SWE3002-42: Introduction to Software Engineering

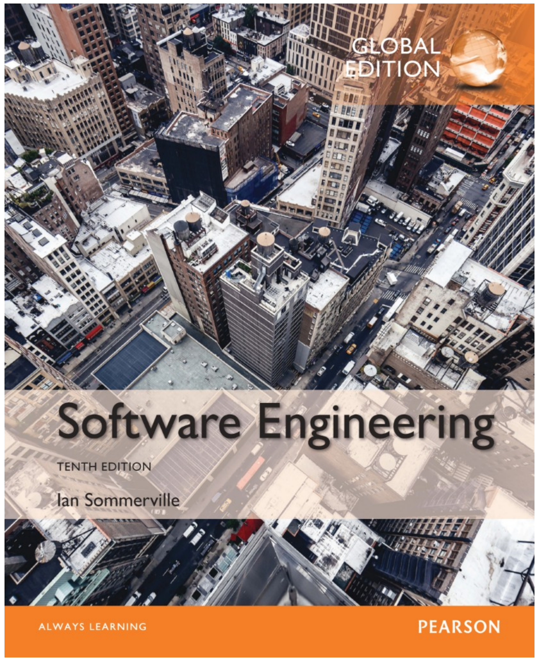
Lecture 1 – Introduction

Sooyoung Cha

Department of Computer Science and Engineering

Chapter1. Introduction

Topics covered

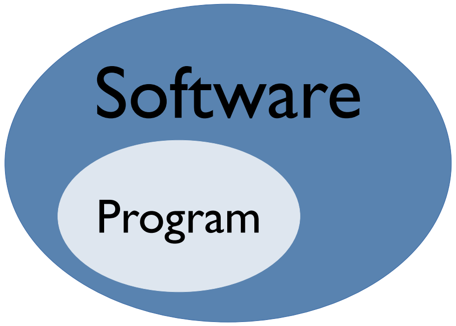


Chapter 1. (p.15 ~ p.42)

What is Software?

[Program ⊆ Software]

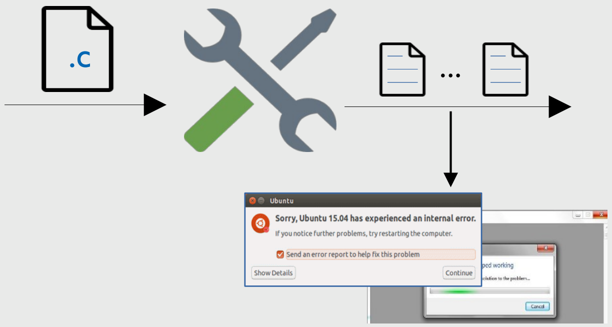
Program	**A collection of instructions** executed by a computer, including compiled code as well as source code
Software	Not only programs (codes), **but all associated documentation** required during software development
Software e.g.	Testing results (e.g., code coverage, bug-finding results), user manual, documents from each development stages, ...



What is Software?

[Program]

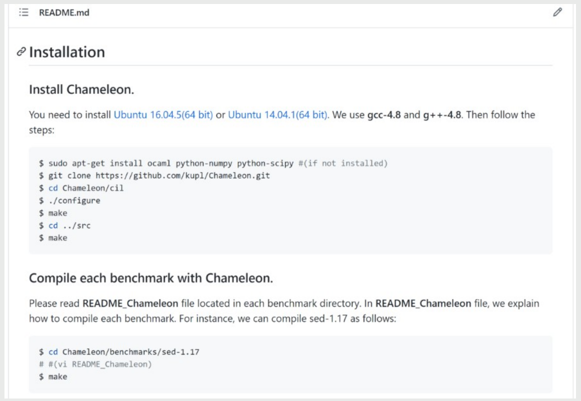
https://github.com/kupl/Chameleon/
(Source code)



Sorry, Ubuntu 15.04 has experienced an internal error.
If you notice further problems, try restarting the computer.
Send an error report to help fix this problem

program bugs

[Software]



README.md

Installation

Install Chameleon.

