By : NongNoon <3

บทที่ 1 Software Engineering Principle

Software Design : กระบวนการที่เราใช้เทคนิคความรู้เกี่ยวกับsoftwareมาสร้าง software ทั้งภายนอกและภายในเพื่อให้ software มีประสิทธิภาพสูงสุดและใช้ทรัพยากรอย่างคุ้มค่า

Software Development : กระบวนการที่แปลงความต้องการ user เป็น product

Software engineering : กระบวนการแก้ไขปัญหาของลค.โดยก.ทำงานเป็นระบบ โดยเป็นการทำแบบ high-quality ที่มี cost time จำกัด

เป้าหมายซอฟต์แวร์ที่มีคุณภาพดี

Systematic : สร้างซอฟต์แวร์อย่างเป็นขั้นตอนเป็นระบบ

Disciplined : มีวินัยทำตามขั้นตอนที่ได้วางเอาไว้ Quantifiable : สามารถวัดเป็นตัวเลขได้

Software development life cycle (SDLC)

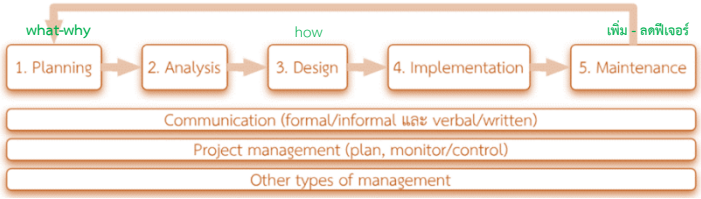
การวิเคราะห์ความต้องการ (requirement analysis)

what-why how

- การออกแบบ (design) - การดำเนินการสร้าง (implementation) - การทดสอบ (testing)

อาจจะเจอ bug ทำให้มีการเพิ่ม - ลดฟีเจอร์

- การนำไปใช้และบำรุงรักษา (deployment and maintenance)



- การทำโมเดล (modeling) : requirement analysis + design -> (what + how)

Requirement analysis และ design

- หลังจากผู้บริหารตกลงให้เดินหน้ากับโปรเจค = เข้าสู่กระบวนการสร้างซอฟต์แวร์ (SDLC)

- เริ่มด้วย requirement analysis และ design คืออธิบายการสร้างซอฟต์แวร์ที่ต้องการ

Requirement analysis (what --- problem) Design (how --- solution)

- การวิเคราะห์ requirement และการออกแบบ

-- สอนได้แค่แนวทางและหลักการ -> requirement - เก็บข้อมูลให้ตรงกับความต้องการของผู้ใช้จริง ๆ, design - หลักการออกแบบต่างๆ

-- อาศัยประสบการณ์และการฝึกฝน

- อธิบายการวิเคราะห์และออกแบบ (ด้วย model หรือ diagram ต่างๆ) = สอน รูปแบบหลักไวยกรณ์, เมื่อคล่องจะทำให้สามารถสื่อสารได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

Software modeling

- Representation ของระบบ = ใช้สัญลักษณ์มาตรฐาน, สื่อแต่ละสื่อจะอธิบายแง่มุมของแนวคิดที่ต่างกัน, ช่วยในการสื่อสารแนวคิดให้ stakeholder อื่นๆ

- ในช่วง requirement เรียกว่า requirement modeling = model หลักและไม่ละเอียด

- ในช่วง design เรียกว่า design modeling = มีรายละเอียดเพิ่มขึ้นใกล้เคียงกับสิ่งที่จะ implement มากขึ้น ไม่รู้ว่าแต่ละ method ทำงานยังไง อธิบายการทำงานของ method (behavioral)

- 2 มุมมอง = โครงสร้าง, ขั้นตอนการทำงานตรรกะการประมวลผล (จำเป็นต้องมีทั้ง 2 มุมมอง = ทำให้เห็นภาพซอฟต์แวร์ได้ดีก่อนสร้างจริง)

แนวทางการสร้างงาน

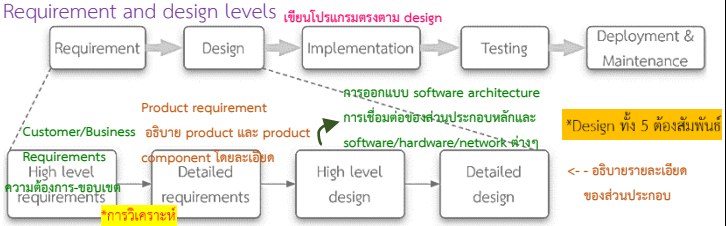
แนวทางการพัฒนาซอฟต์แวร์ (paradigm) -> แบบ structured หรือ procedural อยู่ในขั้นตอน Data and process analysis and modeling

