Kommunikation mellan simulerad övning och utbildningsplattform

Pierre Aupeix
dat11pau@student.lth.se
Daniel Eckerström
dat14dec@student.lth.se

Department of Electrical and Information Technology
Lund University

Supervisor: Christin Lindholm

Examiner: Christian Nyberg

19 juni 2017

Printed in Sweden
Tryckeriet i E-huset, Lund

Sammanfattning

Följande examensarbete utreder SCORM, xAPI och Moodle Web Service API i syfte att användas som kommunikationskanal mellan utbildningsplattformen Moodle och Saabs träningssystem.

Avsikten var att använda Moodle som plattform för att lagra träningsdeltagare och deras resultat samt nyttja Moodles inbyggda funktionalitet för kurser och lärande. I utredningen ingick en analys av vilka kommunikationsmöjligheter som finns för Moodle som plattform, samt hur väl dessa passar in med befintliga träningssystem.

Analysen har resulterat i denna rapport, en utvecklad prototyp samt rekommendationer för framtida utveckling.

Prototyp
En webbapplikation, *SimLink*, har utvecklats som tillhandahåller kommunikation mellan Moodle och Saabs träningssystem. SimLink möjliggör generering av utvärderingsrapporter på en specifik simulerad övning, där användaranpassade resultat sparas i Moodle.

Rekommendationer till Saab
Ett resultat av examensarbetet är en fungerande prototyp där följande tas i beaktande vid fortsatt utveckling.

1. Moodle Web Service API är inte optimerat för en eventuell framtida produkt. En reviderad webbtjänst för Moodle bör utvecklas på grund av prestandaskäl.
2. En ny aktivitetstyp i form av ett Moodle-plugin kan med fördel utvecklas för enklare integration med befintliga och framtida system.
3. Eventuell omarbetning av befintligt träningssystems API till att möjliggöra generering av utvärderingsrapporter.

Nyckelord
Moodle, LMS, xAPI, TinCan, SCORM, AKKA

Abstract

The following bachelor thesis examines the possibility of using SCORM, xAPI or Moodle Web Service API as a communications channel between the learning management system Moodle and an existing training system delevoped by Saab.

The aim is to use Moodle for storing users and their results while using Moodles built in funcionality for e-learning. The thesis contains an analysis of the potential communication channels and their respective capability to communicate with the existing training system.

The analysis has resulted in this thesis, a prototype as well as recommendations for future development.

Prototype
A web application, *SimLink*, was developed to provide a communication channel between Moodle and Saabs existing training system. SimLink provides the ability to generate evaluation reports for a specific exercise where individualized reports are stored in Moodle.

Recommendations for Saab
The theisis resulted in a functioning prototype where the following has to be taken in consideration for future development.

1. Moodle Web Service API is not optimized with regards to performance. A bespoke web service should be developed.
2. A custom-made Moodle activity plugin would simplify the integration process for any future system.
3. Amendments to existing training systems API should be made to provide the ability to generate evaluation reports.

Keywords
Moodle, LMS, xAPI, TinCan, SCORM, AKKA

Innehållsförteckning

Kapitel 1

Inledning

1.1 Bakgrund

Saab AB är ett multinationellt företag med cirka 15 000 anställda som utvecklar produkter inom den militära domänen. Exempel på några av de produkter som utvecklats av Saab är kamouflagenät, vapensystem, stridsflyg och ubåtar.

Examensarbetet utfördes på Saabs kontor i Helsingborg som främst utvecklar produkter för träning i simulerad miljö. Utgångspunkten för arbetet var att undersöka möjligheterna till kommunikation mellan ett *Learning Management System* och Saabs träningssystem samt att utveckla en prototyp som möjliggör denna kommunikation.

1.2 Syfte

Examensarbetet undersöker möjligheten att digitalisera en hel utbildningsprocess, där de simulatorer och träningssystem som Saab tillhandahåller knyts samman med en utbildningsplattform.

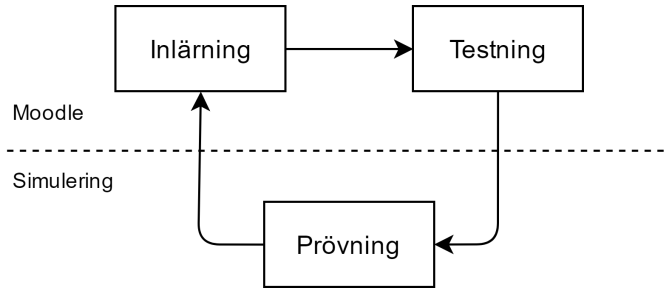
I dagsläget sker stegen i utbildningsprocessen till stor del med penna och papper. För att en individ ska genomgå ett träningsmoment krävs mycket manuell hantering av resurser. Ett träningsmoment ser ofta ut enligt följande:

Inlärning	Inlärning med hjälp av resurser såsom böcker och videoklipp.
Testning	Skriftligt prov på det inlärda.
Rättning	Manuell rättning av skriftligt prov.
Prövning	Praktiskt prövning av de inlärda kunskaperna.
Utvärdering	Manuell utvärdering av praktiskt prov.
Arkivering	Manuell arkivering av provresultat.

En digitalisering av ett träningsmoment möjliggör en automatisering av rättning, utvärdering och arkivering samtidigt som en gemensam portal kan användas för inlärning, testning och arkviering.

1.3 Målformulering

En digitaliserad utbildningsprocess kan delas in i tre delar: inlärning, testning samt prövning. Ett exempel på hur dessa delar samverkar i ett träningsmoment visas i figur 1.1.



Figur 1.1: Översiktlig bild av ett digitaliserat träningsmoment

Delmoment över den streckade linjen sker i Moodle varpå delmoment under den streckade linjen genomförs med hjälp av simulerad träning i Saabs träningssystem. Beskrivning av vad som sker i de olika delmomenten visas nedan.

Inlärning	Inlärning av resurser. Samtliga resurser finns sparade i Moodle.
Testning (Rättning)	Test av inlärda resurser genomförs i Moodle. Rättning sker automatiskt.
Prövning (Utvärdering) (Arkivering)	Simulerad övning där användarens resultat utvärderas. Utvärderade resultat arkiveras automatiskt i Moodle.

För att flödet i figur 1.1 ska fungera måste en kommunikationskanal mellan Moodle och simulering finnas.

Målet är att en prototyp ska utvecklas som tillåter kommunikation mellan Moodle och Saabs träningssystem. Resultat från övningar utvärderas varpå de utvärderade resultaten sparas i Moodle för att sedan analyseras i syfte att binda ihop hela utbildningsprocessen.

1.4 Problemformulering

Examensarbetet kommer att behandla följande frågeställningar:

1. Vilka kommunikationsmöjligheter finns det mellan Moodle och simulerad övning?
2. Vilka för- och nackdelar finns med de olika kommunikationsmöjligheterna?
3. På vilket sätt kan resultat från simulerade övningar sparas i Moodle?
4. Hur utvärderas dessa resultat?
5. Hur kan dessa resultat i sin tur påverka vidare simulerad övning?
6. Hur utvärderas resultat på grupp- och individnivå?
7. Hur kopplas en Moodleidentitet till en användare av en simulerad övning?

1.5 Motivering av examensarbetet

Efter att vi tidigare arbetat med Saab på ett annat projekt blev vi erbjudna möjlighet till examensarbete. Samarbetet vi hade var väldigt givande och en fortsatt samverkan var därför tilltalande. Vi tror också att utveckling av ett utbildningssystem både är intressant och utmanande.

1.6 Avgränsningar

På grund av säkerhetsmässiga och administrativa anledningar kommer examensarbetet inte att behandla någon form av implementation eller testning av kommunikation mellan faktisk simulerad övning och AKKA (avsnitt 2.2).

Arbetet kommer istället baseras på simulerad kommunikation mellan simulering och AKKA.

Kapitel 2
Teknisk bakgrund

Kapitlet innehåller den bakgrundsinformationen som krävs för att ge förståelse för de tekniker och produkter som använts och utvärderats.

Kapitlet är skrivet för läsare med någon datatekninsk bakgrund, där läsaren saknar kunskap om tekniker specifika för examensarbetet.

2.1 Moodle

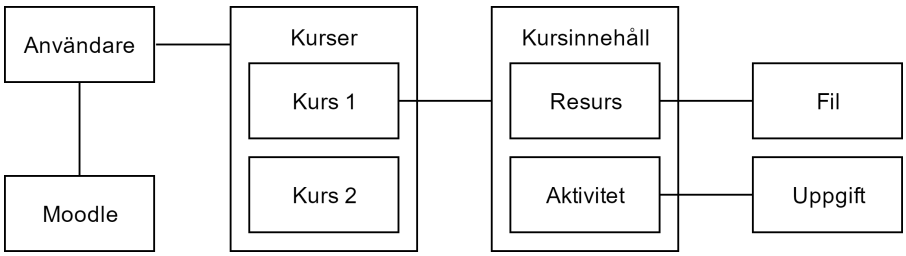
Moodle, *modular object-oriented dynamic learning environment* [20], är ett learning management system, *LMS*, skriven med öppen källkod. Användare interagerar med Moodle genom en webbportal [3].

Moodlesystemet är modulärt uppbyggt. Det finns möjlighet att lägga till funktionalitet i from av plugins. Plugins är av specifika typer. Fyra exempel på plugintyper är *teman*, *aktiviteter*, *resurser* och *språkpaket* [21].

2.1.1 Kurser

Kurser är den del av Moodle där lärare lägger till och ändrar innehåll som kursregistrerade studenter kan ta del av [22].

Ett exempel på hur en kurs kan se ut för en användare kan ses i figur 2.1.



Figur 2.1: Översiktlig bild av Moodles datamodell

En kurs kan innehålla aktiviteter och resurser, vad dessa är beskrivs i avsnitt 2.1.1.1 och 2.1.1.2.

2.1.1.1 Resurser

Lärare kan lägga till resurser i en kurs [19]. Det finns sju olika resurser i standardinstallationen av Moodle. Det finns möjlighet att lägga till resurstyper i form av plugins. Tre exempel på resurser som medföljer standardinstallationen är *file*, *page* och *URL*.

Resurser används för att utöka kursens innehåll med information.

2.1.1.2 Aktiviteter

Aktiviteter är till skillnad från resurser möjliga för studenter att medverka i och interagera med [18]. Det finns 14 olika aktiviter i standardinstallationen av Moodle. Det finns möjlighet att lägga till aktivitetstyper i form av plugins. Tre exempel på aktiviteter som medföljer standardinstallationen är *assignments*, *quiz* och *survey*.

Aktiviteter används för att ge kurser interaktivitet. Resultat för de olika aktiviteterna kan spåras av lärare.

2.2 AKKA

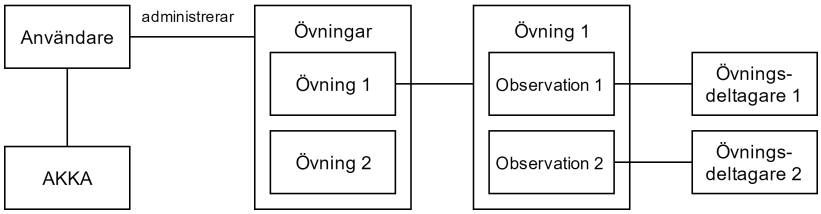
AKKA är en programvara utvecklad av Saab [40]. AKKA används för att skapa, lagra och analysera observationer.

En observation (se avsnitt 2.2.2) är ett uppmätt resultat för ett delmoment specificerat i en övning. Observationer kan komma från praktiska och simulerade övningar (se avsnitt 2.2.1).