You need to install Ubuntu 16.04.5(64 bit) or Ubuntu 14.04.1(64 bit). We use **gcc-4.8** and **g++-4.8**. Then follow the steps:

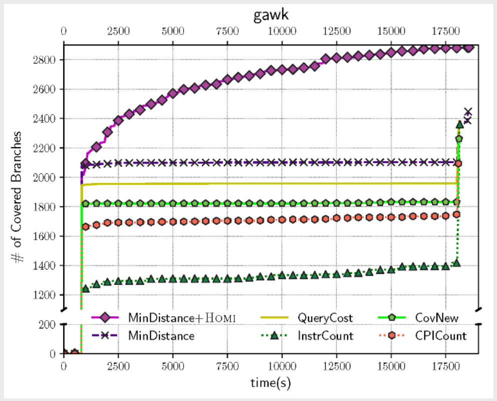
```
$ sudo apt-get install ocaml python-numpy python-scipy #(if not installed)
$ git clone https://github.com/kupl/Chameleon.git
$ cd Chameleon/cil
$ ./configure
$ make
$ cd ../src
$ make
```

Compile each benchmark with Chameleon.

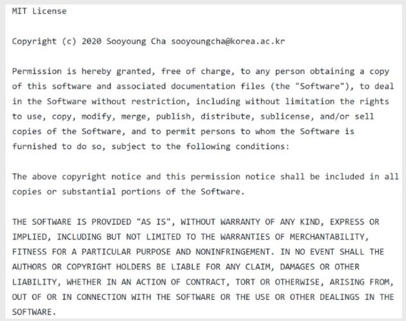
Please read **README_Chameleon** file located in each benchmark directory. In **README_Chameleon** file, we explain how to compile each benchmark. For instance, we can compile sed-1.17 as follows:

```
$ cd Chameleon/benchmarks/sed-1.17
# #(vi README_Chameleon)
$ make
```

"User manual"

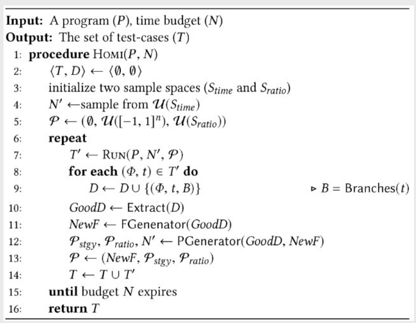


"Testing result"



```
MIT License

Copyright (c) 2020 Sooyoung Cha sooyoungcha@korea.ac.kr

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

"Licence"