UML (Unified Modeling Language)

- UML as sketch : explore problem or solution - UML as blueprint : detailed design for reverse engineering or code generation (forward engineering)

• UML as programming language : executable specification (อยู่ในขั้น R&D)

Requirement and design levels เขียนโปรแกรมตรงตาม design



- Design อื่น : Input and output design, User interface design, Data design (DB)

การ refactor : การเปลี่ยน code ให้มีค.ยืดหยุ่น

Software testing (การทดสอบ), Software inspection (การตรวจทาน)

การทดสอบไม่สามารถหา requirement ที่ขาดหายไปได้ -> ทำให้หา defect ในช่วง design ได้

การนำไปใช้ (deployment) -> (Activate = ใส่ code)

บำรุงรักษา -> Repair (แก้ไขให้ถูก), Enhance (ป้องกันไม่ให้เกิดขึ้น)

บทที่ 2 Requirement analysis and engineering

Shared understanding of what needs to be built -- อยู่ในรูปเอกสารและ diagrams เพื่อให้ทุก stakeholders เข้าใจซอฟต์แวร์ที่กำลังจะสร้างแบ่งออกเป็น 2 แบบ - Functional requirements -> Behaviors and features: การคำนวณ การประมวลผล input / output

- Nonfunctional requirements เช่น Performance : ต้องทำงานได้อย่างรวดเร็ว Reliability and availability : downtime, fault rate น้อย,Maintainability : ปรับปรุง แก้ไขได้ง่าย,Portability : ใช้ได้กับหลาย platform หรือ environment,Usability : ใช้งานง่าย GUI สวยงาม

Requirement development (กระบวนการที่ทำให้ได้มาซึ่ง requirement) 4 ขั้นตอน elicit, analyze, document, validate (epic > feature > user) *การวิเคราะห์เครื่องมือดูจากกราฟข้างบน

Requirement management : การทำความเข้าใจและตกลงกันในเรื่อง requirement

Stakeholders : การสัมภาษณ์และการ,การสำรวจโดยใช้แบบสอบถาม , การสังเกตการทำงาน

บทที่ 3 Software Process - Scrum Overview

Waterfall model : เข้าใจและนำไปปรับใช้ได้ง่าย,จัดการ resource ได้ง่ายแต่ต้องมี requirement ที่ชัดเจน,ไม่มี feedback,อาจพบ defect ช้า,ไม่มีการทำงานแบบ parallelism

Iterative development : การแบ่งการพัฒนาออกเป็นรอบๆแต่ละรอบจะทำครบหรือเกือบครบทุก phase และได้ผลเป็น software

Incremental development : เป็น iterative development ชนิดหนึ่งที่แต่ละรอบจะมีบางส่วนของ working software ถูก release ออกมา **เพิ่มฟีเจอร์เรื่อยๆเป็น Working software

Spiral model : สามารถจัดการกับ risk ได้ตั้งแต่เนิ่นๆ,ได้รับ feedback ตั้งแต่เนิ่น,รองรับความเปลี่ยนแปลงที่เกิดขึ้น แต่ ซับซ้อน ทำให้การจัดการยาก,ไม่เหมาะกับโปรเจคเล็กๆ

Product owner (PO) คือคนที่จะบอกได้ว่า ความต้องการมีอะไรบ้าง แล้วเป็นคนเรียง priority มาให้เราได้ด้วย ว่าอะไรทำก่อนหลัง

Scrum master คือคนที่จะรู้หมดว่าใครทำอะไร หรือเป็นคนที่ตั้งโครงขึ้นมาแล้วถอยหลังออกมา เพื่อมองว่า ใครต้องทำอะไร รวมทั้งจะต้องเป็นคนที่ปกป้องทีม เพื่อเป็นปากเป็นเสียงแทนทีม