Det finns två sätt att interagera med AKKA. Det finns en webbportal [42] genom vilken användare kan lägga till, ändra eller ta bort observationer. Det finns även ett externt gränssnitt som tillhandahåller funktionalitet för andra applikationer att lägga till, ändra och ta bort observationer. Det externa gränssnittet används av simuleringar för att lägga till automatiskt genererade observationer.

2.2.1 Övningar

En övning är den del i AKKA där användare eller simuleringar lägger till och ändrar observationer. Ett exempel på hur en övning kan se ut ses i figur 2.2.

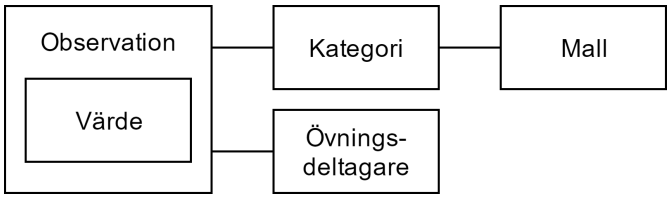


Figur 2.2: Översiktlig modell av en övning i AKKA

Varje observation tillhör en kategori och observerar en övningsdeltagare. Vad en observation, kategori och övningsdeltagare än beskrivs i avsnitt 2.2.2, 2.2.3 respektive 2.2.4.

2.2.2 Observation

En observation är resultatet av ett delmoment i en övning. Varje observation tillhör en kategori som i sin tur definierats enligt en mall. En observation innehåller också enligt kategorin avlästa mätvärden, se figur 2.3.



Figur 2.3: Översiktlig modell av en observation i AKKA

2.2.3 Kategori

En kategori används för att beskriva en observation. Kategorier definieras av mallar som beskriver hur mätdata ska tolkas. En mall kan användas av flera kategorier och en kategori kan användas av flera observationer [42].

2.2.4 Övningsdeltagare

En övningsdeltagare, *audience*, är den identitet eller användare som en observation är bunden till.

2.2.5 Trigger

AKKA tillåter användning av plugins [40]. En trigger är ett plugin som prenumererar på objekttyper och händelser som påverkar dessa objekttyper. Två exempel på objekttyper är en *nyhet* och en *observation*. En händelse är ändring, borttagning eller skapandet av en objekttyp.

En trigger som prenumererar på observationer kommer alltså att startas så fort en observation skapas, ändras eller tas bort.

2.3 SCORM

SCORM, *Sharable Content Object Reference Model*, är en standard utvecklad för att öka interoperabiliteten av resurser till LMS:er. SCORM består av två huvudsakliga delar, CAM samt RTE.

CAM *Content Aggregation Model*, definierar hur innehåll, tex kurser eller lektioner ska paketeras för att uppfylla standarden [4].

RTE *Run-Time Environment*, definerar hur SCORM-paketerade innehåll ska startas av, och kommunicera med en LMS [5].

2.4 xAPI

xAPI, *eXperience API*, även känd som *TinCan*, är en standard utvecklad för att tillhandahålla kommunikation av inlärningsdata i form av statements mellan olika plattformar.

Statements är definierade enligt xAPI:s Statement Object Model, där strukturen av ett statement beskrivs nedan [7].

Actor	vem som åsyftas av statementet
Verb	definierar händelsen mellan *Actor* och *Object*
Object	definierar vad eller vem *Actor* interagerade med

2.4.1 LRS

LRS, *Learning Record Store*, är del av xAPI:s implementation. LRS:en tar emot, sparar och skickar den informationen som överförs mellan plattformar som använder xAPI [6].

LRS:en är även den del av xAPI:s implementation som validerar överföringens innehåll gentemot den specificerade datamodellen [7].

2.5 LMS

LMS, *Learning Management System*, definieras som en applikation vars huvudsakliga syfte är att digitalisera administrering, tillhandahållning och uppföljning av lärande [1]. Ett exempel på ett LMS är Moodle. Svenska synonymer är lärplattform samt utbildningsplattform [2].

2.6 REST

REST, *Representational State Transfer*, är en standard som beskriver hur en Web Service (se avsnitt 2.10) kan implementeras, implementationer som följer REST kallas även för RESTful.

En av restriktionerna som REST-standarden beskriver är att överföringen ska vara stateless, detta innebär att varje request ska vara oberoende av någon sparad kontext på servern [8].

2.7 Node.js

Används för exekvering av Javascript utanför webbläsare. Möjliggör användning av Javascript för att utveckla webbserverapplikationer [31].

2.8 Vue.js

Vue.js är ett ramverk för utveckling av användargränssnitt för webbapplikationer [9].

2.9 CORS

I enlighet med *Same-origin policy* begränsar webbläsare ett skripts möjlighet att ta del av information från en annan källa än skriptets ursprungskälla, för att kringå detta beteende kan CORS användas.

CORS, *Cross-origin resource sharing*, är en standard som tillåter ett webskript att ta del av information från flera källor. CORS möjliggör att specifika *headers* läggs till HTTP-förfrågningar där skriptets källa finns deklarerad, detta ger mottagaren möjlighet att godkänna eller avvisa förfrågan baserat på om källan är pålitlig eller ej [23].

2.10 Webbtjänst (Web Service)

En tjänst för exponering av applikationsfunktioner och resurser i syfte att göra det åtkomligt för externa applikationer över ett nätverk [34].

2.11 PHP

PHP, *PHP: Hypertext Preprocessor*, är ett skriptspråk utvecklat med öppen källkod. PHP används mestadels för dynamisk generering av webbsidor [35].

Kapitel 3
Metod

3.1 Arbetsprocess

Arbetsprocessen kan delas in i tre delar, informationsinhämtning, undersökning samt implementation. En tydlig indelning av arbetsprocessen är dock svår att göra då delarna har överlappats under utvecklingen. Dagbok fördes under arbetets gång vilket underlättade såväl vid utveckling som vid rapportskrivning.

3.1.1 Informationsinhämtning

Informationsinhämtning har skett kontinuerligt under arbetets gång. All inhämtad information var från webbsidor eller webbdokument förutom information om AKKA [39][40][41][42] samt information från boken *Interaction Design: beyond human-computer interaction* [36]. Under den initiala informationsinhämtningen studerades främst standarderna xAPI och SCORM som kandidater för kommunikation mellan Moodle och extern applikation.

Informationsinhämtningen övergick senare till att även innefatta Moodles egna Web Service.

3.1.2 Undersökning

Ett mål för informationsinhämtningen var att undersöka standarderna xAPI och SCORMs förmåga att kommunicera mellan Moodle och extern applikation.

För att svara på frågeställningen *Vilka för- och nackdelar finns med de olika kommunikationsmöjligheterna?* (se avsnitt 1.4) togs fyra kriterier fram. Dessa fyra kriterier påvisar vilka för och nackdelar som finns med de olika kommunikationsmöjlighterna sett till projektets målbild. Dessa kriterier beskrivs i avsnitt 3.1.2.1.

Frågeställningen besvarades genom undersökning av SCORMs [5] respektive xAPIs [13] dokumentation. Undersökningen övergick senare till att även innefatta Moodles egna webbtjänst [16].

3.1.2.1 Kriterier

Nedan visas samtliga kriterier och varför de är väsentliga för att uppfylla målbilden.

Hämta data om specifik användare
För att kunna mappa en användares unika identifierare i Moodle med en övningsdeltagare i AKKA på måste det vara möjligt att hämta data om en specifik användare.

Hämta betygstatus på tidigare aktivitet
För att kunna avgöra om en användare ska få utföra en simulerad övning eller ej måste det vara möjligt att hämta betygstatus från tidigare, av användare, utförda aktiviteter.

Ändra betyg för specifik användare
För att kunna automatisera betygsättningen av en simulerad övning måste det vara möjligt att ändra betygstsatus på en aktivitet i Moodle för en specifik användare.

Hämta konfigurationsdata för simulering från Moodle
För att kunna ändra inställningar i simulerad övning måste det vara möjligt att hämta konfigurationsdata för simulering från aktivitet i Moodle.

3.1.3 Implementation

Implementationsdelen av arbetsprocessen syftar till utveckling av prototyp. Likt *Hi-fi prototyping* [36] utvecklades prototypens funktionalitet stegvis tills full funktionalitet uppnåddes, varpå prototypen övergick till slutprodukt.

Utvecklingen skedde med hjälp av parprogrammering, dels av praktiska skäl då AKKA endast fanns att tillgå lokalt på en dator, dels för att minska antal fel i koden [33].

3.2 Källkritik

Samtliga källor delas in i fem kategorier, *Moodle*, *Akka*, *SCORM*, *xAPI* och *övrigt*. Indelning baseras på trovärdigheten hos källorna och hur väl äktheten går att verifiera.

3.2.1 Moodle

Moodle är programvara skriven med öppen källkod. Samtliga påståenden om funktionalitet kan därför verifieras genom att läsa den källkod som finns tillgänglig för alla [3].

All Moodlefunktionalitet vilken berörs i denna rapport och/eller användes av examensarbetets utvecklade prototyp har varit konsistent med listade källor gällande Moodle.

3.2.2 AKKA

Källor för AKKA består av Saabs interna dokumentation samt samtal med Saab-utvecklare ansvariga för AKKAs programvara.

Använd dokumentation för AKKAs API täcker version 3.0 av AKKAs programvara. Den version av AKKA som används i examensarbetet är version 4.3. På grund versionsskillnaden av dokumentation och programvara var vissa delar av dokumentationen inte användbara för examensarbetet.

För att lösa de problem som uppstod vid avsaknandet av uppdaterad dokumentation gjordes en avstämning med ansvarig utvecklare vilket garanterade korrekthet.

3.2.3 SCORM

Informationen om SCORM grundar sig på två källor, Rustici Software och ADL. ADL är ett statligt projekt som lyder under USA:s försvarsdepartemente (Department of Defence). Då ADL är ett statligt projekt kan antagandet göras att projektet blir utsatt för revisioner, vilket borde stärka informationens korrekthet.

Rustici Software är ett amerikanskt företag som bland annat implementerat xAPI på beställning av ADL, som underleverantör åt ADL kan antagandet göras att Rustici Software har liknande krav vad gäller korrekthet som ADL.

3.2.4 xAPI

Informationen om xAPI grundar sig, likt SCORM, på källorna ADL och Rustici Software. Samma antagande om korrekthet kan därför göras. xAPI bygger på öppen källkod vilket underlättar verifikation.

3.2.5 Övrigt

Källor vilka refererar till GitHub och andra portaler för kodversionshantering kan anses vara trovärdiga då samtliga påståenden om funktionalitet kan verifieras genom att läsa den källkod som finns tillgänglig för alla.

3.3 Verktyg

Under utvecklingen av prototypen har flera verktyg använts, nedan listas vilka.

PhpStorm Integrerad utvecklingsmiljö för PHP [24]. Har främst använts vid den initiala utvecklingen av prototypen. Då Moodle är skrivit i PHP och vi till en början gjorde en egen Web Service var PHP nödvändigt som utvecklingsspråk.

NPM *Node Package Manager*, Pakethanterare för Node [30]. Har använts under utvecklingen av SimLink för lätt hantering och installation av ytterligare JavaScript-paket.

Atom Textredigeringsprogram med möjlighet att installera insticksmoduler [25]. Användes främst vid utveckling av SimLink som beskrivs i avsnitt 4.6.

Visual Studio Integrerad utvecklingsmiljö från Microsoft [28]. Visual Studio 13 användes vid utveckling av trigger, se avsnitt 2.2.5

WampServer Verktyg för webbutveckling i Windowsmiljö [27]. Har använts för att låta det lokala datorsystemet vara värddator åt PHP-implementationen av SimLink.