```
Input: A program ($P$), time budget ($N$)
Output: The set of test-cases ($T$)
1: procedure HOMI($P, N$)
2:     $\langle T, D \rangle \leftarrow \langle \emptyset, \emptyset \rangle$
3:     initialize two sample spaces ($S_{time}$ and $S_{ratio}$)
4:     $N' \leftarrow$ sample from $\mathcal{U}(S_{time})$
5:     $\mathcal{P} \leftarrow (\emptyset, \mathcal{U}([-1, 1]^n), \mathcal{U}(S_{ratio}))$
6:     repeat
7:         $T' \leftarrow$ RUN($P, N', \mathcal{P}$)
8:         for each $(\Phi, t) \in T'$ do
9:             $D \leftarrow D \cup \{(\Phi, t, B)\}$                ▷ $B$ = Branches($t$)
10:        $GoodD \leftarrow$ Extract($D$)
11:        $NewF \leftarrow$ FGenenator($GoodD$)
12:        $\mathcal{P}_{stgy}, \mathcal{P}_{ratio}, N' \leftarrow$ PGenerator($GoodD, NewF$)
13:        $\mathcal{P} \leftarrow (NewF, \mathcal{P}_{stgy}, \mathcal{P}_{ratio})$
14:        $T \leftarrow T \cup T'$
15:    until budget $N$ expires
16:    return $T$
```

"Algorithm"

...

What is Software?

[Software In Academia]

ICSE (International Conference on Software Engineering)

- **The premier software engineering conference in the world.**

1. **A README file:** A main file describing what the software does.
2. **A REQUIREMENTS file:** This file should cover aspects of hardware environment requirements (e.g., storage) and software environments (e.g., Docker, OS).
3. **A LICENSE file:** The file describing the distribution rights.
4. **An INSTALL file:** These installation instructions should include notes illustrating a very basic usage example or a method to test the installation.

https://conf.researchr.org/track/icse-2022/icse-2022-artifact-evaluation

What is Software?

[Software In Academia]

ICSE (International Conference on Software Engineering)

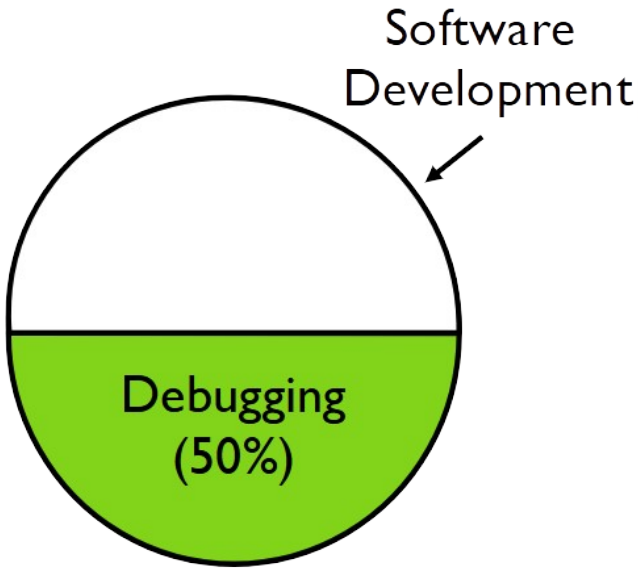