** KISS :: Keep It Simple, Stupid = รูปแบบการทำงานของ code **

Team : จะทำงานแบบ Self-Management ซึ่งในหนึ่งทีมจะประกอบด้วยคนประมาณ 5-9 คน และรวมทุกตำแหน่งทั้ง Designer, Programmer, UI/UX, Testing เข้าด้วยกัน เพื่อให้ทีมหนึ่งทีมสามารถทำงานตั้งแต่ต้นจนจบได้ด้วยตัวเอง โดยไม่ต้องข้ามแผนก ควรมี รายละเอียด, Estimated(ประเมินitem), Emergent(ค.เร่งด่วนไว้วัดการเปลี่ยนลำดับ back lock)

Product Backlog : ใน Scrum คือลิสต์ของงานใน Product นั้น must choose would,บอกเลข

Sprint Phase : Agile นั้น เน้นการส่งงานให้เร็วและบ่อย ซึ่ง Period นั้นจะเรียกว่า Sprint โดยมีกำหนดประมาณ 1-4 สัปดาห์ โดยเป้าหมายของ Sprint คือการ Deliver บางสิ่งบางอย่างให้สำเร็จ ซึ่งเมื่อจบ Sprint ก็จะมีการ Review ผลงาน (Sprint Review) ให้กับคนอื่นๆ ที่เกี่ยวข้องอาจจะเป็นทีมเซลล์ Users หรือลูกค้า เพื่อให้รับทราบถึงความคืบหน้าของโปรเจคอยู่เรื่อยๆ

Daily Scrum Meeting : ในทุกๆ เช้าทีมจะมีการประชุมสั้นๆ 10-15 นาที เพื่อบอกว่าเมื่อวานทำอะไรวันนี้จะทำอะไร และมีปัญหาอะไรบ้าง เพื่อให้การทำงานในทุกๆ วันเป็นไปอย่างราบรื่น ,รู้ว่ากำลังเดินเข้าสู่เป้าหมายหรือยัง และมีการแก้ไขปัญหาอย่างต่อเนื่อง Sprint goal : ควรโฟกัสตรงไหนในแต่ละ sprint

Sprint Planning : งานที่จะถูกทำใน Sprint นั้นได้รับการวางแผนใน Sprint Planning การวางแผนนี้ทำขึ้นโดยความร่วมมือของสมาชิกทีม Scrum ทั้งหมด

Sprint Review : ทำหลังจบทุก Sprint โดยจะพูดถึง Feature ที่เสร็จแล้วใน Sprint ก่อนๆและ Sprint นี้ เพื่อให้ Product Owner หรือ User ทดลองใช้งานและ Feedback กลับมา

Sprint Retrospective : ช่วงเวลาในการที่ประชุมและบอกให้ได้ว่า Sprint ที่ผ่านมานั้น อะไรทำได้ดี – ไม่ดี – และอะไรที่อยากจะนำมาใช้ ในการทำงานรอบถัดไปเพื่อพัฒนาการทำงานในรอบถัดๆไปให้ดีขึ้น

Product Backlog Refinement : การแตก Item ขนาดใหญ่, วิเคราะห์, ประเมินใหม่, จัดลำดับความสำคัญใหม่ สำหรับ sprint ต่อไป

บทที่ 5 Test-driven development (TDD) - Review jUnit

```
@Test
void testDistanceFromOrigin() {
Point p1 = new Point(3, 4);
double dist = p1.distanceFromOrigin();
assertEquals(5.0, dist, 0.001);
// expected ต่างจาก actual ได้เท่าใด
assertEquals(20, dist);
}
```

```
@Test
void withdrawMoreThanBalanceThrownException() {
    BankAccount account = new BankAccount(100);

    Throwable exception =
            assertThrows(InsufficientFundException.class,
                    () -> { account.withdraw(500); });

    assertEquals("Amount withdrawn must be less than balance",
            exception.getMessage());
}
```

บทที่ 6 Dependency Injection & Spring Framework

Dependency Injection -> คลาสผู้ใช้ object ไม่สร้าง obj ที่ต้องการเองรับ obj ผ่าน constructor/setter methods ทำให้มีการสร้าง obj อยู่ในที่ที่เดียวเท่านั้น ทำให้ สลับผลัดเปลี่ยน implementation ได้ง่าย, loosely Coupled (แต่ละ module เป็นอิสระ ไม่ขึ้นต่อกัน)

Spring IoC Container : เป็นเพียงหลักการที่ต้องการลดความซับซ้อนของ component หรือ object ต่างๆที่มันผูกติดกันมากจนเกินไป จนทำให้มัน maintain และ test ยาก

pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.1.1.RELEASE</version>
    </dependency>
</dependencies>
```

main / resources / config.xml

