Forside

Forord

Denne rapport er et resultat af 3. semesters projekt på diplomingeniøruddannelsen i Informations- og Kommunikationsteknologi ved SDU.

I Projektforløbet har vi arbejdet med de områder, metoder og teknikker vi er blevet undervist i på de tre første semestre på uddannelsen. Vi lægger i dette 3. semester projekt vægt på de kompetencemål der er sat for de enkelte fag.

Projektvejleder

Steffen Peter Skov har været projektvejleder på projektet og vi har løbende haft møder med ham.

Indledning

Rapporten har til formål at dokumentere udviklingen af et system, samt redegøre for de metoder og teknikker der anvendes i forbindelse med projektarbejdet.

Rapporten vil udelukkende henvende sig til undervisere og censor, da opgavens problemformulering er fiktiv og ikke omhandler udvikling af et system til løsning af et konkret problem. Projektgruppen har sammen formuleret opgaven.

Læsevejledning

Rapporten er skrevet i sammenhæng og vil give bedst mening hvis den læses i den rigtige rækkefølge fra start til slut.

Rapporten er delt op i følgende afsnit:

1. Projektgrundlag
2. …
3. …
4. …
5. ..
6. …
7. …
8.

Begreber i rapporten

Der vil blive brugt mange begreber og forkortelser i rapporten, så for at læseren har forståelse af begreberne som vi har under udarbejdelse, så er de listet i tabel XXX med navn og tilhørende beskrivelse.

Vi har valgt at begreber skrives på engelsk i rapporten.

Begreb	Beskrivelse
CSS	Central Lager System
DSS	Detail Store System. Det system der sender ordre til *DSS*.
RCS	Robot Central System findes på det fysiske lager. Robotten placerer Items på

	lageret ud fra instrukser fra *CSS*.
Stock	Lageret der er inddelt i mange *StockPositions*.
StockPosition	Er en given plads på lageret, der kan holde én Item.
Scanner	Findes på lageret og bruges når en vare ankommer til lageret.
Stregkode (QR kode)	Den kode der sidder på en vare fra dens producent.
OrderRequest	Når en Detailbutik sender en ordre modtages denne af *CSS* og laves til en *OrdreRequest* i systemet.
Order	En ordre med tilhørende Items oprettes i *CSS* ud fra en *orderRequest*.
Confirmation	Er en bekræftelse der sendes til *DSS* når en ordre er modtaget i *CSS*.
ItemType	Er en varetype, som en Item tilhører. Den kan fx være Gevalia kaffe Økologisk.
Item	Er én vare i systemet. Kan fx være én sofa, men også én kasse kaffe med 10 stk i. Det er altså den mindste enhed af en *ItemType,* der kan bestilles.
Administrator	Den person der har fuld adgang til systemet og har mulighed for at oprette og slette fra systemet.
StockEmployee	Ham der er ansat på *Stock* og har til opgave at ekspederer ordre.

Figur xxx : begrebstabel

Afprøvning af program

Der vil bagerst i rapporten være en lomme med en dvd, der indeholder programmet. (indstallatione?).

Bla bla

1. Projektgrundlag

1.1 Interessenter i projektet

- *Projektgruppen*: Der skal bedømmes ud fra projektforløbet og resultat af dette.
- *Steffen Peter Skov*: Der er gruppen vejleder og lige ledes skal vurdere rapporten.
- *Censor*: Der skal vurdere rapporten og systemet.

1.2 Projektgruppen

Projektgruppen består af fire personer: Anders Kold, Henning Fich, Nico Rasmussen og Kristina Hussak. Alle i projektgruppen vil deltage aktivt i projektarbejdet.

1.3 Beskrivelse

Central Stock System (CSS).

Et nyt centrallager skal implementere et system til styring af varer: Bestilling, Modtagelse, lager og udlevering samt grundlæggende bogholderi.

Bestilling af vare til lageret foretages hos eksterne leverandører og registreres, som "bestilt" i systemet.

En vares rejse gennem centrallageret forløber som følger: Ved ankomst til varemodtagelsen (EUR paller) scannes varen ind med stregkode eller QR kode og køres i dybdereol. Et robotstyret lager sørger automatisk for at varen opbevares forsvarligt til den skal udleveres. Varer der skal kasseres skal scannes ud af systemet, som kasserede.