Draw.io Webbaserat ritverktyg [29]. Användes vid framställning av diagram till examensarbetets rapport.

ShareLaTex Textredigeringsprogram [26]. Användes vid framställning av examensarbetets rapport.

Chrome Webbläsare utvecklad av Google [32]. Chrome DevTools användes för anlys av HTTP-förfrågningar, debuggning och tolkning av stackspår vid fel.

3.4 Kommunikation

Då examensarbetet utförts av två personer verksamma brevid varandra på Saabs kontor har förutsättningarna för kommunikation varit goda.

Kommunikationen med kund skedde företrädesvis under samtal med handledare på Saabs kontor, även E-post nyttjades.

Kapitel 4
Analys

Syftet med detta kapitel är att ge en mer ingående bild över informationsinhämtningen och utvecklingsarbetet, vilka val som gjorts under utvecklingens gång och varför.

4.1 Målbild

Målbilden förändrades under examensarbetets gång på grund av erfarenheter som gav insikt om felaktiga antaganden och begränsningar med målbilden. All analys gjordes med utgångspunkten att på bästa och enklaste sätt lösa problemenen i avsnitt 1.4.

4.1.1 Initial målbild

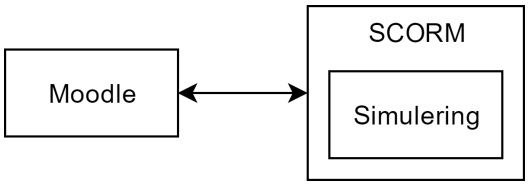
Målet med projektet var delvis att undersöka möjligheter till kommunikation mellan Moodle och en extern simulering. Den initiala målbildens syfte var att ge en övergripande bild över hur informationen flödade i systemet. Antaganden som gjordes baserades på en begränsad mängd kunskap.

Figur 4.1: Initial målbild.

Antagandet gjordes att både SCORM och xAPI var protokoll för kommunikation mellan LMS:er och externa resurser. Tanken var att en tolk skulle översätta informationsflödet mellan Moodle och extern simulering.

4.2 Utvärdering av SCORM

Först utvärderades SCORM, vilka möjligheter och brister fanns med standarden och på vilket sätt kunde den användas för att implementera en utbildningsplattform.



Figur 4.2: Översiktlig bild av en lösning med SCORM.

4.2.1 Felaktigt antagande

Antagandet hade tidigare gjorts att SCORM är ett kommunikationsprotokoll mellan LMS:er och externa resurser.

Efter analys av SCORM observerades det att SCORM är en standard över hur kurser kan paketeras för bättre kompatibilitet mellan olika utbildningsplattformar. Även om viss funktionalitet finns för informationsutbyte är SCORM inte ett kommunikationsprotokoll [10].

4.2.2 Fördelar med SCORM

Vid användning av SCORM-paketerade lösningar finns möjlighet till begränsad kommunikation mellan LMS och extern simulering.

Standarden möjliggör en viss typ av informationsutbyte mellan Moodle och extern simulering [11]

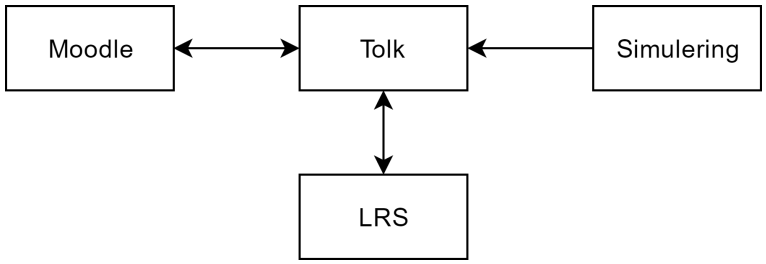
4.2.3 Brister med SCORM

Funktionaliteten som tillhandahåller kommunikationsmöjligheter mellan LMS och extern programvara förutsätter att den externa programvaran är paketerad som ett SCORM-paket, samt att det använda LMS:et har möjlighet att tolka den information som skickas. Detta medför att om SCORM ska användas som kommunikationskanal bör den utbildningsplattform som används modifieras för att hantera informationen.

Ett senare tillkommande krav från Saabs sida var att simuleringen inte bör startas från Moodle. Då ett SCORM-pakets kommunikation är beroende av att det startas av en LMS, försvåras om inte omöjliggörs en implementation där SCORM-standarden används som kommunikationskanal mellan extern simulering och Moodle [12][11].

4.3 Utvärdering av xAPI

Granskning av xAPI gjordes för att fastställa vilka möjligheter och brister det fanns med standarden och på vilket sätt den kunde användas för att implementera en utbildningsplattform.



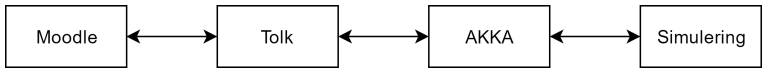
Figur 4.3: Översiktlig bild av en lösning men xAPI.

4.3.1 Fördelar med xAPI

xAPI kan tillhandahålla kommunikation mellan LMS och extern simulering. Kommunikationen görs med xAPI:s Web Service där data i form av statements skickas till och från en obligatorisk LRS. Statements är uppbyggda på så sätt att de tillåter detaljerad information om händelseförlopp. En LRS möjliggör långtidslagring av dessa statements. xAPI:s standard tillåter också att påbörja ett moment på en plattform för att sedan ha möjligheten att fortsätta på en annan plattform utan avbrott [13][14].

4.3.2 Brister med xAPI

En detalj vilken Saab tidigare inte nämnt var hur den externa simuleringen kommunicerade utåt. Saab använder sig av ett system som lagrar alla händelser under en simulerings gång för att sedan tillhandahålla möjligheter att extrahera dessa. Händelserna liknar till strukturen de statements som xAPI använder sig av. En implementation med hjälp av xAPI ansågs därför som en överflödig lösning på grund av överlappande funktionalitet [15].



Figur 4.4: Målbild med AKKA inkluderad.

Flödet i systemet antogs efter dessa insikter likna det i figur 4.4

4.4 SCORM och xAPI som lösning

Både SCORM och xAPI hade brister och fördelar som hade påverkat hur väl målbilden uppfylldes.

En implementation där SCORM används som kommunikationskanal mellan Moodle och extern simulering skulle delvis innebära att någon form av inkapsling av extern simulering måste göras. Det hade också inneburit ett krav att externa simuleringar måste startas från Moodle. Det är det enda sättet SCORM-inkapslat material kan kommunicera tillbaka till ett LMS.

Efter utvärdering av SCORM ansågs lösningen bristfällig och möjlig implementation hade inte uppfyllt målbilden tillräckligt.

En implementation där xAPI används som kommunikationskanal mellan Moodle och extern simulering skulle innebära ett krav på användning av en LRS.

Efter utvärdering av xAPI ansågs en implementation med viss kommunikationsmöjlighet möjlig, däremot var behovet av en LRS ett problem då detta var funktionalitet som Saab redan hade system för.

4.4.1 Alternativt användningsområde för SCORM

I samband med analys av Moodles integrering av SCORM hittades ett alternativt område för SCORM-standarden. SCORM-paketerade delmoment kan importeras av Moodle vilket skulle innebära en möjlighet för Saab att låta underleverantörer skapa delmoment för att sedan importera dessa.

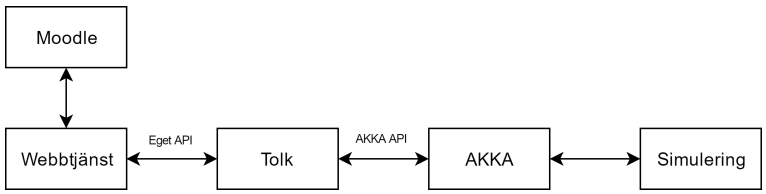
4.5 Utvärdering av Moodle API

Resultatet från utvärderingen av xAPI och SCORM visade att ingen av standarderna passade kundens målbild, därför undersöktes möjligheten att använda Moodles Web Service API.

4.5.1 Egen Web Service

På grund av bristfällig dokumentation och missförstånd om hur Moodle hanterade web services gjordes först ett lokalt tilläggsprogram till Moodle. Tanken var att låta tilläggsprogrammet hantera den funktionalitet tolken krävde från Moodles sida. Till exempel behöver tolken ha tillgång till vilka aktiva kurser en användare har, huruvida någon av dessa kurser innehåller något moment där en extern simulering krävs [16].

Då tolken behövde tillgång till funktionalitet i det lokala tilläggsprogrammet exponerades funktionaliteten i from av en web service. Det insågs dock att tilläggsprogrammet fungerade som ett eget API för Moodle.

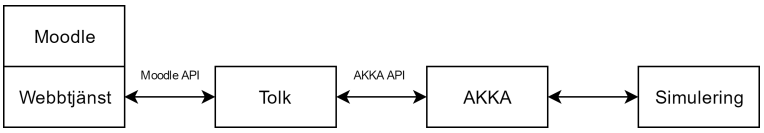


Figur 4.5: Översiktlig bild av lösning med egen Web Service.

Efter vidare undersökning hittades dokumentation över Moodles Web Service-API. Vilket vid första anblick hade samma (och utökad) funktionalitet som tilläggsprogrammet.

4.5.2 Moodle Web Service

Analys av Moodles Web Service-API gjordes med en Web Service-klient skriven i PHP, denna klient använder sig av funktioner från Moodles Web Service-API genom HTTP-förfrågningar med protokollet XML-RPC [17].



Figur 4.6: Översiktlig bild av lösning med Moodles Web Service.

4.6 SimLink

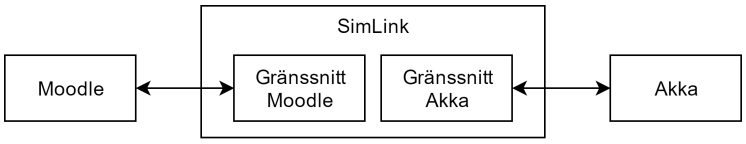
Tolken, vilken syns i figurerna 4.1 - 4.6, refereras hädanefter som SimLink.

Syftet med SimLink var att möjliggöra kommunikation mellan Moodle och AKKA, detta genom användandet av Moodles respektive AKKAs externa API.

4.6.1 Syfte

Då utvärderingen av Moodle Web Service-API (avsnitt 4.5.2) resulterat i en PHP-applikation med kommunikationsmöjligheter mot Moodle fortsattes utvecklingen på samma applikation.

Syftet var att utöka funktionalitet för att tillåta kommunikation med AKKAs externa API. Målet var alltså att i SimLink sammanfoga gränssnitt för kommunikation med Moodle samt gränssnitt för kommunikation med AKKA.



Figur 4.7: Översiktlig bild av SimLink.

4.6.2 Initial prototyp

Den första versionen av SimLink baserades på PHP-kod skriven för testning av Moodle Web Service-API. Koden utnyttjar XML-RPC som protokoll för kommunikation med Moodle. Protokollet använder sig av XML-formaterad data för kommunikation.

Moodles Web Service API tillåter användning av REST, SOAP och XML-RPC som protokoll [17].

AKKAs API följer REST-standard med JSON som dataformat. För att slippa konvertera mellan olika dataformat gjordes valet att skriva om SimLink till att istället utnyttja REST med Json-formaterad data.

4.6.2.1 Problem med initial prototyp

Brister med den initiala prototypen var avsaknandet av ett användargränssnitt samt oförmågan att analysera de HTTP-förfrågningar som skickades. För att lösa detta porterades SimLink till en Node.js-applikation med ett användargränssnitt skrivet i Vue.js, se avsnitt 4.6.4.