```
<bean id="dataS" class="atm.DataSource">
    <constructor-arg value="customers.txt"/>
</bean>

<bean id="bank_" class="atm.Bank">
    <constructor-arg ref="dataS"/>
</bean>
```

Ref = class
Value = ค่าต่างๆ ตัวแปร

JAVA

```
public class MovieLister {

    private MovieFinder finder;

    public MovieLister() { ..... }

    public void setFinder(MovieFinder finder) {
        this.finder = finder;
    }
```

```
<bean id="lister" class="springlesson.movie.MovieLister">
    <property name="finder" ref="csv-finder"/>
</bean>
```

```
@Bean
public CashRegister cashRegister() {
    return new CashRegister(caTaxCalculator());
```

```
<context:annotation-config/>
<context:component-scan base-package="ku.cashregister"/>
```

Main.java

```
ApplicationContext context =
        new ClassPathXmlApplicationContext("config.xml");
ATMSimulator atmSimulator = context.getBean("ATMSimulator_", ATMSimulator.class);
atmSimulator.run();
```

Lab 01 - Git Basic + Branch and Merge

git init คำสั่งนี้จะสร้างแฟ้มข้อมูลย่อยชื่อ .git ///// git status ตรวจสอบสถานะ

git add . หรือ filename เพิ่มไฟล์ //// git commit -m "text" /// git push ขึ้น git

git config user.name "John Smith" -- git config user.email "john.s@ku.th" ยืนยันตัว

git log --all --decorate --oneline –graph -- ดูการ commit ทั้งหมดได้ commit-id ด้วย

เราสามารถเอาเวอร์ชันเก่ากลับมาได้ โดยจะทำได้ 2 แบบ –> กลับมาทั้งหมด (git checkout <commit-id>) , กลับมาบางไฟล์ = (git checkout <commit-id> <ชื่อไฟล์>)