Når en butik ønsker at bestille vare fra lagret, sender de en bestilling til systemet.

Vare bestilt i systemet bliver hentet på lager og gjort klar til pakning. Ved udlevering bliver varen udskrevet fra lageret, og Systemet sørger for bogføring.

Systemet skal være indlysende at bruge for personalet(brugervenligt) og stabilt, det må bl.a. ikke gå ned på brugerfejl. Der skal laves backup en gang pr. døgn. Systemet skal være så uafhængig af platform som muligt.

1.4 Formål

At fremstille et System der kan varetage den grundlæggende funktionalitet for drift af et centrallager. Systemet skal medvirke til nedbringelse af driftsomkostninger gennem automatisering af arbejdsprocesser.

1.5 Mål

1.5.1 Produktmål

Målet for produktet er:

1. at få udviklet et system, der kan håndtere og administrere varer på et lager.
2. at få udviklet et distribueret system (f.eks. Client/Server)
3. at få udviklet et system der indbefatter automation (PLC)

4. at anvende netværksprotokoller i kommunikationen mellem de enkelte arbejdsstationer i systemet.
5. at få det beskrevet i en god rapport.

1.5.2 Procesmål

Ved projektets afslutning skal det enkelte gruppemedlem alene og i samarbejde med andre:

1. være blevet bedre til at bruge de udviklingsværktøjer vi lærer om (UP, Scrum) til systemudvikling.
2. kunne opbygge og forstå en netværksbaseret softwareløsning.
3. have forståelse for softwarearkitekturen og have fokus på genbrugeligt design.
4. kunne anvende versionsstyringssystem til versionering af dokumenter og kildetekster.
5. have opnået større erfaring med MySQL, Java, netværksprotokoller og automation.
6. kunne lave en holdbar projektplan så milepæle og artefakter bliver klar til tiden.
7. have fået gode erfaringer med udvikling af et **distribueret system.**

1.6 Inddragelse af fagområder

- SRO bruges i forbindelse med lagerrobot og implementering i et distribueret system.
- SUD bruges i forbindelse med udvikling af systemets forskellige dele og fastlæggelse af systemets arkitektur og design.
- KOM bruges i forbindelse med socket programmering i det distribuerede system.

1.7 Samarbejdet

Projektgruppen skal arbejde som en fladt struktureret gruppe, hvor alle er aktive og tager hånd om projektet.

Der afholdes møde fast hver torsdag kl. 12.15 med mulighed for vejledning (i løbet af mødet). Der laves dagsorden og referat. Der er mødepligt. Yderligere møder i den kommende uge besluttes her. Arbejdsopgaver til næste møde uddelegeres.

Konflikter håndteres på en fornuftig måde, efterhånden som de opstår.

1.8 Dokumenter

Rapporten skrives i Word, mens diagrammer laves i Cacoo. Alle dokumenter gemmes i fælles Dropbox.

Der arbejdes i iterationer og der arbejdes på rapporten undervejs. De enkelte artefakter til rapporten skrives undervejs og lægges i særskilte dokumenter. Ved afslutning samles alle dokumenter til en samlet rapport.

Kildekode versionsstyres gennem GitHub som er et Open Source versionsstyringssystem.

1.9 Ressourcer

Automations delen er ikke endeligt fastlagt men vi forestiller os enten at bruge BradleyAllen Logix5000 eller Lego Mindstorm til at visualisere automation af et varelager. Der bruges alm. standard

værktøjer til produktion af artefakter (Word til dokumenter, Cacoo til diagrammer, MySQL til Database osv.)

Til rådighed under projektet er vejleder Steffen Peter Skov samt vores undervisere indenfor de respektive områder – PLC, netværk og systemudvikling.

1.10 Projektstyring

Der anvendes UP til den overordnede systemudvikling (iterationer, artefakter). Scrum vil blive afprøvet som redskab under inceptionsfasen.

