SOFTWARE

Software Characteristic

A
D
E
M

Maintainability:

Yazılım müşterilerin değişen ihtiyaçlarını karşılamalı. (Değişebilir olmalı)

Dependability and Security:

Güvenilir yazılım, sistem arızasında fiziksel ve ekonomik hasara neden olmamalı.
Kötü niyetli kullanıcılar sisteme erişememeli.

Efficiency:

Yazılım, sistem kaynaklarını israf etmemeli;
verimlilik, yanıt verebilirliği
işlem süresini
hafıza kullanımını
) içerir.

Acceptability: Tasarlandığı kullanıcı tipine uygun olmalı.

understandable
compatible uyumlu
usable
} with other systems

Çoğu sistemde maliyetlerin çoğu sistem değişirken gerçekleşir.

Application Types

1. Stand Alone Apps
PC'lerde çalışan, gerekli işlevleri için ağa [illegible] olmayan

2. Interactive Transaction Based Apps
Remote computer, kullanıcılar kendi comp.larından erişir. e-ticaret uygulamaları, web uyg

3. Embedded Control Systems
Donanımı kontrol eden app.

4. Batch Processing Systems
Large batches dataları işlemek için tasarlanırlar.

5. Entertainment systems
personal use

6. Systems for modeling and simulation
that are developed by scientists and engineers
durumları modellemek için geliştirilen

7. Data Collection Systems
sensörler kullanarak çevreden data toplayan, işlemek için diğer sistemlere gönderir.

8. Systems of systems

- 1. 17 slayt

SOFTWARE PROCESS ACTIVITIES

S
D
V
E

Software Specification: müşteri yemin.
üretilecek yazılımı ve çalışmasındaki kısıtlamaları tanımlar

Software Development: Design ve Programlama

Software Validation: Customer requires lar doğrultusunda olduğunu onaylamak için check

Software Evolution: müşteri ve pazar gereksinime göre yazılım değiştirme

General Issues that affect Most Software

- Heterogenity:
- Business and social change: Hızla yeni yazılımlar geliştirilmeli
- security and Trust

CHAPTER 2 - SOFTWARE PROCESSES

Plan Driven Process

Tüm faaliyetlerin planlı ilerleme bu plana göre ölçülür.

Agile Process

Planning is incremental.
zamanla gelişen ilerleyen..
* Değişen müşteri gereksinimlerini kolayca ayarlar

Most practical process is both puhahaha :D

Software Process Models

Plan Driven / Agile
PD / A

Waterfall model:
× Plan-driven
× Spesifikasyon ve gelişimin ayrı ve farklı aşamaları

Incremental Development
× Specification development and validation interleaved.
× PD or A can be

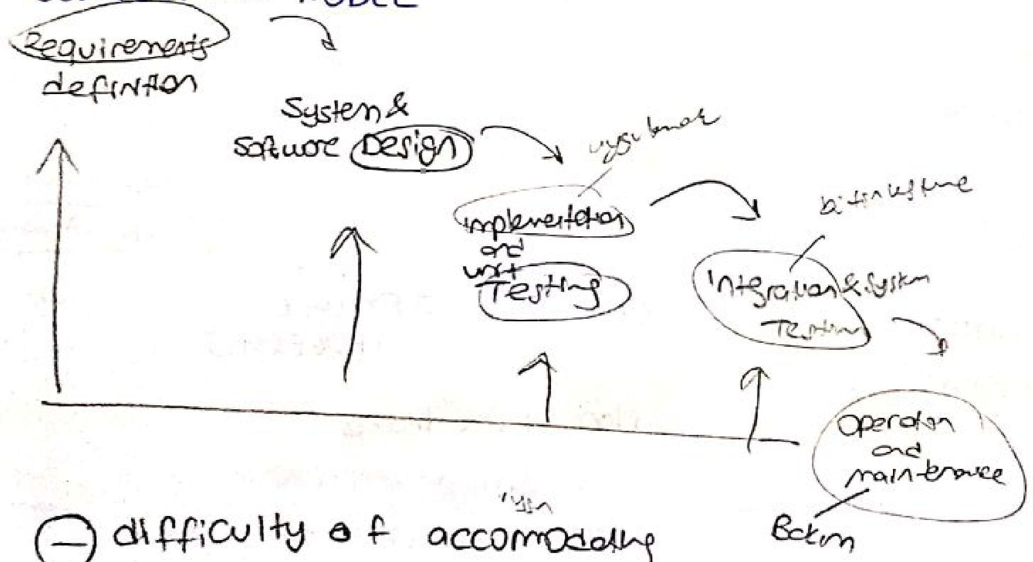
Re-use Oriented Software Engineering
Sistem mevcut componentleri assemble
bileşen ederek birleştirerek

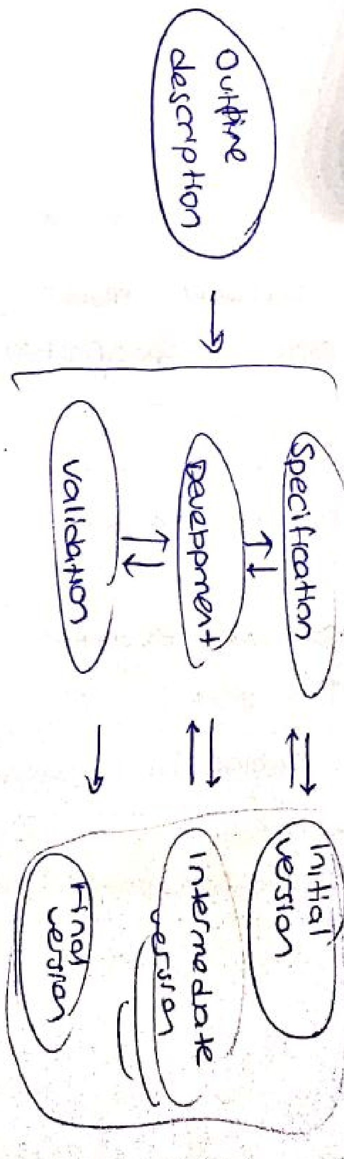
Nerede?

Large systems engineering projects where a system is developed at several sites.
→ bu sistemlerde wf modelin doğası gereği olayı cordinasyon kolaylaşır.