git branch name สร้าง branch //// git branch มี branch ใดบ้าง เราอยู่* branch ไหน

git checkout changeB เปลี่ยน branch ///// การ merge โดยให้อยู่ใน master แล้วค่อย merge feature เข้ามา -> git checkout master -> git merge feature (ไม่ conflict)

ถ้า conflict พอ merge แล้วจะมีแจ้งเตือนให้เข้าไปที่ไฟล์ที่แจ้งเตือน เลือกแบบที่เราต้องการ แล้วทำการ add+commit แล้วก็ git push origin master ได้เลย (ยังไม่ต้อง push ก็ merge ก่อนได้)

git pull origin master -- Pull แบบมี conflict

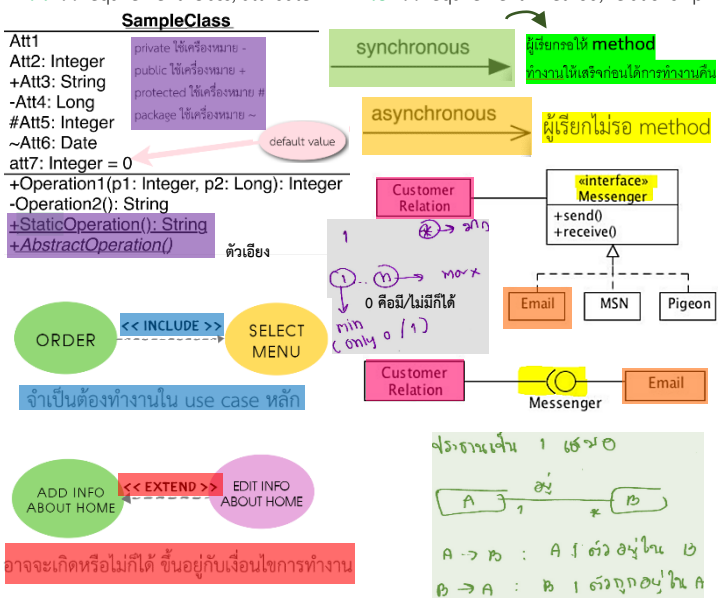
--- User Story --- (priority +estimate = เวลา)

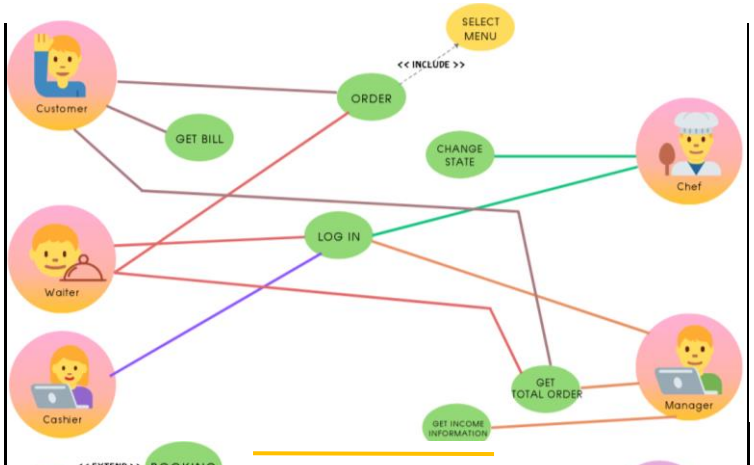
epic (ครอบคลุมหลาย feature) > feature (ครอบคลุมหลายกรณีของ user) > user (ระดับบุคคล)

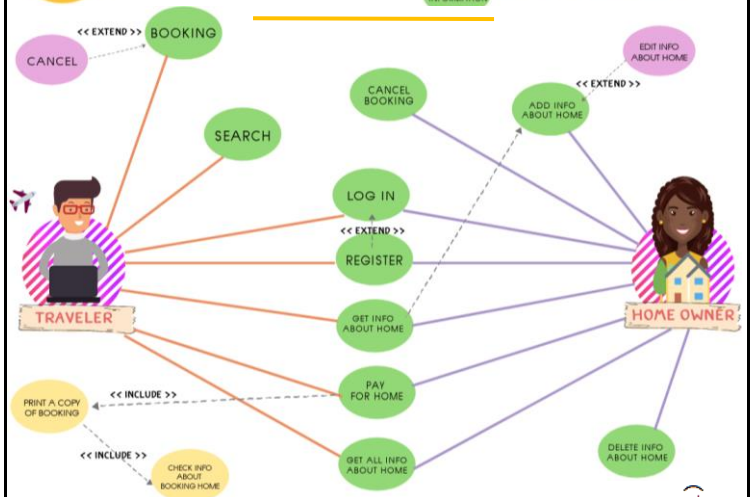
Epic	Improve User Experience	Active	Fabrikam\P1 1	Fiber Suite
Feature	Integrate client application with popular email clients	Active	Fabrikam\P1 1\Sprint 1	Fiber Suite
User Story	Implement a factory which abstracts the email client	Active	Fabrikam\P1 1\Sprint 1	App
User Story	As a user, I can select a number of support cases and use...	Active	Fabrikam\P1 1\Sprint 1	App
Feature	Integrate client app with IM clients	Active	Fabrikam\P1 1\Sprint 1	Fiber Suite
Feature	Emoticon feedback enabled in client application	Active	Fabrikam\P1 1\Sprint 1	App
User Story	As a user, I can select an emoticon and add a short descri...	Active	Fabrikam\P1 1	App

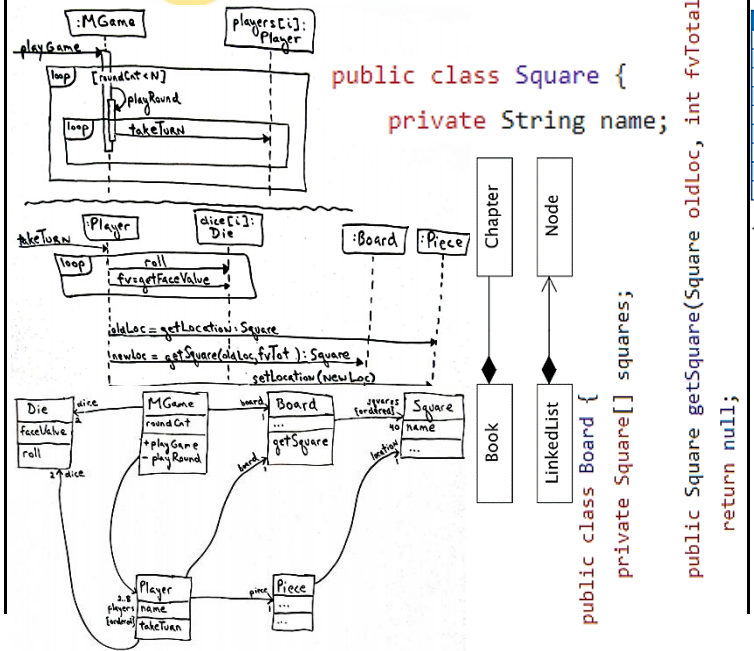
บทที่ 7 UML Class Modeling

คำนาม ใน requirement=class, attribute --- คำกริยา ใน requirement=method, relationship