1.11 Den overordnede projektplan

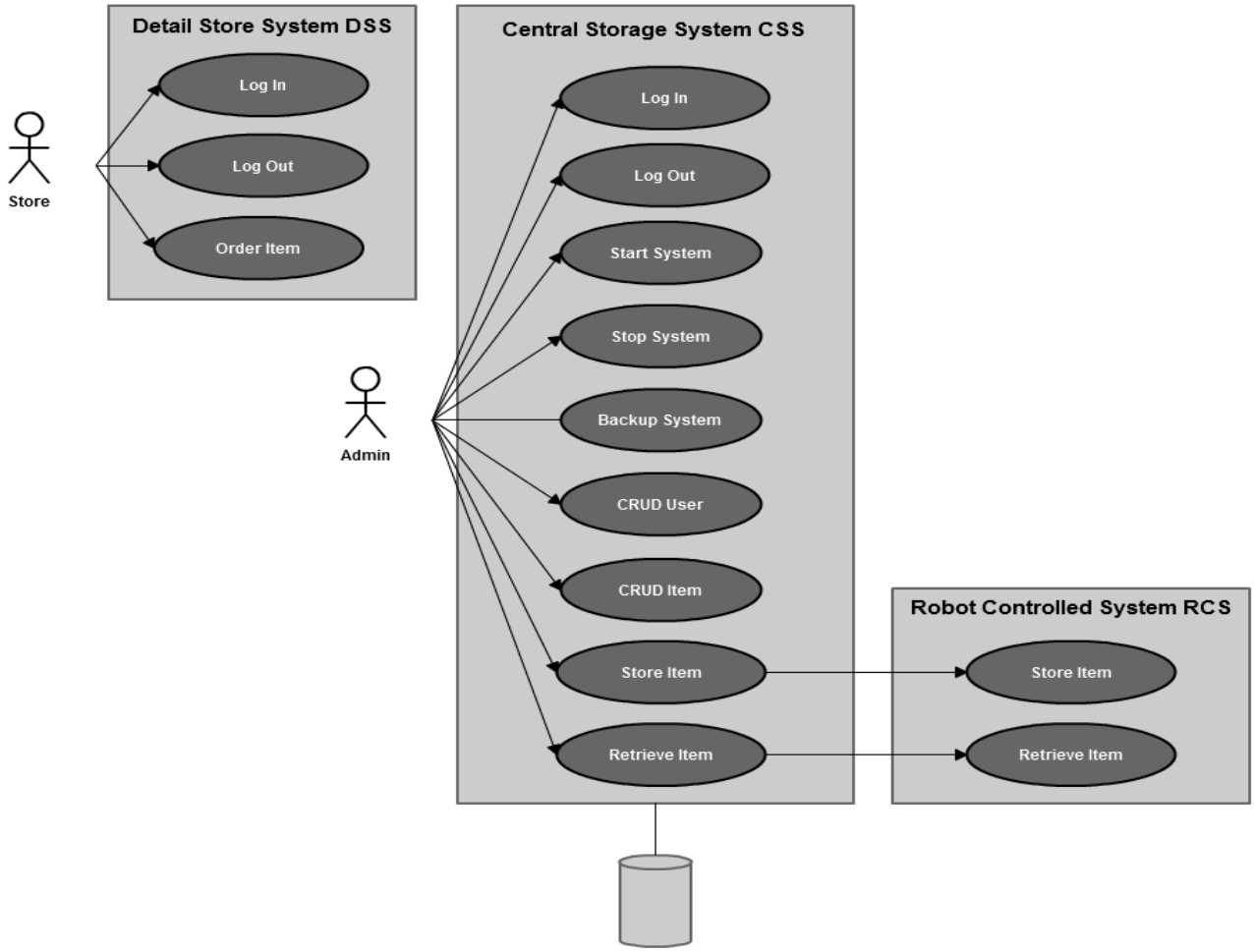
Den overordnede planlægning for projektets forløb fra start til slut.