INCREMENTAL MODEL

WATERFALL MODEL


Requirements definition
System & software Design
uygulamak
Implementation and unit Testing
birleştirme
Integration & System Testing
Operation and maintenance
Bakım



⊖ difficulty of accomodating change after the process is underway
Başladıktan sonra değiştirmek zor.
⊖ In principle, bi öncekü adım tamamlanmalıdır
⊖ customer requirements } değişirse, proje değişimi zor

Ne olmalı?

* requirements are well-understood
* changes will be fairly limited during the design process

Incremental Dev. Benefits

* cost of Accommodating changing customer req. is reduce

✓ müşteri değişiklik isteyse, hemen claps

⊕ The amount of analysis and documentation) that has to be redone is much less than WM.

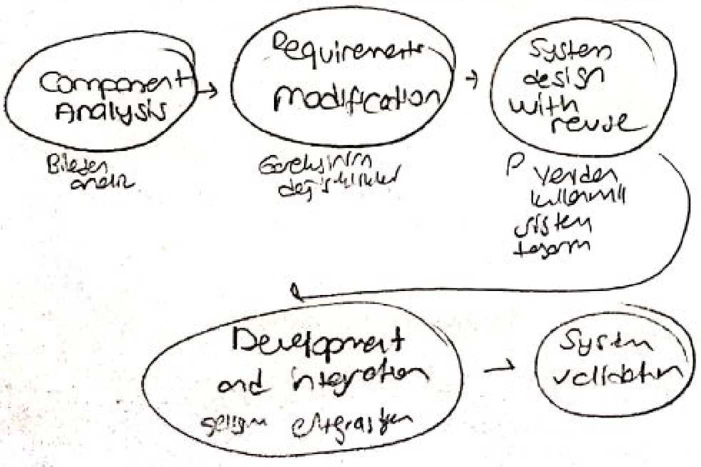
* Easier to get Customer feedback

* Rapid delivery + deployment of useful sw
hızlı
faydalı yazılımın müşteriye dağıtımı

⊖ process is invisible
ilerlemeyi ölçmek için düzenli teslim alınabilir.
Ama bu hızlı ilerleyen sistemde maliyetli iş.

⊖ Yeni özelliklerle sistemin yapısı bozulma eğiliminde.

REUSE ORIENTED SOFTW ENG


Component Analysis
Bileşen analiz
Requirements modification
Gereksinim değişiklikleri
System design with reuse
yeniden kullanımlı sistem tasarım
Development and integration
gelişim entegrasyon
System validation

Nerede?
Belirli bir ortamda kullanılmak üzere yapılandırılmış bağımsız yazılım sistemler

Gerçek yazılım süreçleri
Yazılım sis belirleme → tasarlama → uygulama → test etme

specification Development Validation Evolution	waterfall in sequence	Incremental Development interleaved

YM3

Requirements engineering process

1) Feasibility study
yapılabilirlik
Teknik ve fiziksel açıdan bu sistem inşa edilebilir mi?

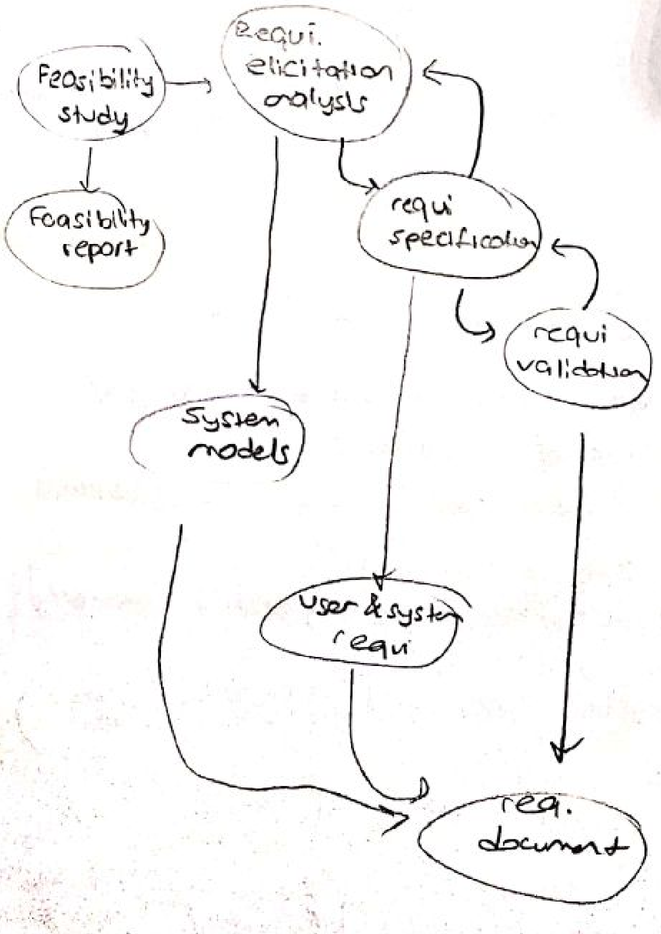
2) Requirements elicitation and analysis.
Gereksinimlerin ortaya çıkarılması ve analizi.