```
public class Square {
    private String name;
public Square getSquare(Square oldLoc, int fvTotal)
public class Board {
    private Square[] squares;
    return null;
```

```
public class MGame {
    private static int N = 25;

    private int roundCnt;
    private Board board;
    private Player[] players;
    private Die[] dice;

    public void playGame() {
        for (roundCnt = 0; roundCnt < N; roundCnt++
            playRound();
        }
    }

    private void playRound() {
        for (int i = 0; i < players.length; i++) {
            players[i].takeTurn();
```

```
public class Die {
    private int faceValue;

    public void roll() {

    }
    public int getFaceValue
        return faceValue;
```

```
public class Player {
    private String name;
    private Die[] dice;
    private Board board;
    private Piece piece;
public void takeTurn() {
    int fvTotal = 0;

    for (int i = 0; i < dice.length; i++) {
        dice[i].roll();
        fvTotal += dice[i].getFaceValue();
    }

    Square oldLoc = piece.getLocation();
    Square newLoc = board.getSquare(oldLoc, fvTotal);
    piece.setLocation(newLoc);
```

```
public class Piece {
    private Square location;

    public Square getLocation() {
        return location;
    }

    public void setLocation(Square location
        this.location = location;
```

```
Set setA = new HashSet();
int a[]=new int[5];        // add,remove
ArrayList<String> alist=new ArrayList<String>();
```

Burn down Chart เป็นกราฟง่ายๆที่ใช้บอกเราว่า “เราทำงานเสร็จไปแล้วเท่าไรและเราเหลืองานที่ต้องทำอีกเท่าไร”

Burn Up Chart ซึ่งอันนี้คิดตรงกันข้าม ก็คือเริ่มต้นจากความว่างเปล่า ให้หยิบงานที่เสร็จแล้วมากองเพิ่มขึ้นไปเรื่อยๆ

Extreme Programming = ก.พัฒนาSWตามค.สนใจคนในทีม (Communication, Simplicity, Feedback, Courage)

Agile เป็นกระบวนการที่จะช่วยให้ทำงานได้เร็วขึ้น โดยลดการทำงานที่เป็นขั้นตอนและงานด้านเอกสารลง และมุ่งเน้นเรื่องการสื่อสารกันในทีมให้มากขึ้น เพื่อร่วมกันพัฒนาผลิตภัณฑ์ให้เร็วขึ้น พร้อมนำมาทดสอบ และเก็บผลตอบรับต่าง ๆ เพื่อกลับไปแก้ไขปรับปรุง ซึ่งจะทำให้สามารถพัฒนาผลิตภัณฑ์ได้รวดเร็วและตอบโจทย์ผู้ใช้งานมากขึ้น

Agile manifesto คนและการมีปฏิสัมพันธ์กัน มากกว่าการทำตามขั้นตอนและเครื่องมือ,ซอฟต์แวร์ที่นำไปใช้งานได้จริงมากกว่าเอกสารที่ครบถ้วนสมบูรณ์,ร่วมมือทำงานกับลูกค้า มากกว่าการต่อรองให้เป็นไปตามสัญญา,การตอบรับกับการเปลี่ยนแปลง มากกว่าการทำตามแผนที่วางไว้

Tasks	D1	D2	D3	D4	D5	D6	D7	D8	D9	...	D15
Task 1	8	4	4	2							
Task 2	12	8	16	14	9	6	2				
Task 3	5	5	3	3	1						
Task 4	7	7	7	5	10	6	3	1			
Task 5	3	3	3	3	3	3	3				
Task 6	14	14	14	14	14	14	14	8	4		
Task 7						8	6	4	2		
Tasks 8–30	151	139	143	134	118	99	89	101	84		0
Total	200	180	190	175	155	130	115	113	90		0

- burndown chart : plot “Total” (of remaining effort each da