4.6.3 Representation av övning i Moodle

För att göra en koppling mellan simulerad övning och Moodle måste en övning vara representerad i Moodle. Moodle har 14 typer av aktiviteter (se avsnitt 2.1.1.2) av vilka ingen representerar en simulerad övning. Ett alternativ var att utveckla en skräddarsydd aktivitet för att mer specifikt representera en simulerad övning.

Valet gjordes att utnyttja aktiviteten *Assignment* för att representera en övning, då det finns möjlighet att ladda upp och ned filer till och från aktiviteten. Upp- och nedladdning av filer möjliggör lagring och hämtning av till exempel scenariodata för simulering eller kriterier för utvärderingsgenerering (se avsnitt 4.7.1).

4.6.4 Utveckling

SimLink utvecklades som en Node.js-applikation. Ett användargränssnitt gjordes i Vue.js för att enklare testa de funktioner som utvecklades. Node.js tillåter möjlighet att utöka funktionalitet genom import av olika JavaScriptbibliotek.

4.6.4.1 Importerade bibliotek

Följande Javascriptbibliotek importerades i applikationen.

Axios tillhandahåller HTTP-funktionalitet och användes i SimLink för att skicka HTTP-förfrågningar till både Moodle och AKKA

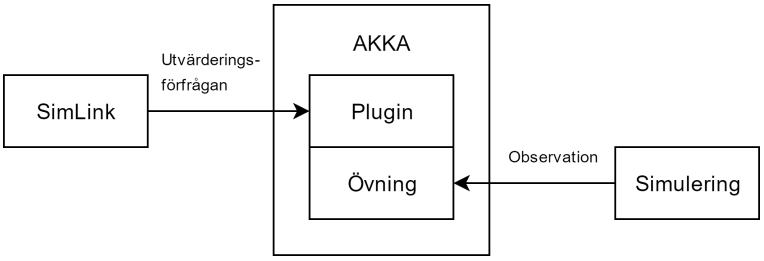
Lodash tillhandhåller funktionalitet för manipulation av objekt och vektorer och användes i SimLink för att hantera de svar som skickats från både Moodle och AKKA

ObjTree tillhandahåller funktionalitet för konvertering mellan XML och Json-objekt och användes i SimLink för att hantera den XML som AKKA skickar som utvärdering.

4.7 AKKA

AKKA är en applikation som samlar och analyserar information från övningar och simuleringar via inmatade observationer. Dessa observationer kan matas in direkt av en simulering eller via observatörer på fältet.

AKKAs Web API [39] tillhandahåller kommunikation med extern programvara, i detta fall SimLink.



Figur 4.8: Översiktlig bild AKKA.

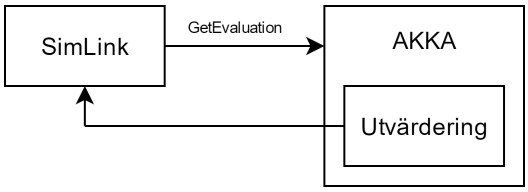
4.7.1 Utvärderingsresultat

AKKA kan vid förfrågan skicka ett utvärderingsresultat i XML-format specifikt för given övning. Resultatet är genererat utifrån de observationer som simuleringen har matat in.

Dock observerades det att den utvärdering som hämtas först måste genereras via ett plugin. Detta resulterade i en lösning där en trigger användes för generering varpå en utvärderingsförfrågan kan skickas och ett utvärderingsresultat ges som svar.

4.7.1.1 Utvärderingsförfrågan

I dess enklaste form skickas en utvärderingsförfrågan från SimLink till AKKA, responsen från AKKA består av en utvärderingsrapport.

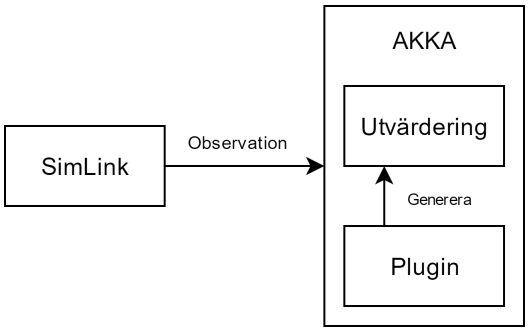


Figur 4.9: Översiktlig bild av utvärderingsförfrågan.

4.7.1.2 Generering av utvärdering

Om ingen tidigare utvärdering är gjord för den specifika övningen måste SimLink trigga en generering av utvärderingsrapporten, detta görs genom att från SimLink

mata in en observation som fångas upp och tolkas av en trigger. I den inmatade observationen finns information om vilken övning och användare som ska utvärderas. Utvärderingen görs enligt mallar bestämda i triggern.



Figur 4.10: Översiktlig bild av utvärderingsgenerering.

När utvärderingen genomförts sparas utvärdering tillsammans med resultat i AKKA och kan på så vis nås genom en utvärderingsförfrågan enligt sektion 4.7.1.1

Kapitel 5
Resultat

Examensarbetet har undersökt vilka kommunikationsmöjligheter som finns mellan Moodle och simulerad övning. Resultatet av undersökningen analyserades i syfte att hitta en passande kommunikationskanal för prototypen

Följande kapitel behandlar resultatet av den undersökning som gjorts, resultatet av den analys som gjordes av undersökningen samt en rekommendation för framtida utveckling.

5.1 Undersökning

En undersökning, se kaptel 3, gjordes av de kommunikationsmöjligheter som finns för Moodle samt hur väl dessa uppnår målbilden för examensarbetet.

Resultatet av undersökningen summeras i tabell 5.1.

	Moodle API	SCORM	xAPI
Hämta data om specifik användare	JA	JA	NEJ
Hämta betygstatus på tidigare aktivitet	JA	NEJ	JA
Hämta konfigurationsdata för simulering från Moodle	JA	NEJ	NEJ
Ändra betyg för specifik användare	JA	JA	NEJ

Tabell 5.1: Tabell över förmågor per protokoll.

Detaljerad förklaring av resultatet i tabell 5.1 kan ses i avsnitt 5.1.2, 5.1.4 samt 5.1.3.

5.1.1 Hämta data om specifik användare

Att hämta data om en specifik användare innebär att hämta ett användarobjekt från Moodle. Användarobjektet innehåller information om användaren, till exempel användar-ID och användarnamn. Syftet med datan är att mappa en användares unika identifierare i Moodle med en övningsdeltagare i AKKA.

xAPI Stödjer inte hämtning av användarobjekt från Moodle.

SCORM SCORM:s RTE tillhandahåller metoder för hämtning av användar-ID, vilket är tillräckligt för att koppla Moodle-användare med övningdeltagare i AKKA.

Moodle API Moodles API tillhandahåller funktionalitet för hämtning av användarobjekt.

5.1.2 Hämta betygstatus på tidigare aktivitet

Då en simulering kan vara beroende av förkunskaper hos tidigare utförda uppgifter undersöktes möjligheten att kontrollera betygstatus för tidigare uppgifter i Moodle. En uppgift är i Moodle representerad av en aktivitet (se avsnitt 2.1), dennes betygstatus hämtas för att fatta beslut om användaren ska utföra simulering eller ej.

xAPI xAPI är ett protokoll för kommunikation till och från en LRS. Om en LRS innehåller information om betygsättnig av föregående aktiviteter går denna information att hämta. Det finns däremot ingen möjlighet att med xAPI hämta informationen från Moodle.

SCORM SCORM-resurser har möjlighet att kommunicera med en LMS vid Run-Time. Kommunikationen styrs av SCORM:s RTE som tillhandahåller metoder för hämtning av såväl specifik användare samt status på avklarad aktivitet.

SCORM-resurser har inte tillgång till andra SCORM- och Moodleresursers information.

Moodle API Moodles API tillhandahåller API-anrop för hämtning av status för specifik användares tidigare aktiviteter.

5.1.3 Hämta konfigurationsdata för simulering från Moodle

Konfigurationsdata är data vars syfte är att ändra en simulerings inställningar, till exempel svårighetsgrad för simulerat delmoment.

Datan kan lagras på fil i en aktivitet, eller som fält i en skräddarsydd aktivitet.

xAPI xAPI stödjer inte hämtning av fil eller fält från skräddarsydd aktivitet direkt från Moodle.

SCORM SCORM:s RTE tillhandahåller inte metoder för att hämta filer från Moodle, ej heller metoder för att hämta fält i skräddarsydda aktiviteter.

Moodle API Moodles API tillhandahåller metoder för hämtning av filer.

5.1.4 Ändra betyg för specifik användare

Ändra betyg för specifik användare innebär att sätta betyg på specifik användare baserat på resultat från AKKA.

I Moodle finns två möjliga betyg, godkänd och icke godkänd. Betyg kan ändras direkt mellan godkänd och icke godkänd, eller via ändring av poäng på en aktivitet som när betygskriteriet uppnås blir godkänd.

xAPI Det finns ingen inbyggd funktionalitet i protokollet som möjliggör ändring av betyg. Det går däremot att med hjälp av ett Moodle-plugin hämta en användares statements för en specifik aktivitet från en LRS. Detta möjliggör ändring av betyg i Moodle baserat på dessa hämtade statements.

SCORM SCORM:s RTE tillhandahåller metoder för ändring av data hos LMS:er, dessa innefattar ändring av såväl poäng som betyg för genomförd aktivitet.

Moodle API Moodles API tillhandahåller funktionalitet för ändring av poäng, vilket i sin tur ändrar betyget i Moodle.

5.1.5 Sammanfattning av resultatet

Resultatet av undersökningen påvisade stora skillnader mellan xAPI, SCORM och Moodle API.

Eftersom målet med prototypen var att möjliggöra kommunikation mellan Moodle och en extern simulering visade sig en implementation med xAPI vara komplicerad. Detta dels då xAPI kräver en LRS för lagring och vidarebefordring av

statements och dels för att statements i sig introducerar ytterligare en datamodell som kräver tolkning mellan både AKKA och Moodle.

En implementation med SCORM visade sig inte nå målbilden för examensarbetet, detta då kommunikationen mellan SCORM och Moodle var begränsad till SCORMs RTE som inte tillhandahöll funktioner för att hämta betygstatus på tidigare aktivitet eller konfigurationsdata för simulering.

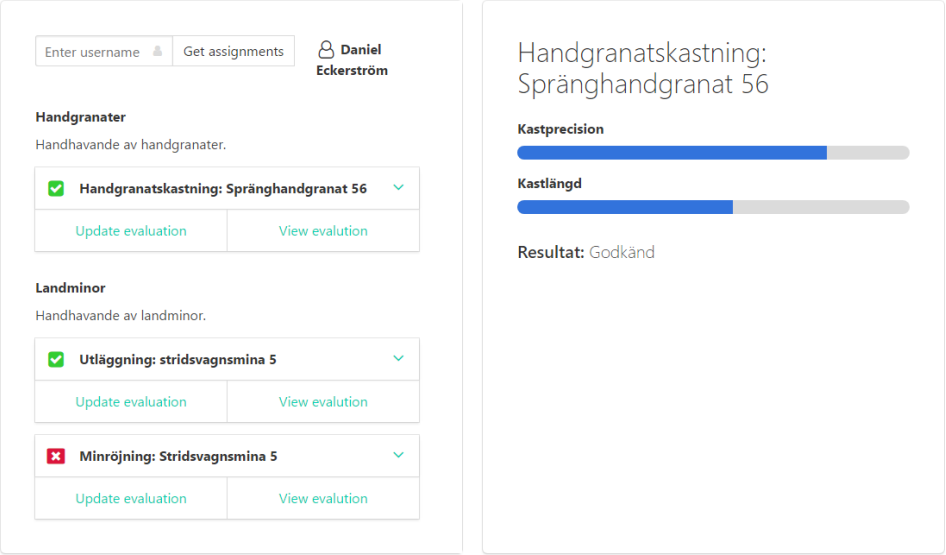
Moodles API visade sig uppfylla de krav på kommunikation som examensarbetet ställde och valdes därför som kommunikationskanal för prototypen.