Sistem stakeholders sistemden ne ister, bekler?
paydaşları

3) Requirements specification

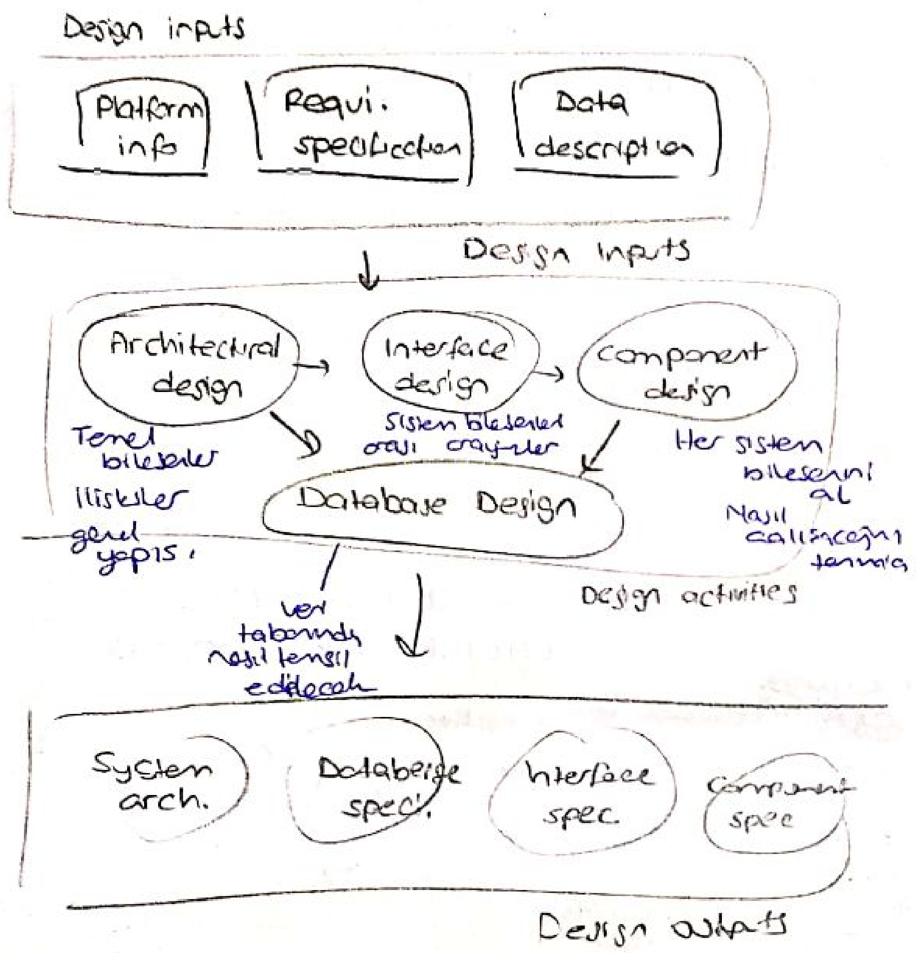
4) Requirements ~~[illegible]~~ validation
Gereksinimlerin geçerliliği kontrol edilecek

THE REQUIREMENTS ENGINEERING PROCESS



implementation: Translate this structure into an executable program.

DESIGN PROCESS



Softwere Validation

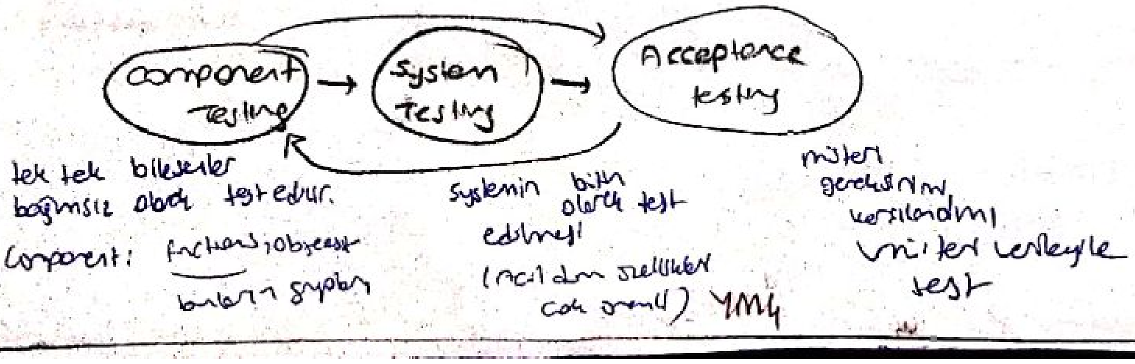
Verification & Validation: V&V

- Sistem özelliklerine uygun mu?
- müşteri gereksinimleri karşılandı mı

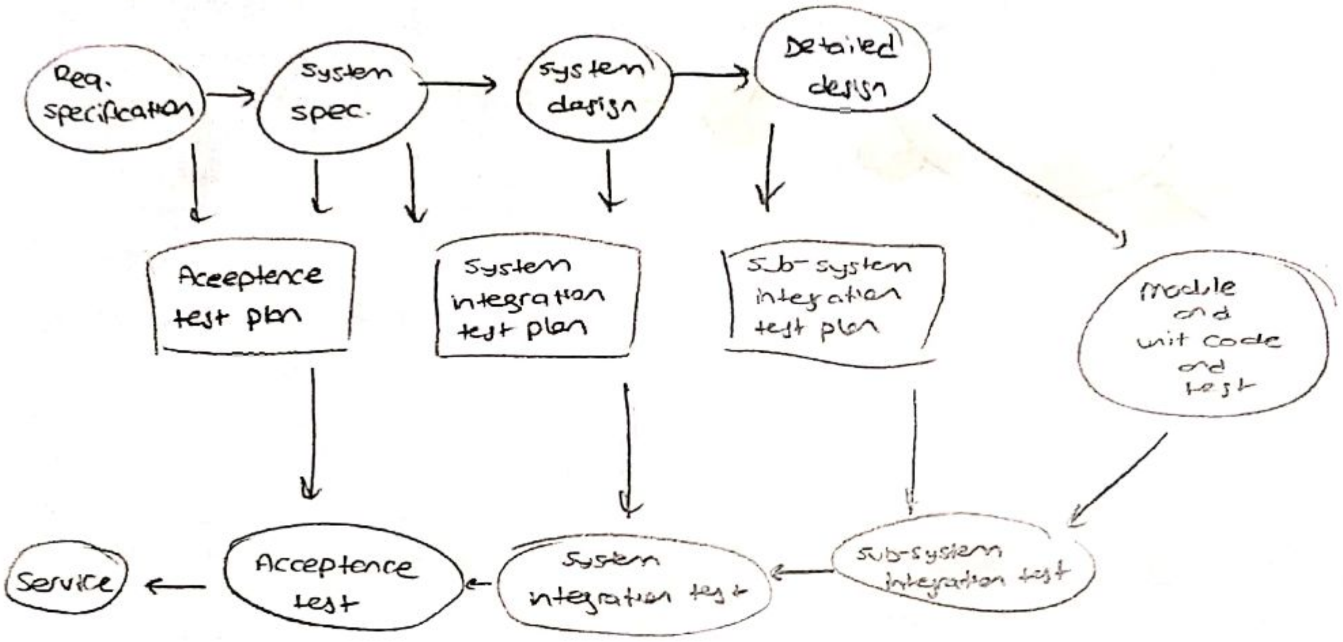
+ Checking → kontrol
+ review processes → inceleme ölçekleri) içerik

+System testing: real datalarla systemi test.) ***

Stage of Testing


Component Testing
System Testing
Acceptance testing
tek tek bileşenler bağımsız olarak test edilir.
Component: functions, objects bunların gruplar
Sistemin bütün olarak test edilmesi
(acil durum özellikleri çok önemli) YMG
müşteri gereksinimi karşılandı mı
müşteri verileriyle test