- **The premier software engineering conference in the world.**

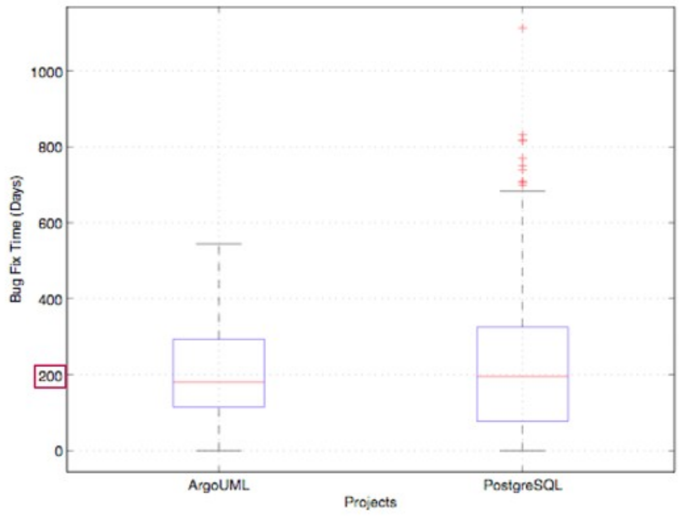
Artifacts Evaluated		Artifacts Available	Results Validated	
Functional	Reusable	Available	Reproduced	Replicated
acm	acm	acm	acm	acm

https://conf.researchr.org/track/icse-2022/icse-2022-artifact-evaluation

Software Costs

- **Software costs more to maintain than it does to develop.**
  - For systems with a long life, maintenance costs may be several times development costs.
  - Debugging takes up half of the time in SW development.
    - Bug-fix time: 200 days[1] (for commercial software)





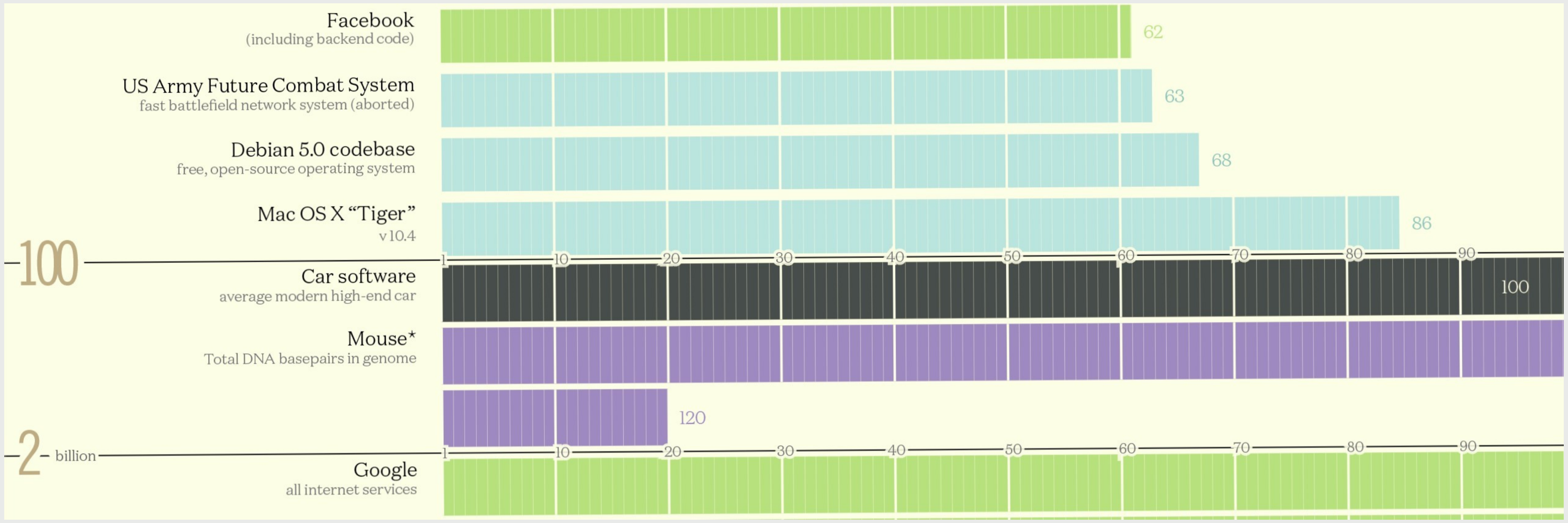
1. How Long Did It Take To Fix Bugs?

Chapter1-1. Software

SW Project Failure Reason (1)

Increasing system complexity

- The size of software is increasing rapidly.
  - Facebook (62M) → Car Software (100M) → Google(2B)
  - Software Complexity ↔ Software Size (exponential)



https://www.informationisbeautiful.net/visualizations/million-lines-of-code/

SW Project Failure Reason (2)

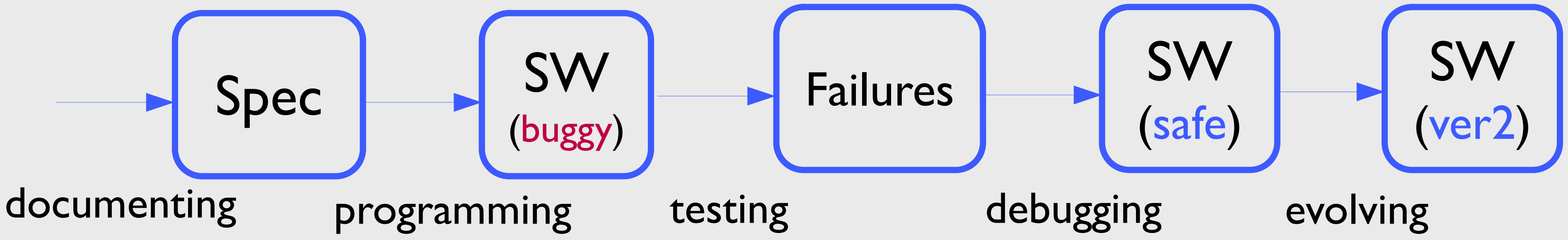
Failure to use software engineering methods

- It is fairly easy for people (including me) to write computer programs **without using software engineering methods and techniques.**
- They do not use software engineering methods in their everyday work.
- Consequently, their **software is often more expensive** and **less reliable than it should be**.

Software Engineering

[Software engineering]

- An **engineering discipline** that is concerned with from the early stages of system specification **all aspects of software production** through to maintaining the system after it has gone into use.



Software Engineering

[Software engineering]