Uge	Emne	Beskrivelse
40	Projektgrundlag	
41	Inception – 1. Iteration	
42	**EFTERÅRSFERIE**	
43	Inception – 1. Iteration	
44	Elaboration – 1. Iteration	
45	Elaboration – 1. Iteration	
46	Elaboration – 2. Iteration	
47	Elaboration – 2. Iteration	
48	Konstruktion	
49	Konstruktion	
50	Rapport	
	AFLEVERING	

1.12 Plan for inceptionsfasen

Projektplan	Start Dato	Slut Dato
Projektgrundlag		
Projektintroduktion	**10/09**	**21/09**
Case-introduktion	**17/09**	**21/09**
Problemstilling	**17/09**	**21/09**
Problemformulering	**17/09**	**21/09**
Målsætning	**17/09**	**05/10**
– Produktmål	**17/09**	**05/10**
– Procesmål	**17/09**	**05/10**
Projektafgrænsning	**24/09**	**05/10**
Projektværktøjer og -styring		
Rapportskrivning	**24/09**	**05/10**
Udviklingsmiljø	**24/09**	**05/10**
UML modellering	**24/09**	**05/10**
Database	**24/09**	**05/10**
Procesmodel	**24/09**	**05/10**
Inception		
Produktvision	**17/09**	**05/10**
Funktionelle krav	**24/09**	**05/10**
Ikke-funktionelle krav	**24/09**	**05/10**
Aktørliste	**24/09**	**05/10**
Use-case diagram(brugsmønster)	**24/09**	**09/10**
Prioritering	**24/09**	**11/10**
Milepæle	**08/10**	**23/10**

1.13 Use-case Diagram



2. Inceptionsfasen

….

2.1 Krav til systemet

I **Tabel 1.xxx** er de funktionelle krav til systemet listet. Kravene er fastlagt af projekt gruppen.

Funktionelle krav		
Krav	**Beskrivelse**	**Prioritet**
001	Systemet skal kunne håndtere *Vare* som er inddelt efter *Varetype.*	
002	Der skal kunne oprettes og slettes *Vare*, *Varetyper og brugere* i systemet.	
003	Der skal kunne redigeres i eksisterende *Vare*, *Varetype og Brugere* i systemet.	
004	En *Vare* skal have tilknyttet en *Lagerplacering.*	
005	Der skal være en søgefunktion i systemet, der gør det muligt at finde informationer om en *Vare*, *Varetype*, *Bruger* og *Ordre* i systemet.	
006	Brugere af systemet skal verificeres af systemet. Evt. via log ind.	
007	Systemet skal kunne modtage en *Ordre* fra en *Detailbutik*.	
008	Systemet skal kunne give besked til *Detailbutik* når en *Ordre* er modtaget.	
009	En ny *Vare* skal indscannes og registreres i systemet inden den bringes til lageret.	
010	Systemet skal give den korrekte *Lagerplacering* til *Lagerrobot*.	
011	Systemet skal kunne bede om at få hentet en *Vare* med en given *Lagerplacering* fra lager vha. *Lagerrobot.*	
012	Systemet skal kunne udskrive en *Følgeseddel* med *Detailbutikkens* informationer når en vare hentes fra *Lageret*.	
013	En *Detailbutik* skal kunne se *Vare* i systemet.	
014	En modtaget *Ordre* skal indeholde informationer om *Detailbutik*.	
015	Systemet skal kunne sende en ordrebekræftelse til *Detailbutik*.	

Tabel 1. xx – De funktionelle krav til systemet

Tabel 1. XX indeholder alle de ikke-funktionelle krav til systemet. Lige som de funktionelle er disse bestemt af projektgruppen selv.

Ikke-Funktionelle krav			
Krav	**Titel**	**Beskrivelse**	**Prioritet**
201	**Robusthed**	Systemet må ikke gå ned pga. brugerfejl	
202	**Brugervenligt**	Systemet skal kunne bruges uden efteruddannelse	
203	**Vareinformation**	Systemet skal kunne levere en dækkende beskrivelse af en vare (inkl. Billede?)	
204	**Dokumentation**	Systemet skal sørge for at lageret er ajourført.	
205	**Backup**	Systemet skal lave backup en gang pr. døgn.	
206	**Hardware**	Systemet skal være platformsuafhængigt (pc windows/linux etc.). Hardware skal kunne udskiftes uden at påvirke systemet	
207	**Sikkerhed**	Brugeradgang til Systemet skal begrænses	

Tabel 1.xx – De ikke-funktionelle krav til systemet

2.2 Prioritering af Use cases

Projektforløbet har været opdelt i flere iterationer og derfor prioriterede vi Use Cases i systemet for at finde ud af hvilke der skulle med i først iteration. **Tabel 1.XX** viser en liste af de use cases der findes i systemet og den prioritet vi har vælge at give dem.