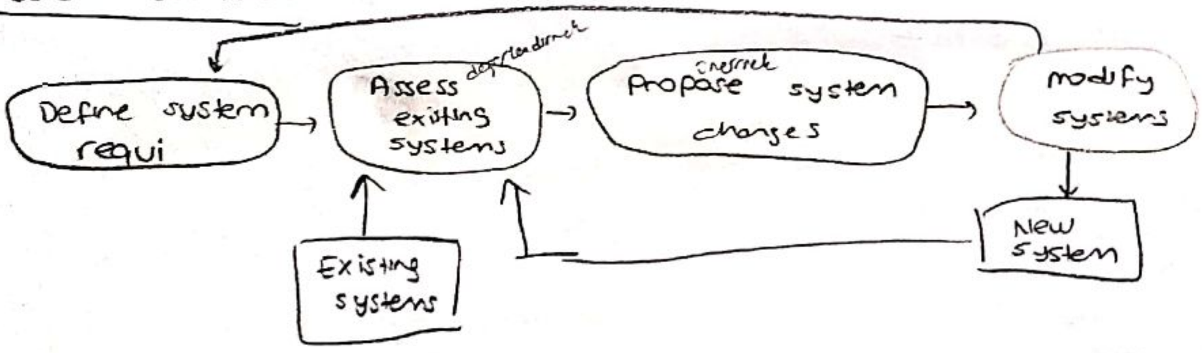
Testing phases in a plan driven software process



Software Evolution

* SW flexible and changeable
* Degisen iş koşullarında, yazılımında değişmesi gerekir.

System Evolution



* Change is irevitable in large SW projects
kaçınılmaz

Platformları değiştirmek uygulama değişikliği maliyet getirir.

* Değişim rework le yol açar.

gereksinimleri yeniden analiz

yeni işlevsellik

× Yeniden işlerre girmeden önce iyi tespit edilmeli ve neler değişecek analizi. Prototip geliştirilebilir.

× Change tolerance, changes 'in düşük maliyete gelmesi için küçük küçük değişiklik yap

Prototype: kavramları göstermek ve tasarım seçeneklerini denemek için kullanılan sistemin ilk sürümü

A prototype can be used in:

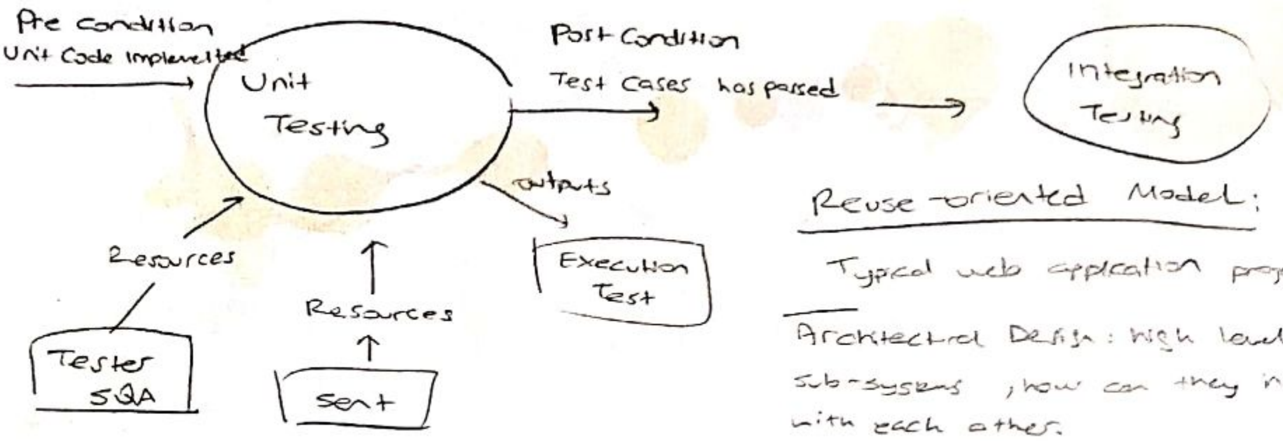
- explore options
- develop UI design
- in testing process to run back-to-back tests

design process

YM5

SOFTWARE ENG

26 Eylül 2015



What are do they doing for success?

Waterfall M. → System design

If we can't finish req. definition we can't ~~p~~ keep going design.

But programers only understand in one way. Design and do it.

Description is not important for them.

You are going to reduce to risk^d

waterfall model purpose ↑

Ama bitirmesi pahalı.

Generally it uses in large ~~systems~~ projects systems.

Incremental Development

For example; money transfer

concurrent activity

Multiple versions is released.

Customer is very tightly involved the project

In waterfall model customer is too late involved the project

But planning is difficult.

Even the customer don't know.

maybe you don't know when it finished.

Refactoring: sonradan değiştirilebilir olmalı.

Reuse-oriented Model:

Typical web application projects

Architectural Design: high level sub-systems, how can they interact with each other.

V & V

Validation:	verification
doing writersystem	Implementing the system rights.

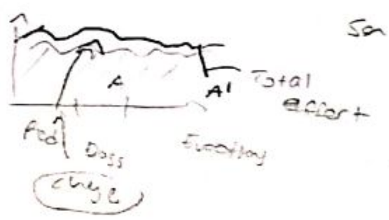
Acceptance Test:
What user will test.
it will answer that

* kesin olmalı * –

Others depens systems.

Acceptance Testing

with users.
And test with interface



Son 3 step atlandı.

A ≥ A'

Benefit of protypeing

Incre P. Seperately requirement

Initial state

UML → Rhapsody
ArgoUML
Rational rose
Paragon

Ne soruları bunlar

umo

Play Station

Incremental
Delivery Problem

⊖ Çoğu system, systemin farklı bölümler tarafından kullanılan temel özellikler gerektirir
→ increment implement edilinceye kadar gereksinimler ayrıntılı bilinmez, ortak özellikleri belirlemek güç