- An **engineering discipline** that is concerned with from the early stages of system specification **all aspects of software production** through to maintaining the system after it has gone into use.

[Engineering discipline]

- Using appropriate **theories** and **methods** to solve problems bearing in mind organizational and financial constraints.

[All aspects of software production]

- Not just **technical process** of development.
- Also, **project management** and the **development of tools**, **methods** etc. to support software production.

Software Engineering

[Topics in Software engineering]

- **AI and software engineering**
  - Machine learning with and for SE, Program synthesis, **Program repair**
- **Testing and analysis**
  - **Software testing**, Program analysis, **Debugging** and Fault localization
- **Software evolution**
  - Evolution and maintenance, Software reuse, Refactoring
- **Social aspects of software engineering**
  - **Agile methods** and **software processes**, **Ethics** in software engineering
- **Requirements, modeling, and design**
  - **Requirements engineering**, Modeling and **model-driven engineering**

Chapter1-2. SW engineering

Importance of SW engineering

Dependency of Software

- **More and more**, individuals and society **rely on complex software systems**.
- We need to be able to produce reliable and trustworthy systems economically and quickly.

Cost Saving

- **It is usually cheaper, in the long run,** to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project.
- For most types of system, **most costs** are the costs of **changing the software** after it has gone into use.

Software Process Activities

Software Specification

where customers and engineers **define the software** that is to be produced and the **constraints** on its operation.

Software Development