5.2 Prototyp

Den färdiga prototypen, *SimLink*, är en webbapplikation utvecklad i Node.js samt en proxy utvecklad i PHP. Tillsammans tillhandahåller webbapplikationen och proxyn kommunikation mellan Moodle och AKKA.

SimLink möjliggör generering av utvärderingsrapporter på en specifik simulerad övning i AKKA, där användaranpassade resultat sparas i Moodle. Användning sker via ett användargränssnitt utvecklat i Vue.js.

Ett exempel på användning av SimLink visas i figur 5.1.



Figur 5.1: SimLink vid visning av utvärderingsresultat

I figur 5.1 visas ett tillstånd i SimLink vid användning. Kurser, delmoment och delmomentens resultat för användaren `Daniel Eckerström` visas på vänser sida. Utvärderingsresultat för valt delmomentetet visas på höger sida. I detta fall visas delmomentet `Handgranatskastning: Spränghandgranat 56`.

En vidare förklaring av hur SimLink används kan ses i avsnitt 5.2.1.

5.2.1 Händelsekedja

Användning av SimLink kan beskrivas med en händelsekedja.

1. **Inmatning av användarnamn**

   Användaren med det specifika användarnamnet hämtas från Moodle. För- och efternamn för användaren visas. Om ingen användare med inmatat användarnamn hittas i Moodle visas ett felmeddelande.

2. **Visning av användares kurser**

   Användarens kurser med en eller flera simuleringsaktiviteter hämtas från Moodle. Kursnamn, kursbeskrivning, namn på simuleringsaktivitet och betyg för varje simuleringsaktivitet visas. Finns ingen kurs med simuleringsaktiviteter visas ett felmeddelande.

3. **Utvärdering av simuleringsaktivitet för användare**

   Vid val av utvärdering av en simuleringsaktivitet skickas en utvärderingsförfrågan för specifik användare och simuleringsaktivitet till AKKA. En trigger tar emot förfrågan och ett utvärderat resultat sparas i AKKAs databas. Finns inte den specifika användaren eller simuleringsaktiviten i AKKA visas ett felmeddelande.

4. **Visning av utvärderingsresltat**

   Vid val av visning av utvärderingsresultat hämtas utvärderat resultat från AKKA. Information om resultatet visas. Om ingen utvärdering för den specifika simuleringsaktiviteten gjorts innan visning av utvärderingsresultat, visas ett felmeddelande.

5. **Uppdatering av användares betyg för simuleringsaktivitet**

   Vid lyckad visning av utvärderingsresultat uppdateras simuleringsaktivitetens betyg i Moodle och uppdaterat resultat visas.

En bildsekvens över beskriven händesekedja visas i appendix A.

5.2.2 System

SimLink ingår som en del i ett större system. Hela systemet består av fyra delar vilka listas nedan.

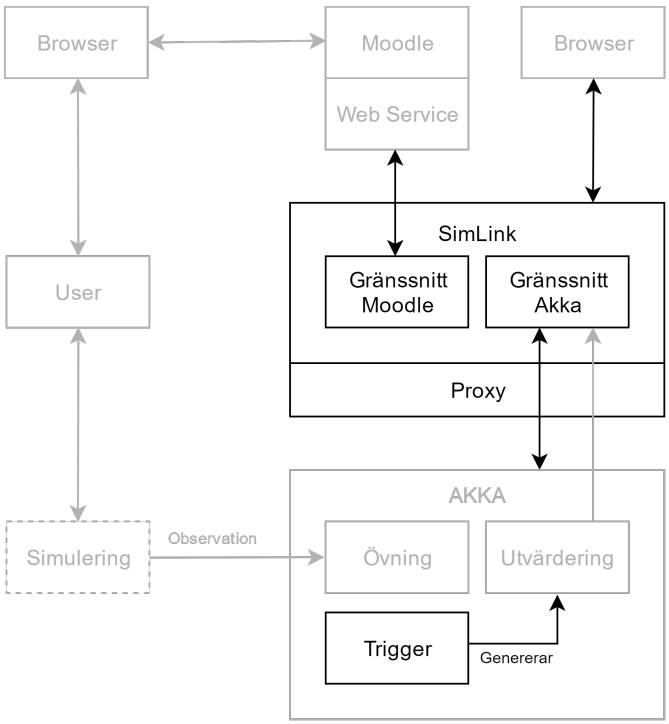
Moodle Består av en Moodle-portal tillgänglig via webbläsare. Inställningar gjordes för att möjliggöra kommunikation via Moodle Web Service API.

SimLink Består av den utvecklade webbapplikationen samt proxy. Beskrivs vidare i avsnitt 5.2.4.

AKKA Består av AKKA samt en utvecklad trigger för generering av utvärderingsresultat.

Simulering Ingår i systemet men har varken analyserats eller testats och berörs därför inte i denna rapport.

En övergripande bild över hur systemet hänger ihop kan ses i figur 5.2. Delar vilka inte utvecklats under examensarbetets gång visas i grått. Simuleringsdelen vilken inte ingår i rapporten visas streckad.



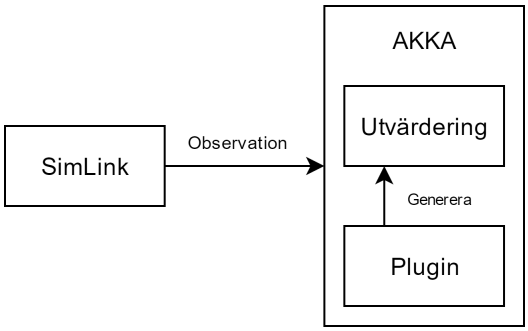
Figur 5.2: Översiktlig bild av systemet i helhet

Vidare beskrivning av SimLink och Trigger ges i avsnitt 5.2.4 respektive 5.2.3.

5.2.3 Trigger

En trigger utvecklades som en plugin till AKKA. Triggern tillhandahåller funktionaliteten för steg 3 i händelsekedjan som visas i avsnitt 5.2.1.

Triggern (se avsnitt 2.2.5) fångar upp och tolkar en uppdatering av en utvärderingsobservation. En utvärderingsobservation är den observation som uppdateras med information om vilken användare och vilken övning som ska utvärderas.

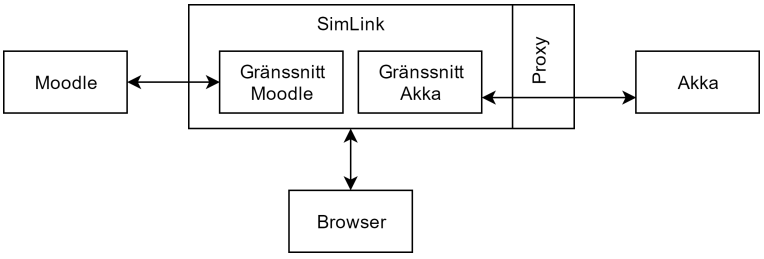


Figur 5.3: Bild av hur triggern genererar en utvärdering

I triggern finns kriterier för hur varje övnings observationer ska utvärderas. Efter en utvärdering är gjord sparas utvärderingsresultatet i AKKA och görs tillgänglig för hämtning via AKKAs API.

5.2.4 SimLink modell

Nedan visas en översiktlig bild av strukturen för SimLink. Funktionaliteten av applikationen styrs genom ett användargränssnitt i en webbläsare. Kommunikation till och från AKKA görs genom en proxy, se avsnitt 5.2.4.3.



Figur 5.4: Översiktlig bild av SimLink.

5.2.4.1 Gränssnitt Moodle

Gränssnittet mot Moodle tillhandahåller kommunikation med Moodle Web Service API. Samtliga dataförfrågningar baseras på ett inmatat användarnamn. Utifrån inmatat användarnamn hämtas användaren, användarens kurser, användarens aktiviteter i hämtade kurser samt dess resultat.

Den enda datan som modifieras på Moodleplattformen via Moodlegränssnittet är betyg i en specifik aktivitet för en specifik användare.

5.2.4.2 Gränssnitt AKKA

Gränssnittet mot AKKA tillhandahåller kommunikation med AKKAs API. Då SimLink porterats till en Node.js-applikation kunde HTTP-förfrågningar mot AKKA enklare analyseras. Efter analys av förfrågningarna gjordes insikten att AKKA inte stöder CORS, se avsnitt 2.9.

Begränsningen innebar att inga HTTP-förfrågningar från webbapplikationen togs emot av AKKA. För att kringgå begränsningarna med CORS utvecklades en proxy (se avsnitt 5.2.4.3) genom vilken samtliga HTTP-förfrågningar slussades.

5.2.4.3 Proxy

En proxyserver utvecklades i PHP med syftet att undvika ovan nämnda bristerna i AKKAs API gällande CORS. Genom att från proxyservern göra HTTP-förfrågningar till AKKAs API undviks användandet av CORS. Därav ges möjlighet att slussa HTTP-förfrågningar från SimLink till AKKA genom proxyservern.

5.2.5 SimLink prestanda

Gränssnittet för Moodle (se avsnitt 5.2.4.1) använder sig av Moodle Web Service API för kommunikation med Moodle. Kommunikationen består HTTP-förfrågningar från SimLink till Moodle. Varje förfrågan beskriver vilken funktion i Moodle Web Service API som ska svara på den specifika förfrågan.

Funktionerna i Moodle Web Service API svarar endast med den efterfrågade datan specifik för funktionen. Om mer data krävs måste flera HTTP-förfrågningar kombineras.

Till exempel krävs det sju stycken HTTP-förfrågningar till Moodle Web Service API om det för en användare med fyra kurser efterfrågas vilken betygsstatus de fyra kursernas samtliga aktiviteter har. Ett exempel i pseudokod visas nedan.

```
user        = get_user(username)
courses     = get_users_courses(user)
assignments = get_assignments(courses)

for course in courses
    get_grade(user, course)
```

> I koden ovan ger `get_grade()` endast information om vilka `assignments` som har vilket `grade`, ingen information ges om de `assignments` som efterfrågats. Därav den extra förfrågan till `get_assignments()`.

Ett idealt scenario skulle vara att endast en HTTP-förfrågan skickades. Detta är möjligt om en egen webbtjänst utvecklas för Moodle.

5.2.5.1 Skräddarsydd aktivitet

SimLink använder sig av aktiviteten `Assignment` för att representera en övning i Moodle (se avsnitt 4.6.3). Aktiviteten innehåller datafält som täcker behoven för att prototypen ska fungera. Dock finns det oanvända fält, samt önskade fält som saknas.

I tabell 5.2 listas några exempel på använda, oanvända och önskade fält. Önskade fält är fält som hade varit till hjälp vid implementationen av SimLink.

Använda fält	Oanvända fält	Önskade fält
`Assignment name`	`Competencies`	`Exercise GUID`
`Description`	`Tags`	`Scenario data`
`Grades`	`Restrict access`	`Result criteria`
`Activity completion`	`Submission settings`	

Tabell 5.2: Tabell över fält.

Avsaknandet av de önskade fälten löstes genom att i SimLink hårdkoda de mappningar som krävdes för en fungerande prototyp.