② BOEHM'S SPIRAL MODEL

• Each loop, phase in the process
• Riskler süreç boyu açıkça değerlendir, çöz

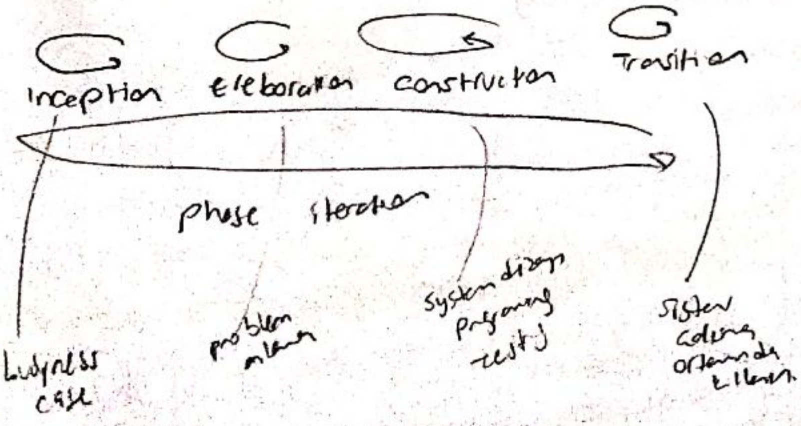
Objective setting
Specific objectives
— Risk assessment and reduction
— Development & validation
— Planning

* Spiral model, insanların yazılımdaki yinelemeyi gerçekçi sağlar ve gelişimine risk odaklı yaklaşımlarını sağlar.
* Çok kullanılmaz

RUP / Rational Unified Process

* Önceki 3 genel işlemin özellikleri bir arada

1) zaman ki aşamaları gösteren dinamik bakış açısı
2) Process aktivitelerini gösteren static
3) iyi pratikler öneren good practice



CHAPTER 3 — AGILE

Rapid Software Development
Hızlı yazılım geliştirme

* Rapid development & delivery en önemli gereksinimdir.
* İşletmeler hızlı değişen bi gereksinim içinde çalışır

* Rapid SW Develop
Specification
Design
implementation
} inter leaved (aynı)

→ sistem bir dizi versiyon olarak gelişir.
→ User interface'ler IDE veya grafiksel ~~ar~~ toolset ile geliştirilir

---

* Tasarımdan ziyade koda odaklan
* SW dev. iterative approach
yinelemeli
* Değişen gelişmelere ayak uydurmak için hızlı delivery

Aim OF AGILE
→ Yazılım süreci genel maliyeti azalt. (Dökümantasyonu sınırlandır.
→ Changing without excessive rework
aşırı yerden yapma

→ Principles of Agile Method ←

① customer involment:
(müşteri katılımı) müşteriler geliştirme sürecince yakından ilgilenmeli.

② Incremental delivery:
Yazılım bazı artışlarla geliştirilir.
Bu artışlar, müşteri istekleri doğrultusu

③ People not process
Development teams skills tanınmalı ve kullanılmalı.
Kırıcı süreçler olmamalı. Herkes kendi çalışma yöntemi geliştirmeli

④ Embrace change (Değişkenliği benimse)
sistemi değişikliği kaldırabilecek şekilde tasarla.

⑤ maintain simplicity: Sadeliğe odaklan
Sistemdeki karmaşıklığı kaldır.

Agile Method Applicability:
(uygulanabilirliği)

× Product development; satış için küçük veya orta ölçekli için geliştirme.

× Yazılımı etkileyen çok fazla dış kural ve düzenlemenin bulunmadığı

× Because of their focus on small, highly-integrated teams, there are problems in scaling agile methods to large systems.

Problems

⊖ Sürece katılan müşterinin ilgisini ayık tutmak zor

⊖ Team members, may be unsuited intense involment

⊖ Multiple stakeholders
(Çoklu paydaşın olduğu yerlerde)
değişikliklere öncelik vermek zor

⊖ Sadeliği korumak zor

⊖ Contracts can be problem.

Çoğu kuruluş, yazılımı sürdürmeye $ > yeni yazılım geliştirme

İki konu:

1) Agile metodla yapılan SW'ler belgeleri en aza indirerek devam edebilir mi?

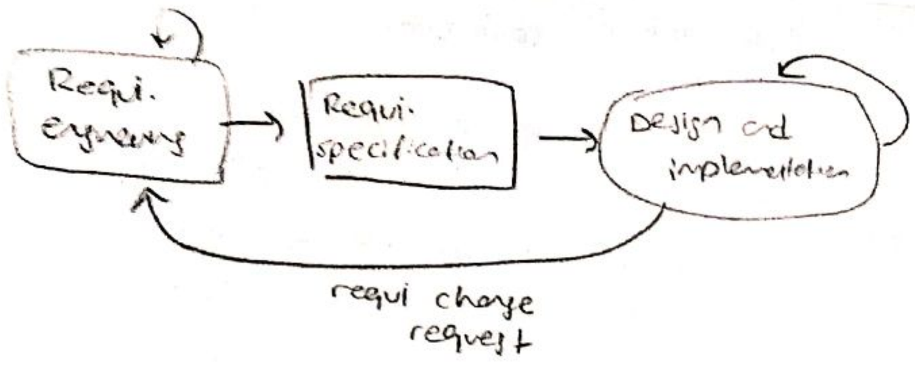
2) Agile systemler müşteri değişen isteklerine göre etkili kullanılabilir mi?

* plan Driven
waterfall or incremental development

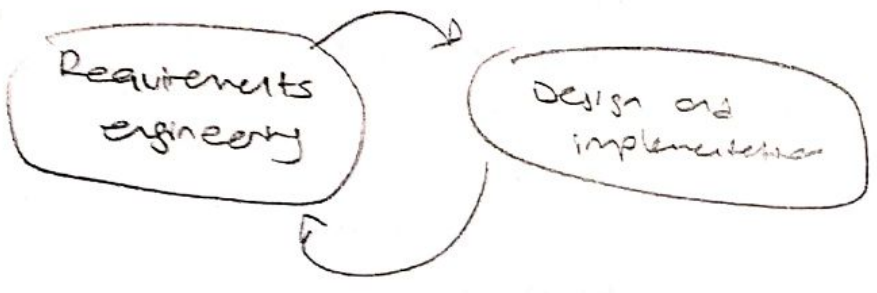
* Yineleme

* Plan odaklı çıktılar

Agile
imp.
design
imp.
testing
} inter leaved



Plan-Based Develop



Agile

→ uygulamaya geçmeden çok ayrıntılı şartname lazımsa → PBD

→ Hızlı teslimat ve hızlı feedback → AD

→ Geliştirilen sistem ne kadar büyük,

when system developed with small co-located team AD

Large development teams → PBD

TYPE OF SYSTEM

PBD → implementasyondan önce çok detaylı analiz

EXPECTED SYSTEM LIFETIME

Long lifetime system → more design documentation

WHAT TECH AVAILABLE

Agile D → Good tools la ihtiyaç var.