where the software is **designed** and **programmed**.

Software Validation

where the software is checked to **ensure** that it is what the customer **requires**.

Software Evolution

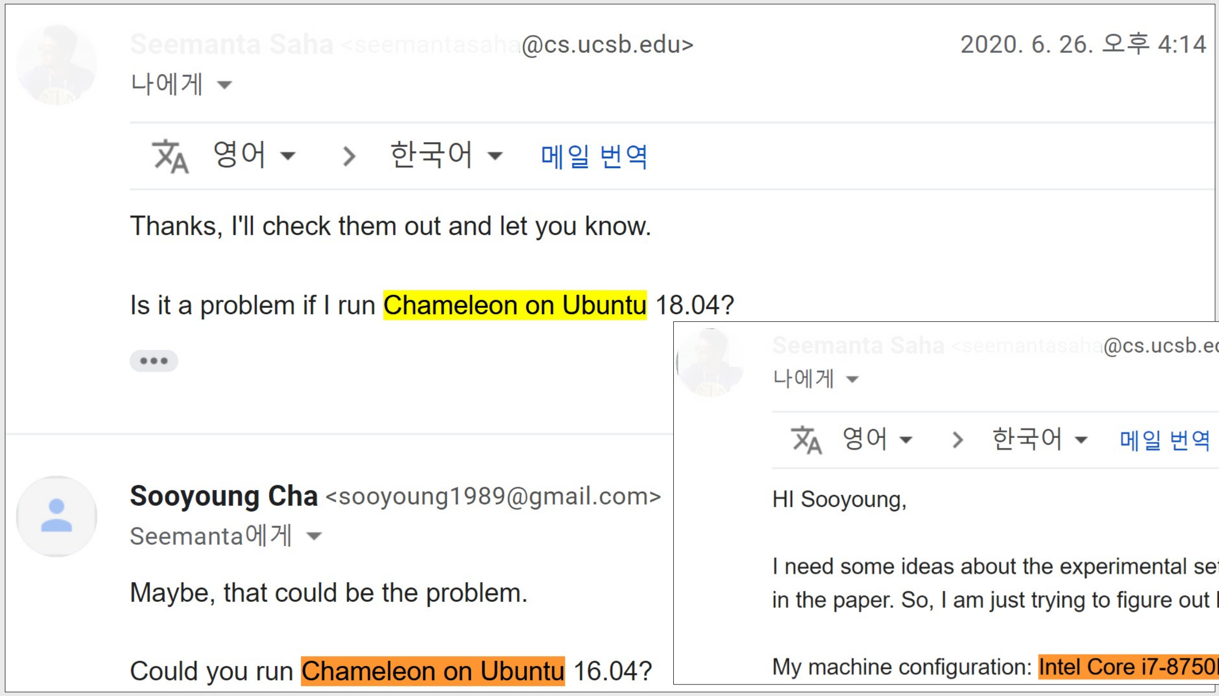
where the software is **modified** to **reflect changing** customer and market requirements.

Chapter1-3. About Developing SW

General Issues that Affect SW (1)

Heterogeneity

- Increasingly, systems are required to operate as **distributed systems** across networks that include **different types of computer and mobile devices.**



Seemanta Saha <seemantasaha@cs.ucsb.edu> 2020. 6. 26. 오후 4:14
나에게

文A 영어 > 한국어 메일 번역

Thanks, I'll check them out and let you know.

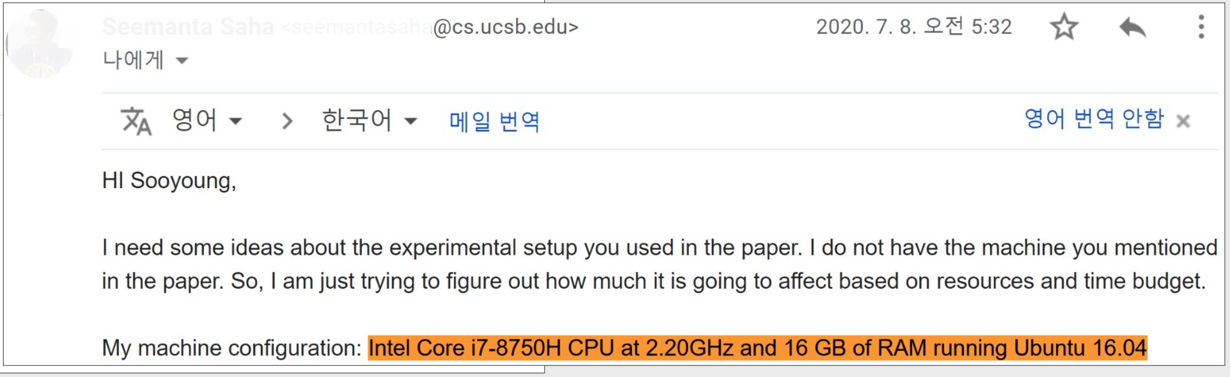
Is it a problem if I run Chameleon on Ubuntu 18.04?

...

Sooyoung Cha <sooyoung1989@gmail.com>
Seemanta에게

Maybe, that could be the problem.

Could you run Chameleon on Ubuntu 16.04?



Seemanta Saha <seemantasaha@cs.ucsb.edu> 2020. 7. 8. 오전 5:32
나에게

文A 영어 > 한국어 메일 번역 영어 번역 안함

HI Sooyoung,

I need some ideas about the experimental setup you used in the paper. I do not have the machine you mentioned in the paper. So, I am just trying to figure out how much it is going to affect based on resources and time budget.

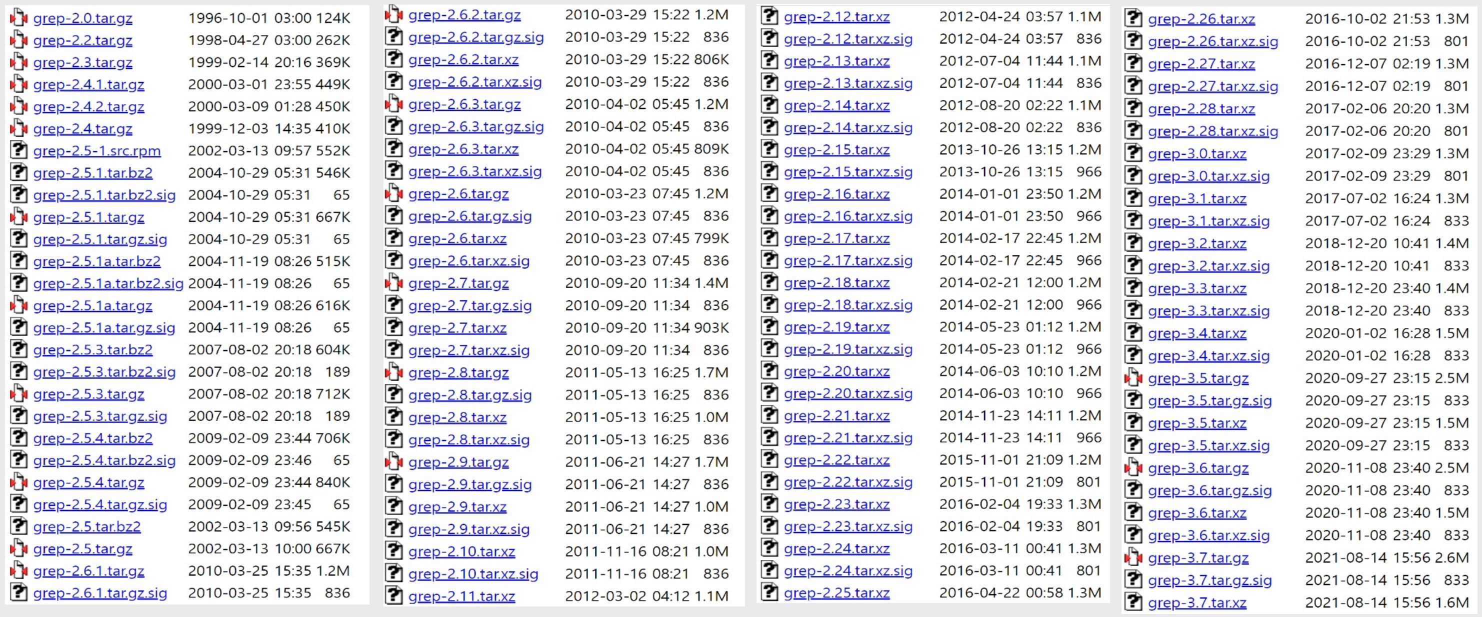
My machine configuration: Intel Core i7-8750H CPU at 2.20GHz and 16 GB of RAM running Ubuntu 16.04

Chapter1-3. About Developing SW

General Issues that Affect SW (1)

Business and social change

- Business and society are **changing incredibly quickly** as emerging economies develop and new technologies become available.
- They need to be able to **change** their existing **software** and **to rapidly develop** new software.
  - ex) grep-2.0(1996.10) ~ grep-3.7(2021.08), 42 version updates)