Till exempel kopplades `Assignment name` till ett `Exercise GUID` för att veta vilken övning det var som en specifik aktivitet representerade.

En skräddarsydd aktivitet tillåter valet av vilka fält som finns tillgängliga i aktiviteten. Prototypen innefattar inte någon utvecklad skräddarsydd aktivitet.

5.3 Rekommendation till Saab

Implementationen av SimLink resulterade i en färdig prototyp med vissa områden som behöver förbättras. Ett förslag på förbättringar har tagits fram för samtliga områden. Förbättringarna kan brytas ned i följande kategorier:

1. Utveckla en specialanpassad webbtjänst för Moodle
2. Utveckla en specialanpassad aktivitet för Moodle
3. Komplettering av AKKA

Redogörelse för samtliga kateogier görs i avsnitt 5.3.1, 5.3.2 och 5.3.3. En övergripande bild över systemet som helhet ges i avsnitt 5.3.4.

5.3.1 Specialanpassad webbtjänst för Moodle

En specialanpassad webbtjänst bör utvecklas för Moodle på grund av prestandaskäl. Det är också ett krav att använda en specialanpassad webbtjänst för att kunna utnyttja en specialanpassad aktivitet (avsnitt 5.3.2).

En specialanpassad webbtjänst innebär ett externt API där full kontroll ges över vilken funktionalitet som exponeras och vilken data som skickas från och tas emot av webbtjänsten.

Utvecklingen av en webbtjänst görs i PHP som ett Moodleplugin. Vidare information för utveckling av webbtjänster finns i dokumentationen för *External functions API* [16].

5.3.2 Specialanpassad aktivitet

En specialanpassad Moodleaktivitet bör utvecklas för att bättre representera en simulerad övning. En specialanpassad Moodleaktivitet tillåter full kontroll över tillgängliga fält samt vilken typ av data som tillåts i dessa. Behovet att hårdkoda mappningen mellan övning i AKKA och aktitvitet i Moodle kan undvikas genom att i aktiviteten ha ett fält som sköter mappningen.

Vidare information för utveckling av aktiviteter finns i dokumentationen för *Activity modules* [37].

5.3.3 Komplettering av AKKA

Komplettering av AKKA bör göras för att tillhandahålla behövd funktionalitet.

5.3.3.1 AKKA API

Den utvecklade prototypen använder sig av en trigger i Akka för att utvärdera resultat. Först skickas en HTTP-förfrågan för att utvärdera resultatet för en specifik övning och användare. Sedan skickas en till HTTP-förfrågan för att hämta den genererade utvärderingen.

Det saknas funktionalitet i AKKAs API för att utvärdera en användares övning beroende på specificerade kriterier. En komplettering av AKKAs API med sådan funktionalitet innebär att behovet av en trigger kan undvikas samtidigt som funktionaliteten går att göra med endast en HTTP-förfrågan.

5.3.3.2 CORS

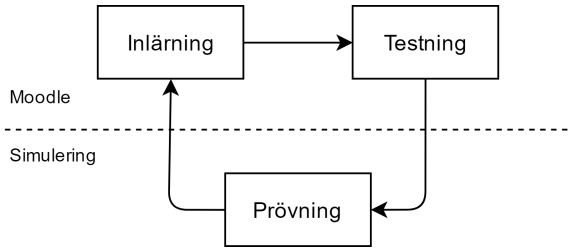
Då en proxy innebär ytterligare utveckling och underhåll bör istället stöd för CORS implementeras i AKKA. Implementeringen hade möjliggjort användning av SimLink utan en proxy.

5.3.4 Rekommenderat komplett system

Ett rekommenderat komplett system innefattar de ändringar som rekommenderats i avsnitt 5.3.1, 5.3.2 och 5.3.3.

Två möjliga kompletta system rekommenderas. De rekommenderade systemen beror på antalet träningsdeltagare som deltar i en simulerad övning.

Ett system rekommenderas för träning som sker av ensam träningsdeltagare, ett annat system rekommenderas för träning som sker med flera träningsdeltagare tillsammans med en eller flera träningsledare. De båda systemen lever upp till kravet på den utbildningsprocess vilken redogörs för i avsnitt 1.3 (se figur 5.5).

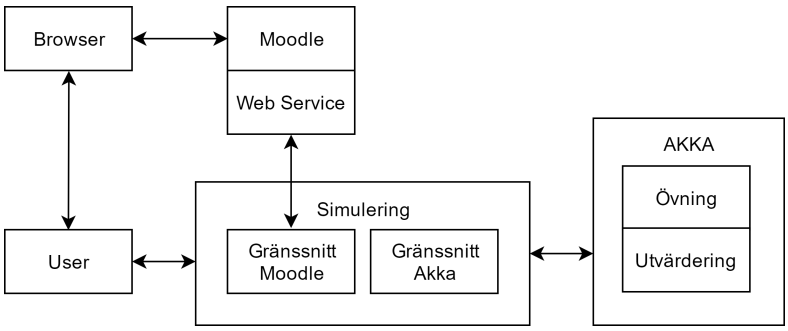


Figur 5.5: Översiktlig bild av ett digitaliserat träningsmoment

Användningsscenarion och vidare förklaring om de båda systemen ses i avsnitt 5.3.4.1 respektive 5.3.4.3.

5.3.4.1 Rekommenderat system för ensam träningsdeltagare

I figur 5.6 nedan visas en översiktlig bild över ett system optimerat för en ensam träningsdeltagare.



Figur 5.6: Rekommenderad lösning med SimLinkfunktionalitet i simulering.

I systemet som visas i figur 5.6 finns inte SimLink som fristående delsystem. SimLinks funktionalitet byggs istället in i simuleringsdelen.

Sammanslagningen av SimLink och simulering resulterar i att behovet för en träningsledare elimineras, samt färre delsteg i det användarscenario som redogörs

för i avsnitt 5.3.4.2. Sammanslagningen kräver dock att integrationen av SimLink görs för alla simuleringssystem som använder AKKA.

5.3.4.2 Scenario för ensam träningsdeltagare

Nedan listas ett exempel för hur samtliga steg i utbildningsprocessen ser ut för rekommenderat komplett systemet där en träningsdeltagarare tränar ensam. Av de 20 stycken steg som listas kan 12 stycken utföras automatiskt. Steg som utförs automatiskt betecknas med ordet *automatiskt*.

Inlärning

1. Träningsdeltagare loggar in på Moodle
2. Träningsdeltagare väljer kurs
3. Träningsdeltagare tar del av listade resurser för inlärning

Testning

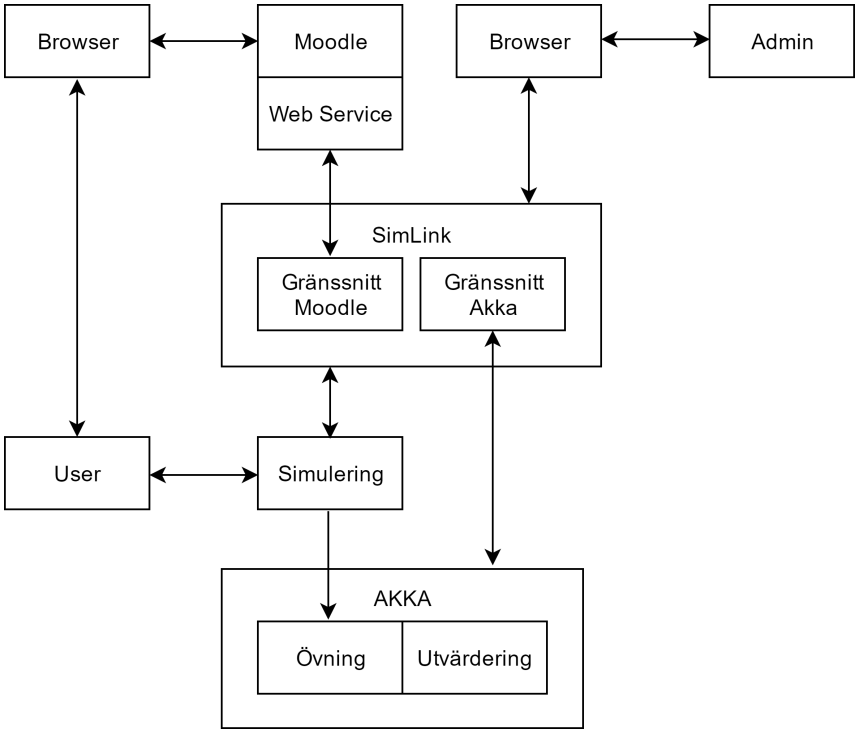
4. Träningsdeltagare loggar in på Moodle
5. Träningsdeltagare väljer kursdelmoment för testning
6. Täningsdeltagare utför test för valt delmoment
7. Resultatet för utfört test sparas *automatiskt* i Moodle
8. Godkänt resultat tillåter träningsdeltagare fortgå till **prövning** för delmoment *automatiskt*

Prövning

9. Träningsdeltagare loggar in på en simuleringsstation
10. Träningsdeltagare väljer övningsmoment
11. Scenariodata för valt övningsmoment hämtas *automatiskt* från Moodle
12. Övning i AKKA skapas *automatiskt* för valt övningsmoment
13. Simuleringsstationen laddas *automatiskt* med scenariodata
14. Övningsmomentet startas *automatiskt*
15. Simuleringsstationen genererar *automatiskt* observationer
16. Genererade observationer sparas *automatiskt* i AKKA
17. Övningsmomentet avslutas *automatiskt*
18. Kriterier för övningsmoment hämtas *automatiskt* från Moodle
19. Avslutat övningsmoment utvärderas *automatiskt* utefter hämtade kriterier
20. Betyg i Moodle för övningsmoment uppdateras *automatiskt* utefter utvärderingsresultat

5.3.4.3 Rekommenderat system för flera träningsdeltagare

I figur 5.7 nedan visas en översiktlig bild över ett system optimerat för flera träningsdeltagare tillsammans med träningsledare.



Figur 5.7: Rekommenderad lösning med SimLink som webb-applikation.

I systemet som visas i figur 5.7 är SimLink ett fristående delsystem, detta för att kunna användas av en övningsledare för att administrera övningsdeltagare under övningstillfället. Användarscenario för flera träningsdeltagare redogörs i avsnitt 5.3.4.4

5.3.4.4 Scenario för flera träningsdeltagare

Nedan listas ett exempel för hur samtliga steg i utbildningsprocessen ser ut för rekommenderat komplett systemet där flera träningsdeltagare tränar tillsammans med en träningsledare. Av de 23 stycken steg som listas kan 12 stycken utföras automatiskt. Steg som utförs automatiskt betecknas med ordet *automatiskt*.