Prioriteringen foregik ud fra *MoSCoW* metoden, der går ud på at prioritere efter begreberne: **M**ust, **S**hould, **C**ould og **W**ont.

Must	Skal med i den endelige version
Should	Burde være med i den endelige version og har derfor høj prioritet
Could	En feature som kunne være god at have med hvis tiden tillader det
Wont	En funktionalitet som ikke kommer med nu, men som evt. kommer med i næste version

Tabel 1.xx – Begreber i MoSCow metoden

Use Cases		
Use case	**Krav ref.**	**Prioritet**
Log in	006	C
Log out	006	C
Start system	-	S
Stop system	-	S
CRUD User	005, 002, 003	C
Receive order	007, 008, 014, 015	M (P30)

View orders	005	M (P8)
Store item	010, 009	M (P10)
Retrieve item	010, 011, 012	M (P10)

Tabel 1.XX – Tabel over use cases i systemet og deres tilhørende prioritering

!! Vi prioriterede ved at bruge SCRUM pointkort. Kortene indeholder et tal mellem ½ og 100.

2.3 High-level Use Cases

High-level Use Cases beskriver hvad der sker i de enkelte situationer i systemet. En Use Case er beskrevet med Aktør, formål og en beskrivelse af situationen.

Nedenfor kan ses *High-level Use Cases* for 1. iteration. (1. iteration er de centrale use cases og de resterende lægges som billag)

Receive Order	
Use case:	Receive Order
Actors:	DSS (Detail Storage System)
Purpose:	Receive an order and handle it accordingly.
Description:	System receives an order from another system. The attached encrypted user identification is validated. Each item on the order-list is cross-referenced with the amount of the specific item in stock. If an item is not in stock the lack is noted and attached to the order-confirmation, which is returned then to the user. The items on the order-list are marked as reserved until they are checked out of the System.

View Orders	
Use case:	View Orders
Actors:	Stock & Admin
Purpose:	Display a list of orders for the user to interact with.
Description:	The System constructs a list with all of the incoming orders and makes the list available on display for the user.

Store Item	
Use case:	Store Item
Actors:	Stock
Purpose:	Store a specific Item on its respective position.
Description:	The item is scanned and its position is retrieved from CSS and forwarded to the RCS (Robot Controlled System). The Item is then moved to its respective position and its information is stored in the database.

Process Order	
Use case:	Process Order
Actors:	Stock
Purpose:	Retrieve all items in a specific order from storage.
Description:	All items are continuously retrieved from storage. Every time an Item is retrieved it is checked out of the

	System. When all Items have been retrieved, the order status is changed.

Retrieve Item	
Use case:	Retrieve Item
Actors:	CSS (Central Storage System)
Purpose:	Retrieve a specific Item from a specific position.
Description:	The RCS is given an Item no. and with it a specific position; the item is then retrieved and delivered to the user. Additionally the system updates the information in the database.

2.3 Expanded Use Cases

I *expanded Use Cases* beskrives hvert Use Case mere detaljeret med *pre- og postconditions*, et typiske hændelsesforløb og et eventuelt alternativt hændelsesforløb.

Nedenfor er Expanded Use Cases for 1. iteration vist. De resterende *Expanded Use Cases* er vedlagt som bilag XXXX.

Receive Order	
Use case:	**Receive Order**
Actors:	DSS(Detail Store System)
Purpose:	Receive Order from DSS and save it in the system.
Overview:	CSS receives an order from a DSS over a secure socket connection. CSS confirms the identity of the detail store and converts the received order to an active order in CSS and stores it in the system. The system goes through the order and updates the stock by reserving each item in the order and associates it to the order. As the system goes through the order; reserving items, it puts together a Confirmation to send back to the detail store. The Confirmation holds information about which items can be delivered and which is in backorder.
Type:	Essential
Preconditions:	The DSS is known by the System.
Postconditions:	An Order is created in the system and the stock is updated.
Special Req.:	No special requirements are needed.