```
grep-2.0.tar.gz            1996-10-01 03:00 124K
grep-2.2.tar.gz            1998-04-27 03:00 262K
grep-2.3.tar.gz            1999-02-14 20:16 369K
grep-2.4.1.tar.gz          2000-03-01 23:55 449K
grep-2.4.2.tar.gz          2000-03-09 01:28 450K
grep-2.4.tar.gz            1999-12-03 14:35 410K
grep-2.5-1.src.rpm         2002-03-13 09:57 552K
grep-2.5.1.tar.bz2         2004-10-29 05:31 546K
grep-2.5.1.tar.bz2.sig     2004-10-29 05:31   65
grep-2.5.1.tar.gz          2004-10-29 05:31 667K
grep-2.5.1.tar.gz.sig      2004-10-29 05:31   65
grep-2.5.1a.tar.bz2        2004-11-19 08:26 515K
grep-2.5.1a.tar.bz2.sig    2004-11-19 08:26   65
grep-2.5.1a.tar.gz         2004-11-19 08:26 616K
grep-2.5.1a.tar.gz.sig     2004-11-19 08:26   65
grep-2.5.3.tar.bz2         2007-08-02 20:18 604K
grep-2.5.3.tar.bz2.sig     2007-08-02 20:18  189
grep-2.5.3.tar.gz          2007-08-02 20:18 712K
grep-2.5.3.tar.gz.sig      2007-08-02 20:18  189
grep-2.5.4.tar.bz2         2009-02-09 23:44 706K
grep-2.5.4.tar.bz2.sig     2009-02-09 23:46   65
grep-2.5.4.tar.gz          2009-02-09 23:44 840K
grep-2.5.4.tar.gz.sig      2009-02-09 23:45   65
grep-2.5.tar.bz2           2002-03-13 09:56 545K
grep-2.5.tar.gz            2002-03-13 10:00 667K
grep-2.6.1.tar.gz          2010-03-25 15:35 1.2M
grep-2.6.1.tar.gz.sig      2010-03-25 15:35  836
```