HOW DEV. TEAM ORGANIZED?

Dev. team dağıtıksa, veya devten bi kısmı dış kaynak kullanıyorsa

You need to develop design document

CULTUREL ve ORGANISYONAL ETKENLER

Tradional engineering organization have culture PBD.

HOW GOOD DESIGNERS AND PROGR'S.

Agile D → daha yetenekli Higher skills level programcı lazım

IS THE SYS. SUBJECT TO EXTERNAL REGULATION

external regulator → dış yapıcı regulator

onaylı lazımsa, iyi dokümantasyon lazım.

EXTREME PROGRAMING

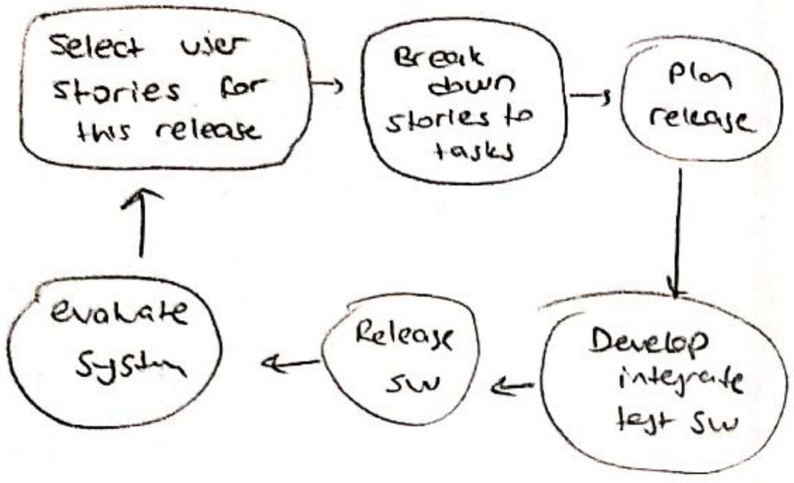
XP → En çok kullanılan agile method

* Yeni sürümler günde birkaç kez oluşturulabilir
* Increments 2 hafta bir müşteriye teslim
* All tests must be run for every build.
  - small frequent system releases
  - customer involvement: full time customer engagement with team

engagement
katılım

•

xp release cycle


Select user stories for this release
Break down stories to tasks
Plan release
Develop integrate test sw
Release sw
evaluate system

Simple $PM = 2.4\,(KDSI)^{1.05} \times M$

Moderate $PM = 3.0\,(KDSI)^{1.12} \times M$

Embedded $PM = 3.6\,(KDSI)^{1.20} \times M$

cocomo

Functional Count F. Point analysis

appeared as the solution to the size measurement problem

$AFP = UFP \times CAF$

adjusted fn point — unadjusted function point — complexity adjustment ~~function~~ factor

pom1.05

$PM = 2.4\,(KDS\,I) \times M$

$3.0\,(KSD\,I)^{1.12} \times M$

$3.1\,(KSD\,I)^{1.20} \times M$

Functional unit		low	Avg	high
EI	external inputs	3	4	6
EO		4	5	7
EQ	external requirement	3	4	6
ILF	internal logic file	7	10	15
EIF	interface files	5	7	10

$EI = 10 \times 3 = 30$

$EO = 13 \times 7 = 91$

$EQ = 4 \times 4 = 16$

$EQ = 2 \times 6 = 12$

$ILF = 2 \;\; 10 = 20$

$EIF = 9 \times 5 = 45$

TOTAL UFP = 214

Q/

EI = 10(low complexity),

EO = 13(high complexity)

EQ = 4 (Avg com)

EQ = 2 (High com)

ILF = 2(Avg Com)

EIF = 9(low com)

SOFTWARE PRODUCTIVITY

- Individual engineers software'i üretme ve dökümantasyon hızları
- Quality assurance : kalite güvencesi
- ESSENTIALLY : useful functionality produced per time unit.
  (zaman birimi başına düşen işlevsel ürün olarak türkiye)

Productivity measures

1) Size related measures : Teslim edilen source code'un satırları olabilir
2) Function related measures : Teslim edilen sw'ın functionality.
   - Function points ✱

MEASUREMENT PROBLEMS

Measure'ın boyut tahmini (kaç function point var?)

Toplam programcı zamanı

Estimating contractor productivity

yüklenici verimliliği

Pricing
1 Market
Rekabet
2
3 Finansal sağlık
4 Belirsizlik

LINES OF CODE

Linear relationship size & volume of documentation

PRODUCTIVITY COMPARISONS

- lower level lang → daha verimli programcı

ama lower level lag daha çok kod satırı

more verbose ayrıntı çoksa verimlilik çok

ayrıntılı kod daha verimli

Function points

- external input, outputs
- user interactions
- external interfaces
- system tarafından kullanılan dosyalar.

Bunların her biri ile bir ağırlık ilişkilendirilir

function point sayısı her satırın ağırlıkla çarpılıp toplanması

$$UFC = \sum (\text{number of elements of given type}) \times (\text{weight})$$

* Function point'ler, projenin karmaşıklığı ile değişir.

* LOC = AVC * numb of FP

Object points

X obje class değil
→ görüntülerin ayrı olan sayısı
→ sistem rapor sayısı
→ Database code için gereken proj modül sayısı

* Erkenden tahmin edilebilir.

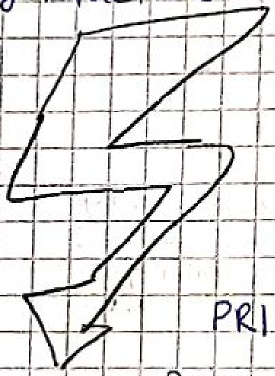
* Bu aşamada code line tahmin çok zor.

Productivity estimate

Real-time embeded system	System Prog.	Commercial App
40/160 LOC/Pmonth	150-400 LOC/Pmonth	200-900 LOC-Pmonth

Productivity'yi etkileyen faktörler

- Application domain experience
- Process quality
- Project size
- Tech support
- Working environment



QUALITY
PRODUCTIVITY

Gereksinimler değişim halindeyse kod sıfırdan yazılmalı olabilir

PRICING TO WIN

Proje, müşterinin harcayacağı paraya mal olur

(+) sözleşme

(−) müşterinin istediği sistem olmayabilir.