Flow of Events

Actor Action	System Response
1. This use case starts when a Detail store system sends a request to CSS containing an order and store information.	**2.** CSS responds to the request and a connection is established.
	3. CSS verifies the Detail store identity.
	4. CSS verifies that the Order is not a duplicate.
	5. CSS convert the incoming order to an order in CSS and store it in the system.
	6. CSS goes through the stock reserving and associating items to the Order.
	7. CSS creates a Confirmation and sends it to the DSS
8. The DSS sends a “Close connection” message to CSS.	
	9. CSS receives the “close connection” message, and closes the connection.

Alternative Flow of Events

Line 2:	CSS rejects the request (return to Line 1).
Line 3:	CSS cannot confirm the Detail store identity and rejects the order (**return to line 1**).
Line 4:	CSS find that the order is a duplicate (**go to line 9**)
Line 7:	The Detail store do not receive a Confirmation from CSS (**return to line 7 until timeout - on timeout go to line 9**)

View Orders

Use case:	**View Orders**
Actors:	Admin & Stock
Purpose:	Display a list of orders for the user to interact with.
Overview:	The System constructs a list with all orders and makes the list available on display for the user.
Type:	Essential
Preconditions:	The user must be known to the System.
Postconditions:	A list of orders is made available on display for the user.
Special Req.:	No special requirements needed.

Flow of Events

Actor Action	System Response
1. This use case starts when the user requests access to the list of orders.	**2.** System produces a list of all orders requested and makes it available on display.

Alternative Flow of Events

Store Item

Use case:	**Store Item**
Actors:	Manager, RCS
Purpose:	Store a specific Item on its associated stock position.
Overview:	The item is scanned and its stock position is retrieved from CSS and forwarded to the RCS (Robot Controlled System). The Item is then moved to its associated stock position and its information is stored in the database.
Type:	Essential
Preconditions:	The item type must exist in the system
Postconditions:	The item is added to storage, and the System is updated.
Special Req.:	No special requirements needed.

Flow of Events

Actor Action	System Response
1. The use case starts when Manager puts an item on a storage buffer, and clicks "Store Item"	2. RCS: The conveyer belt transports the item to a scanner, and scans the item. the scanned info is send to CSS 3. CSS: Looks up the item, using the scanned info, and sends the stock position back to the RCS 4. RCS. Places the item on its position, and returns a confirmation to CSS. 5. CSS: updates stock and returns a confirmation to the Manager.

Alternative Flow of Events	
Line 2:	Item type doesn't exist in the system. Return Item to storage buffer, and notify the Manager.

Process Order	
Use case:	**Process Order**
Actors:	Manager
Purpose:	Manager processes an Order, retrieve the ordered Items from RCS (Stock) and ready the Items for shipment.
Overview:	All items are continuously retrieved from storage. Every time an Item is retrieved it is checked out of the System. When all Items have been retrieved, the order status is changed.
Type:	Essential
Preconditions:	Stock positions of Items are known to the System.
Postconditions:	Order has its State changed to Processed. Items are removed from Stock
Special Req.:	No special requirements are needed.

Flow of Events

Actor Action	System Response
10. This Use Case begins when the Manager wants to process an Order	
11. The Manager requests the Order to be processed.	**12.** For each Item CSS looks up the Item's Stockposition and sends a 'Retrieve Item' with that Stockposition to RCS. RCS retrieves the Item and responds with a 'retrieved'. CSS removes the Item from Stock. On last Item the State of the Order is changed to Processed.

Alternative Flow of Events

Line 3:	If RCS responds with 'failure', CSS indicates a 'failure' for that Stockposition. CSS then looks up another 'not reserved' Item of same Itemtype and retrieves it instead. If no available Item CSS indicates a 'failure' for retrieving that Item.

Retrieve Item som Expanded Use case?!?!?

2.4 System-sekvens Diagrammer

System sekvensdiagrammerne viser aktøres