Inlärning

1. Träningsdeltagare loggar in på Moodle
2. Träningsdeltagare väljer kurs
3. Träningsdeltagare tar del av listade resurser för inlärning

Testning

4. Träningsdeltagare loggar in på Moodle
5. Träningsdeltagare väljer kursdelmoment för testning
6. Träningsdeltagare utför test för valt delmoment
7. Resultatet för utfört test sparas *automatiskt* i Moodle
8. Godkänt resultat tillåter träningsdeltagare fortgå till **prövning** för delmoment *automatiskt*

Prövning

9. Träningsledare loggar in på SimLink
10. Träningsledare väljer övningsmoment
11. Träningsdeltagare loggar in på simuleringsstationerna
12. Träningsdeltagare rolldesigneras *automatiskt* eller manuellt av träningsledare
13. Scenariodata för valt övningsmoment hämtas *automatiskt* från Moodle
14. Övning i AKKA skapas *automatiskt* av SimLink för valt övningsmoment
15. Simuleringsstationerna laddas *automatiskt* med scenariodata
16. Träningsledare startar övningsmomentet
17. Simuleringsstationer genererar *automatiskt* observationer
18. Genererade observationer sparas *automatiskt* i AKKA
19. Träningsledare avslutar övningsmomentet
20. Kriterier för övningsmoment hämtas *automatiskt* från Moodle
21. Avslutat övningsmoment utvärderas *automatiskt* utefter hämtade kriterier
22. Utvärderingsresultat på gruppnivå visas *automatiskt* i SimLink
23. Betyg i Moodle för övningsmoment och träningsdeltagare uppdateras *automatiskt* utefter specifik träningsdeltagares utvärderingsresultat

Kapitel 6
Slutsats

Syftet med examensarbetet var att undersöka möjligheten att digitalisera en utbildningsprocess. Digitaliseringen skulle ske genom att knyta samman utbildningsplattformen Moodle med Saabs träningsssytem.

Sammankopplingen krävde en kommunikationskanal mellan Moodle och Saabs träningsystem. Ett antagande gjordes att xAPI, SCORM och Moodle API var möjliga kommunikationskanaler. För att ge insikt om för- och nackdelar gjordes en undersökning där varje möjlig kommunikationskanal utvärderades.

Resultatet av undersökningen ledde till valet av kommunikationskanal för vidare implementation i prototypen.

6.1 Vilka kommunikationsmöjligheter finns det mellan Moodle och simulerad övning?

Undersökningen omfattade tre olika kommunikationsmöjligheter; xAPI, SCORM och Moodles egna API. En egenutvecklad webbtjänst för Moodle diskuteras även i rapporten (se avsnitt 5.3.1) dock sker ingen implementering av en webbtjänst i prototypen.

6.2 Vilka för- och nackdelar finns med de olika kommunikationsmöjligheterna?

Endast ett av de tre undersöka kommunikationsmöjligheterna passade målbilden för examensarbetet. Målbilden innebär att en kommunikationskanal mellan Moodle och AKKA ska möjliggöras av det givna protokollet, där information om tidigare simulering ska vara tillgänglig och resultat från simuleringar ska kunna sparas i Moodle.

Moodles API är en webbtjänst för kommunikation mellan Moodle och extern applikation. Moodles API var den enda av de tre alternativ som passade målbilden. Då webbtjänsten är skriven för att passa många olika ändamål blev den stundtals ineffektiv för prototypens behov. Ineffektiviteten grundar sig i att vissa av SimLinks funktioner använder sig av ett antal HTTP-förfrågningar när en mer för ändamålet utvecklad förfrågning hade varit tillräcklig (se avsnitt 5.2.5).

SCORM och xAPI antogs från start vara protokoll för kommunikation med utbildningsplattformar, vilket inte var fallet. xAPI är ett protokoll för kommunikation till och från en LRS, det finns möjlighet att låta Moodle skicka och hämta information med xAPI men detta sker då till en LRS, inte till en extern applikation (i detta fall AKKA).

SCORM är integrerat med Moodle från start och SCORM-resurser har möjlighet att skicka information till och från Moodle, detta innebär att om den externa applikationen (AKKA) ska ha möjlighet att skicka data till och från Moodle måste den inkapslas som en SCORM-resurs. SCORMs RTE som specificerar kommunikationen är begränsad och passar inte examensarbetets målbild.

6.3 På vilket sätt kan resultat från simulerade övningar sparas i Moodle?

Simulerade övningar kan representeras i Moodle som aktiviteter. Resultatet från övningar kan sparas i Moodle i form av betyg på aktivitet för enskild träningsdeltagare. En skräddarsydd aktivitet tillåter även egendefinierade fält vilket möjliggör arkivering av till exempel hela utvärderingsrapporter, både på grupp- och individnivå samt annan arbiträr resultatdata.

6.4 Hur utvärderas dessa resultat?

Efter genomförd simulering utvärderas observationer i AKKA och ett resultat genereras. Resultat utvärderas i prototypen endast på individnivå, möjlighet finns dock till utvärdering på gruppnivå. Ingen implementation av detta gjordes i brist på tid.

6.5 Hur kan dessa resultat i sin tur påverka vidare simulerad övning?

För att påverka vidare simulerad övning måste föregående resultat kunna hämtas och läsas. Resultatet av övningar sparas i Moodle som betyg och poäng i en aktivitet. Betyg och poäng kan senare hämtas och utifrån dessa kan beslut som påverkar relevant simulerad övning fattas, till exempel svårighetsgrad. Implementation av detta ligger inom ramen för framtida utveckling.

6.6 Hur utvärderas resultat på grupp- och individnivå?

Resultat på individnivå utvärderas i AKKA genom att för varje specifik användare utvärdera dess observationer. Kriterier för utvärderingen hårdkodades i SimLink. En alternativ lösning hade varit att i en skräddarsydd aktivitet spara kriterier gällande utvärdering för specifik övning.

Resultat på gruppnivå utvärderas på samma sätt som resultat på invidividnivå med skillnaden att fler användares observationer tas i beaktande. En implementering där en utvärdering görs på gruppnivå ligger inom ramen för framtida utveckling.

6.7 Hur kopplas en Moodleidentitet till en användare av en simulerad övning?

Både Moodle och AKKA använder sig av unika identifierare för sina användare. Däremot är en identifierare för en användare i Moodle är inte identisk med samma användares identifierare i AKKA.

En koppling mellan de två olika identifierarna för samma användare görs i SimLink med en associativ vektor. Denna lösning är inte optimal då den skalar dåligt med antal övningsdeltagare. Möjlighet finns att spara AKKAs unika identifierare för en användare i Moodlerepresenationen för samma användare, denna lösning har inte implementerats i brist på tid.

6.8 Reflektion över etiska aspekter

En aspekt av examensarbetet som bör tas upp är behandlandet av personuppgifter.

Prototypen som utvecklats behandlar information lagrad i Moodle. En del av den informationen som behandlas är personlig information om Moodlenavändare. Varje användare i Moodle associeras med för- och efternamn, ett användarnamn samt en e-postadress. Det finns även möjlighet att lägga till information såsom telefonnummer, uppehållsland och profilbild.

Vid behandling av de personuppgifter som finns lagrade i Moodle måste personuppgiftslagen tas i akt.

> **31 § Säkerhetsåtgärder**
>
> Den personuppgiftsansvarige skall vidta lämpliga tekniska och organisatoriska åtgärder för att skydda de personuppgifter som behandlas. Åtgärderna skall åstadkomma en säkerhetsnivå som är lämplig `[...]`
>
> `Personuppgiftslagen (SFS 1998:204)` [38]

SimLink i dess nuvarande stadie behandlar eventuellt känsliga personuppgifter. Inga av de personuppgifter som behandlas långtidssparas i SimLink. Kommunikationen mellan SimLink och Moodle är krypterad och sker över TLS-protokollet.

I och med att all kommunikation är krypterad och inga uppgifter långtidslagras kan antagandet göras att en, för examensarbetet, *lämplig säkerhetsnivå* åstadkommits. Implementering av vidare åtgärder ligger utanför området för detta examensarbete.

Ingen personlig data har behandlats vid utförandet av detta examensarbete.

6.9 Framtida utvecklingsmöjligheter

Under examensarbetets gång har sex utvecklingsmöjligheter identifierats.

Specialanpassad webbtjänst
Kommunikationen mellan SimLink och Moodle är inte optimal ur ett prestandaperspektiv. En rekommendation för att förbättra prestandan är att utveckla en specialanpassad webbtjänst för Moodle, se avsnitt 5.3.1.

Specialanpassad aktivitet i Moodle
Representationen av en övning i Moodle består i dagsskedet av Moodleaktiviteten *assignments*. En rekommendation för att förenkla framtida utveckling är att utveckla en specialanpassad Moodleaktivtet för att istället låta denna representera en simulerad övning i Moodle, se avsnitt 5.3.2.

Komplettering av AKKA
AKKAs API för generering av utvärderingsrapporter rekommenderas en komplettering. Det saknas funktionalitet för att utvärdera en användares övning beroende på specificerade kriterier, se avsnitt 5.3.3.

Implementering av CORS för AKKA
Prototypen använder sig av en proxy vid kommunikation med AKKA. För att undvika behovet av en proxy kan istället CORS implementeras för AKKA, se avsnitt 5.2.4.2.

Alternativt användingsområde för SCORM
Även om SCORM inte ansågs passa examensarbetets målbild vad gäller kommunikationsförmåga mellan simulerad övning och utbildningsplattform insågs alternativa användningsområden för SCORM. Ett SCORM-paketerat delmoment för inlärning och testning (se avsnitt 1.3) kan importeras av Moodle. Detta innebär en möjlighet för Saab att låta underleverantörer skapa delmoment för inlärning och testning, se avsnitt 4.4.1.

Implementering av xAPI för AKKA
Den standard som xAPI följer för skapandet av statements (se avsnitt 2.4) liknar den AKKA följer för behandling av observationer. En komplettering av AKKAs API hade möjliggjort användning av xAPI som standard. Detta i sin tur hade tillåtit AKKA att hantera observationer från andra källor som följer standarden.

Kapitel 7
Terminologi

7.1 Förkortningar

Nedan listas rapportens samtliga förkortningar i alfabetisk ordning.

API	Applikationsprogrammeringsgränssnitt.	
CAM	Content Aggregation Model.	(avsnitt 2.3)
CORS	Cross-origin resource sharing.	(avsnitt 2.9)
GUID	Globally Unique Identifier.	
HTTP	Hypertext Transfer Protocol.	
JSON	JavaScript Object Notation.	
LMS	Learning Management System.	(avsnitt 2.5)
LRS	Learning Record Store.	(avsnitt 2.4.1)
REST	Representational state transfer.	(avsnitt 2.6)
RTE	Run-Time Environment.	(avsnitt 2.3)
SCORM	Sharable Content Object Reference Model.	(avsnitt 2.3)
SOAP	Simple Object Access Protocol.	
xAPI	eXperience API.	(avsnitt 2.4)
XML	eXtensible Markup Language.	
XML-RPC	XML-remote procedure call.	

