheden zijn en zorgt voor een betere identificatie van opties voor het herconfigureren van het SoS als er een verandering binnen de systeem omgeving plaatsvindt.
* Begrijpen welk type SoS mogelijk is en welke autoriteiten hiervoor nodig zijn. Ook is het handig om te kijken waar de risk ownership ligt.
* Focus leggen op relaties tussen systemen. Dit is belangrijker dan weten wat er binnen elk systeem wordt gedaan.
* Zowel formele als informele relaties zijn meestal aanwezig binnen een SoS. Vertrouwen is daarom een belangrijk punt. Zonder vertrouwen is het onderhouden van informatie waarop keuzes zijn gebaseerd niet goed mogelijk.
* Het delen van informatie met andere deelnemers om ze bewust te laten zijn wat de situatie van de systemen zijn en te melden of bepaalde gebeurtenissen binnen een systeem invloed gaan hebben op een ander systeem.
* Begrijpen dat veel problemen met het systeem niet technisch zijn, denk aan sociaal, politiek of economisch. Het is belangrijk dat er op deze problemen de focus wordt gelegd omdat deze vaak de reden zijn tot het falen van een compleet systeem of onverwachts opkomende problemen.
* Goede en bijgewerkte documentatie over de architectuur. Dit is van belang zodat iedereen een goed beeld hiervan heeft en op basis hiervan er duidelijke beslissingen kunnen worden genomen.
* Bouw systemen op basis van modulaire architectuur zodat het implementeren en aanpassingen in een systeem gemakkelijker en sneller kunnen gaan.
* Maak gebruik van open standaarden om het risico op inter operationele problemen tussen de verschillende systemen te verminderen. Dit houd ook in dat systemen en oplossingen zoveel mogelijk moeten worden hergebruikt. Ook moeten de systemen zoveel mogelijk moeten worden getest zodat het risico op falen zo klein mogelijk blijft en de kosten beperkt blijven.

# Conclusie

Probleemstelling:

**Één middleware tech stack vinden dat het beste bij het project past dat wordt beschrijving in de case study.**

We hebben de termen System of systems, middleware tech en integration patterns onderzocht en we zien in hoe dit kan helpen bij het kiezen van middleware tech stack.

We hebben de kennis onderling afgestemd en onze bevindingen besproken. Dit is de zevende stap van de “seven step approach‘. Dit hebben we gedaan door met groepjes van 2 een vraag te beantwoorden en dan de bevindingen en conclusies in de groep te bespreken.

Dit heeft geleid tot de conclusie:

Om het probleem op te lossen om middleware tech stack te vinden die het best bij het project van de study case past op te lossen is niet mogelijk door het missen van context.

# Literatuurlijst

<https://www.mitre.org/publications/systems-engineering-guide/enterprise-engineering/systems-of-systems/treating-systems-of>

<https://www.enterpriseintegrationpatterns.com/patterns/messaging/IntegrationStylesIntro.html>

<https://searchapparchitecture.techtarget.com/definition/Object-Request-Broker-ORB>

<https://homepages.cwi.nl/~paulk/thesesMasterSoftwareEngineering/BasTerwijn.pdf>

1. <https://azure.microsoft.com/nl-nl/overview/what-is-middleware/> [↑](#footnote-ref-0)