Maliyet olduğu yerden itibaren

Estimation Techniques

Algorithmic cost modelling
Expert judgement
Estimation by analogy
Parkinson's Law
Pricing to win

TOP DOWN BOTTOM UP ESTIMATES

genelden özele

bileşen düzeyinde başla

Doğru değer ayrıntı taşıyor

• Sistem mimarisi hakkında bilgi sahibi olmadan kullanılabilir

ayrıntı bilgi yoksa bu mantıklı

ALGO COST MOD.

$$Effort = A \times Size^{B} \times M$$

A: organizasyon

B: değişken, katsayı

M: [illegible]

* Code size

cocomo model

- Deneysel model
- Proje deneyimi ***
- yazılım satıcısına bağlı olmayan "independent"
- well documented

cocomo 81
→ waterfall process
→ Yazılım Oldan gelişti

cocomo 81

Pro complexy	Formul	Description
simple	$PM = 2.4 (KDS\,1)^{1.05} \times M$ Small team	well understood app.
moderate	$PM = 3.0 (KDS\,1)^{1.12} \times M$ member have limited experience	complex pro.
Embedded	$PM = 3.6 (KDS\,1)^{1.20} \times M$	complex

Cocomo 2

sub models:

Application composition model:
yazılım mevcut parçalardan oluşturulduğunda kullanılır

Early Design model
Requi. hazır, tasarım [illegible]

Reuse model
Yazılım [illegible] öğelerin entegrasyon [illegible]

Post Architecture model
[illegible] [illegible] [illegible] [illegible]
[illegible]

Early Design model
1) requi. karar kıldığında
2) algo. modeller için std. formüle dayanır

$PM = A \times Size^{B} \times M$

$m = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$

MULTIPLIERS

RCPX	product reliability and complexity
RUSE	reuse required
PDIF	platform difficulty
PREX	personel experience
PERS	personal capability
SCED	required schedule
FCIL	team support facilities

Post achi. Level
kod boyutu tahmini:
- Geliştirilecek yeni kod satırı sayısı
- Reuse modelle tahmini yeni kod sayısı
- Gereksinim değişti, değişmesi gereken kod satırı

13

1 → $FP = CFP \times CAF$

unadjusted function point

adjusted function point

	L	Avg	H
EI	3	4	8
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

3 $EI = 10 (L) = 30$

2 $EO = 13 (H) = 91$

1 $EQ = 4 (AV) = 16$

4 $EQ = 2 (H) = 12$

5 $ILF = 2 (AV) = 20$

0 $EIF = 8 (L) = 40$

$\boxed{214} = CFP$

214×0.84

2 $CAF = 0.65 + 0.01 \times \Sigma F$

RCAF

$RCAF = 19$

$0.84 = 0.65 + 0.19$

$FP = CFP \times CAF$

sonda eklé A

$30.000 = 30$

$/1000 = size$

1.05 - 1.20 kadar değişir

$Effort = A \times Size^B \times M$

$= 2.94 \times$

Person / months

B → precidence + Dev. flex + Team cohesion + process maturity + Arc. risk

$+ \frac{Toplam}{100} + 1.01 \Rightarrow B$

Exponent Value

I Software E. %50 → UmL

· when you given a description you will try to turn that descrip into umL
· Kitaptaki örneklere bak!

Critical System Specification:
Propositional logic ⇒

S: The book is in the stock.
R: The book is on reserve.
L: The book is on loan.
Q: The book is requested.

*A book can only be on stock, on reserve or loaned.

- $S \wedge \neg (R \vee L)$
negation
- $R \wedge \neg (S \vee L)$
- $L \wedge \neg (S \vee R)$

If a book is on the stock or on reserve, then it can be requested.

- $Q \Rightarrow (S \vee R)$

To prove: $L \Rightarrow \neg Q$ ✓

$\neg (L \Rightarrow \neg Q)$ (Assume negation for proof by contradiction)

$\neg (\neg L \vee \neg Q)$ (Rewriting)

$L \wedge Q$ (De morgan)

L (Simplifying)

~~$\neg Q$ (Rewriting)~~

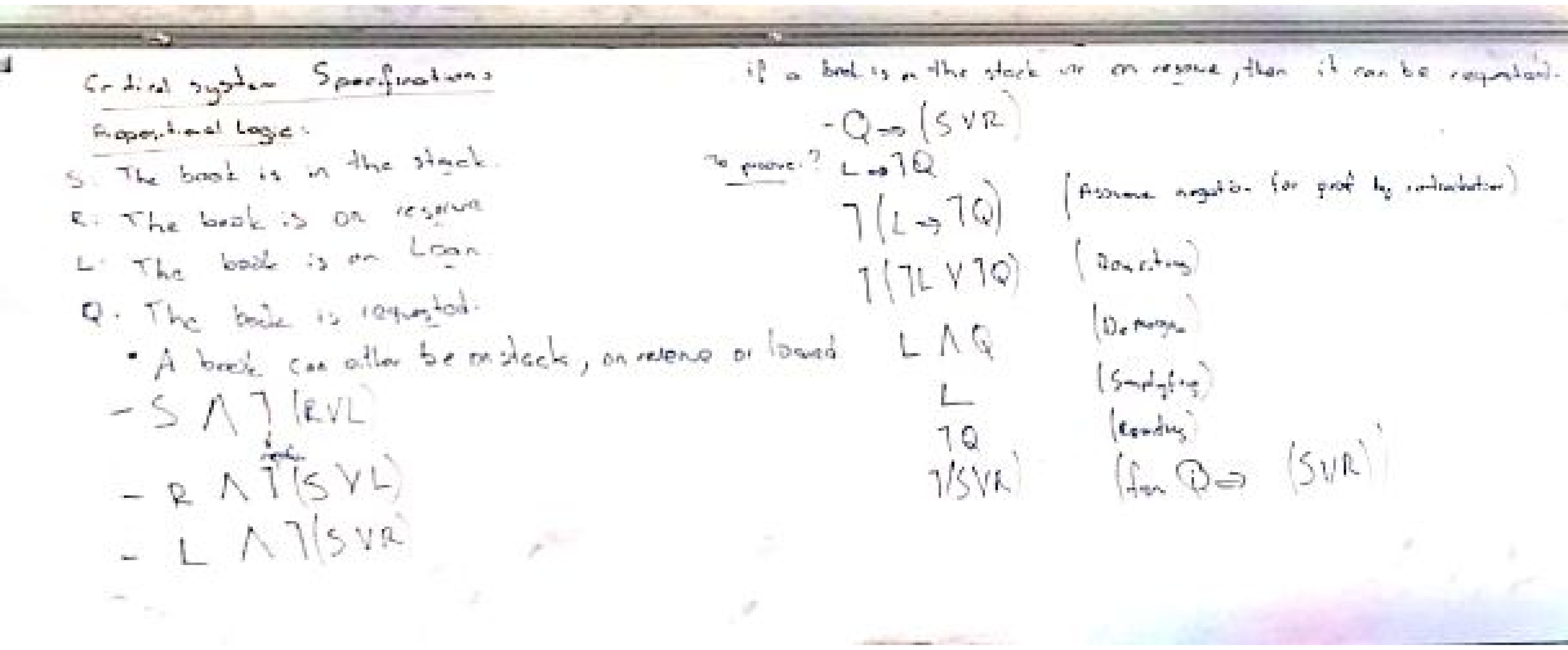
~~$\neg (S \vee R)$ (from $Q \Rightarrow (S \vee R)$)~~