Källförteckning

[1] R. Ellis, *A Field Guide to Learning Management Systems*,
http://www.astd.org/~/media/Files/Publications/LMS_fieldguide_20091
Hämtad: 19 juni 2017

[2] Universitets- och Högskolerådet, *Svensk-engelsk ordbok för den högre utbildningen*,
https://www.uhr.se/publikationer/svensk-engelsk-ordbok/larplattform
Hämtad: 19 juni 2017

[3] GitHub, *Moodle*,
https://github.com/moodle/moodle
Hämtad: 19 juni 2017

[4] P. Jesukiewicz, *Content Aggregation Model [CAM]*,
https://www.adlnet.gov/public/uploads/SCORM_2004_4ED_v1_1_Doc_Suite.zip
Hämtad: 19 juni 2017

[5] P. Jesukiewicz, *Run-Time Environment [RTE]*,
https://www.adlnet.gov/public/uploads/SCORM_2004_4ED_v1_1_Doc_Suite.zip
Hämtad: 19 juni 2017

[6] RUSTICI Software, *What is Learning Record Store (LRS)?*,
http://scorm.com/tincanoverview/what-is-an-lrs-learning-record-store/
Hämtad: 19 juni 2017

[7] ADL, *xAPI Architecture Overview*,
https://www.adlnet.gov/adl-research/performance-tracking-analysis/experience-api/xapi-architecture-overview/
Hämtad: 19 juni 2017

[8] R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, Chapter 5,
http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
Hämtad: 19 juni 2017

[9] Vue.js, *What is Vue.js?*,
https://vuejs.org/v2/guide/
Hämtad: 19 juni 2017

[10] RUSTICI Software, *SCORM Explained - What is SCORM?*,
http://scorm.com/scorm-explained/
Hämtad: 19 juni 2017

[11] RUSTICI Software, *SCORM Run-Time Environment*,
http://scorm.com/scorm-explained/technical-scorm/run-time/
Hämtad: 19 juni 2017

[12] RUSTICI Software, *SCORM Explained - Technical SCORM*,
http://scorm.com/scorm-explained/technical-scorm/
Hämtad: 19 juni 2017

[13] Advanced Distributed Learning (ADL), *Experience API*,
https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-Data.md#parttwo
Hämtad: 19 juni 2017

[14] Advanced Distributed Learning (ADL), *xAPI Architecture Overview*,
https://www.adlnet.gov/adl-research/performance-tracking-analysis/experience-api/xapi-architecture-overview/
Hämtad: 19 juni 2017

[15] RUSTICI Software, *The Learning Record Store (LRS)*,
http://tincanapi.com/learning-record-store/
Hämtad: 19 juni 2017

[16] Moodle, *External functions API*,
https://docs.moodle.org/dev/External_functions_API
Hämtad: 19 juni 2017

[17] Moodle, *Web service API functions*,
https://docs.moodle.org/dev/Web_service_API_functions
Hämtad: 19 juni 2017

[18] Moodle, *Activities*,
https://docs.moodle.org/32/en/Activities
Hämtad: 19 juni 2017

[19] Moodle, *Resources*,
https://docs.moodle.org/32/en/Resourcers
Hämtad: 19 juni 2017

[20] Martin Dougiamas, *Moodle origins*, 2005-07-17
https://moodle.org/mod/forum/discuss.php?d=27533#p129848
Hämtad: 19 juni 2017

[21] Moodle, *Moodle architecture*,
https://docs.moodle.org/dev/Moodle_architecture
Hämtad: 19 juni 2017

[22] Moodle, *Courses*,
`https://docs.moodle.org/28/en/Courses`
Hämtad: 19 juni 2017

[23] Mozilla Development Network, *HTTP access control (CORS)*,
`https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS`
Hämtad: 19 juni 2017

[24] JetBrains - PhpStorm, *Features*,
`https://www.jetbrains.com/phpstorm/features/`
Hämtad: 19 juni 2017

[25] Atom, *Atom - A hackable text editor*,
`https://atom.io/`
Hämtad: 19 juni 2017

[26] ShareLaTex, *ShareLaTex - What is LaTeX*,
`https://www.sharelatex.com/learn/Learn_LaTeX_in_30_minutes#/What_is_LaTeX.3F`
Hämtad: 19 juni 2017

[27] WampServer, *WAMPSERVER - a Windows web development environment*,
`http://www.wampserver.com/en/`
Hämtad: 19 juni 2017

[28] Microsoft, *Visual Studio*,
`https://www.visualstudio.com/vs/ide/`
Hämtad: 19 juni 2017

[29] Draw.io, *Draw.io*,
`https://www.draw.io/`
Hämtad: 19 juni 2017

[30] NPM Inc, *Node Package Manager*,
`https://www.npmjs.com/`
Hämtad: 19 juni 2017

[31] Node.js, *About Node.js*,
`https://nodejs.org/en/about/`
Hämtad: 19 juni 2017

[32] Google, *Chrome*,
`https://www.google.com/chrome/browser/desktop/index.html`
Hämtad: 19 juni 2017

[33] KTH - Cerisegruppen, *Parprogrammering*,
`http://www.csc.kth.se/tcs/projects/cerise/parprogrammering/`
Hämtad: 19 juni 2017

[34] W3C, *Web Services Architecture*,
`https://www.w3.org/TR/ws-arch/#whatis`
Hämtad: 19 juni 2017

[35] The PHP Group, *PHP: General Information*,
`http://php.net/manual/en/faq.general.php`
Hämtad: 19 juni 2017

[36] Preece J, Rogers Y, & Sharp H, (2001) *Interaction Design: beyond human-computer interaction*, S. 249

[37] Moodle, *Activity Modules*,
`https://docs.moodle.org/dev/Activity_modules`
Hämtad: 19 juni 2017

[38] Svensk författningssamling, *Personuppgiftslag (1998:204)*
`http://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/personuppgiftslag-1998204_sfs-1998-204#SÃďkerhetsÃětgÃďrder`
Hämtad: 19 juni 2017

[39] Saab, *AKKA 3.0 JSON API*

[40] Saab, *AKKA 4.3 SDK DEVELOPER'S GUIDE*

[41] Saab, *AKKA 4.3 SETUP MANUAL*

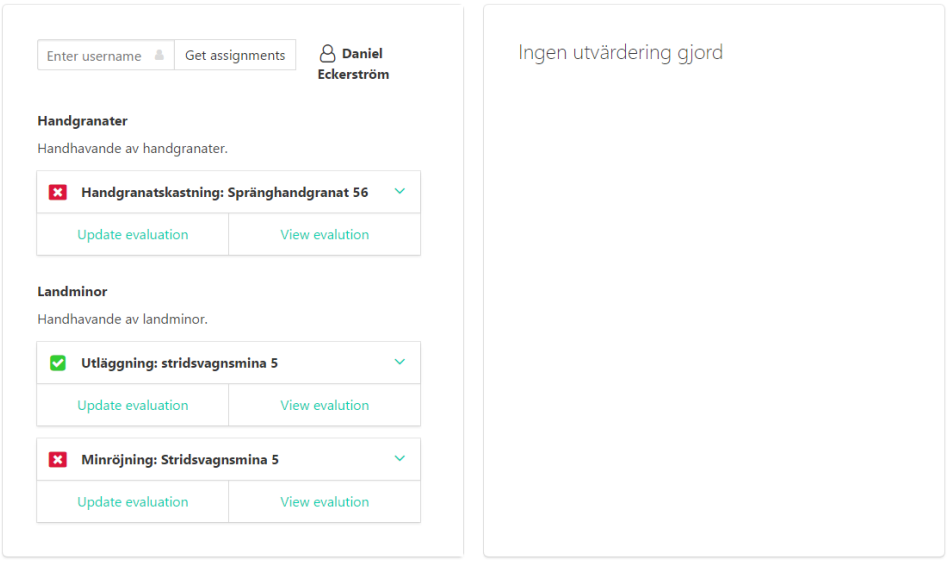
[42] Saab, *AKKA 4.3 USER'S MANUAL*

Appendix A
Användning av SimLink

Nedan visas ett exempel på hur SimLink används. Samtliga bilder refererar till den händelsekedja som beskrevs i avsnitt 5.2.1.

A.1 Visning av användares kurser

Innefattar steg 1 och 2 av händelsekedjan.

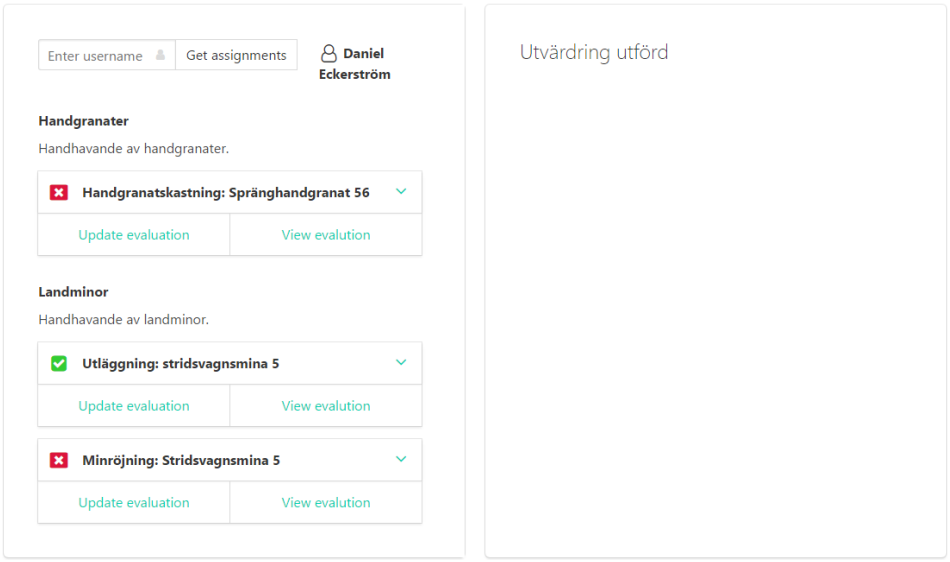


Figur A.1: SimLink vid visning av användares kurser

Efter inmatning av ett användarnamn, i detta exempel `daniel`, visas användarens kurser och delmoment. Samtliga delmoment visas även med dess betygstatus. Betygstatusen ses som ett rött kryss för icke godkänt eller som en grön bock för godkänt.

A.2 Utvärdering av simleringsaktivitet för användare

Innefattar steg 3 av händelsekedjan.

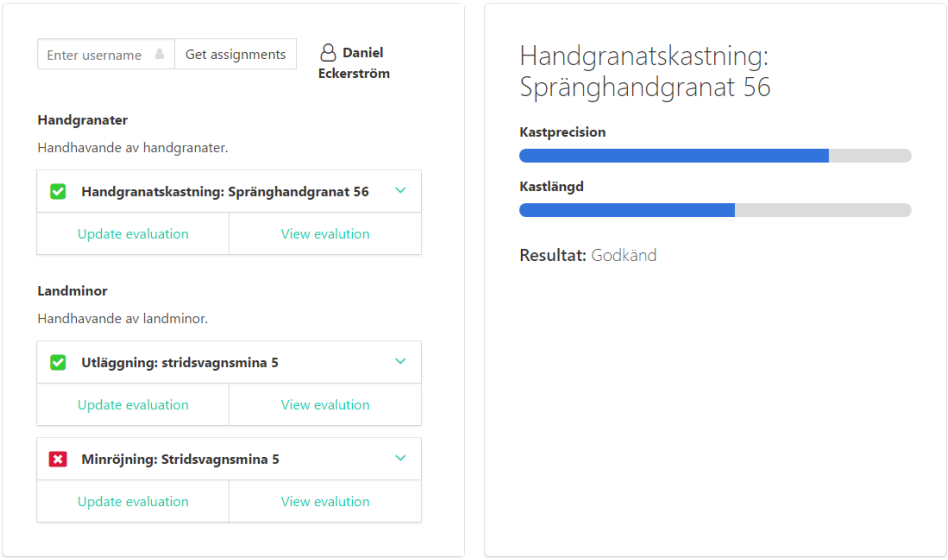


Figur A.2: SimLink vid utvärdering av simuleringsaktivitet

Varje delmoment har två knappar. Den vänstra knappen används för att utvärdera valt delmoment. I detta exempel har en utvärdering gjorts för `Handgranatskastning: Spränghandgranat 56`. Ett meddelande visas som bekräftelse.

A.3 Visning av utvärderingsresultat

Innefattar steg 4 och 5 av händelsekedjan.



Figur A.3: SimLink vid visning av utvärderingsresultat

Den högra knappen för de delmoment som listan används för visning av utvärderingsresultat. Endast det delmoment som tidigare utvärderades kan visas.

I detta exempel visas utvärderingsresultatet för det tidigare utvärderade delmomentet `Handgranatskastning: Spränghandgranat 56`.

Det utvärderade resultat påvisar godkänt vilket i sin tur uppdaterar betygstatusen för delmomentet till en grön bock.