$\neg (S \vee R)$ ($L \Rightarrow \neg (S \vee R)$)

contradiction

Q

$S \vee R$



Critical system Specifications

Propositional Logic:

S: The book is in the stack.

R: The book is on reserve

L: The book is on Loan

Q: The book is requested.

- A book can either be on stack, on reserve or loaned

$- S \land \neg(R \lor L)$

$- R \land \neg(S \lor L)$

$- L \land \neg(S \lor R)$

if a book is in the stack or on reserve, then it can be requested.

$- Q \Rightarrow (S \lor R)$

To prove: ? $L \Rightarrow \neg Q$

$\neg(L \Rightarrow \neg Q)$ (Assume negation for proof by contradiction)

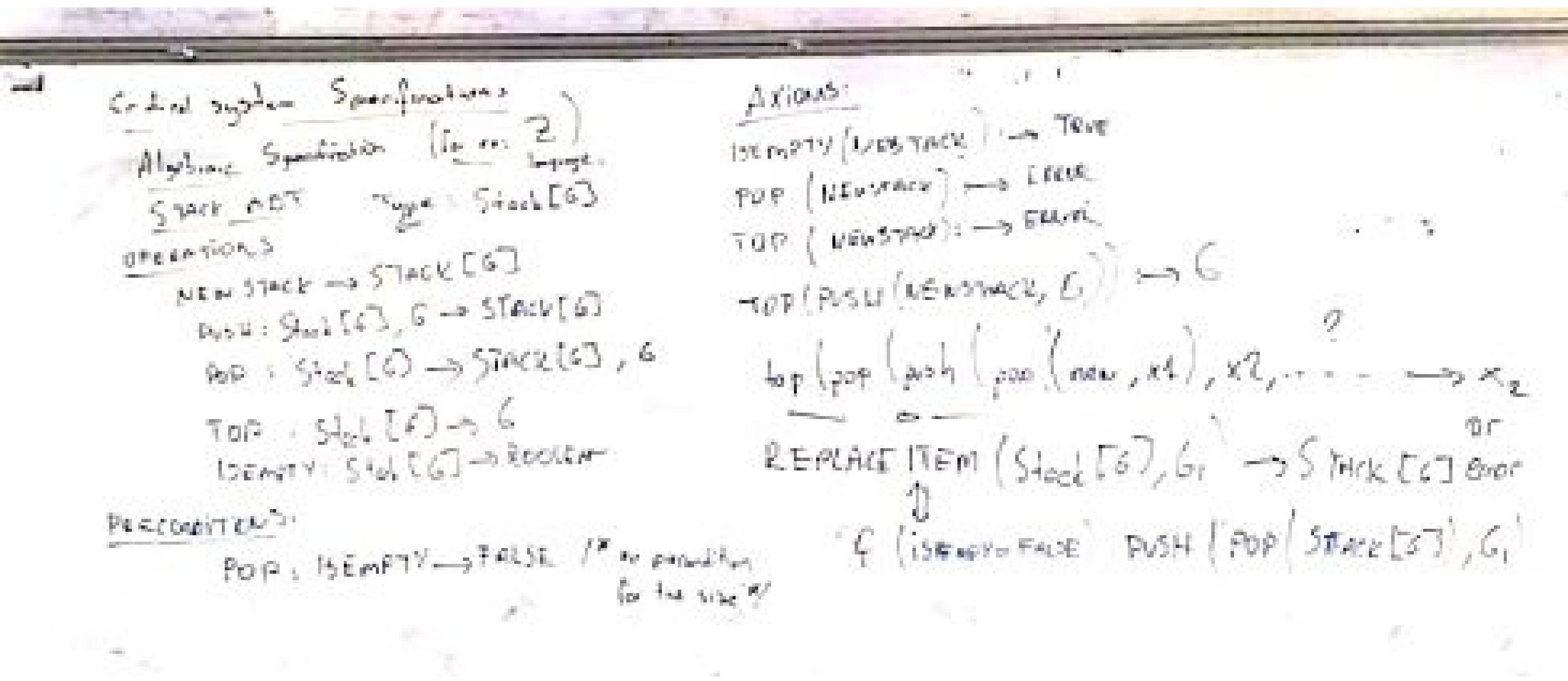
$\neg(\neg L \lor \neg Q)$ (Rewriting)

$L \land Q$ (De Morgan)

L (Simplifying)

$\neg Q$ (Reading)

$\neg(S \lor R)$ (from $Q \Rightarrow (S \lor R)$)



Critical system Specifications

Algebraic Specification (Is an Z language)

Stack_ADT Type : Stack[G]

OPERATIONS

NEW STACK → STACK[G]

PUSH : Stack[G], G → STACK[G]

POP : Stack[G] → STACK[G], G

TOP : Stack[G] → G

ISEMPTY : Stack[G] → BOOLEAN

PRECONDITIONS:

POP : ISEMPTY → FALSE /* no precondition for the size */

AXIOMS:

ISEMPTY (NEWSTACK) → TRUE

POP (NEWSTACK) → ERROR

TOP (NEWSTACK) : → ERROR

TOP (PUSH (NEWSTACK, G)) → G

top (pop (push (pop (new, x1), x2, . . . ? . . . → x_2

REPLACE ITEM (Stack[G], G_1) → STACK[G] or Error

⇕

if (ISEMPTY = FALSE) PUSH (POP (STACK[G]), G_1)