```
grep-2.6.2.tar.gz          2010-03-29 15:22 1.2M
grep-2.6.2.tar.gz.sig      2010-03-29 15:22  836
grep-2.6.2.tar.xz          2010-03-29 15:22 806K
grep-2.6.2.tar.xz.sig      2010-03-29 15:22  836
grep-2.6.3.tar.gz          2010-04-02 05:45 1.2M
grep-2.6.3.tar.gz.sig      2010-04-02 05:45  836
grep-2.6.3.tar.xz          2010-04-02 05:45 809K
grep-2.6.3.tar.xz.sig      2010-04-02 05:45  836
grep-2.6.tar.gz            2010-03-23 07:45 1.2M
grep-2.6.tar.gz.sig        2010-03-23 07:45  836
grep-2.6.tar.xz            2010-03-23 07:45 799K
grep-2.6.tar.xz.sig        2010-03-23 07:45  836
grep-2.7.tar.gz            2010-09-20 11:34 1.4M
grep-2.7.tar.gz.sig        2010-09-20 11:34  836
grep-2.7.tar.xz            2010-09-20 11:34 903K
grep-2.7.tar.xz.sig        2010-09-20 11:34  836
grep-2.8.tar.gz            2011-05-13 16:25 1.7M
grep-2.8.tar.gz.sig        2011-05-13 16:25  836
grep-2.8.tar.xz            2011-05-13 16:25 1.0M
grep-2.8.tar.xz.sig        2011-05-13 16:25  836
grep-2.9.tar.gz            2011-06-21 14:27 1.7M
grep-2.9.tar.gz.sig        2011-06-21 14:27  836
grep-2.9.tar.xz            2011-06-21 14:27 1.0M
grep-2.9.tar.xz.sig        2011-06-21 14:27  836
grep-2.10.tar.xz           2011-11-16 08:21 1.0M
grep-2.10.tar.xz.sig       2011-11-16 08:21  836
grep-2.11.tar.xz           2012-03-02 04:12 1.1M
```

```
grep-2.12.tar.xz           2012-04-24 03:57 1.1M
grep-2.12.tar.xz.sig       2012-04-24 03:57  836
grep-2.13.tar.xz           2012-07-04 11:44 1.1M
grep-2.13.tar.xz.sig       2012-07-04 11:44  836
grep-2.14.tar.xz           2012-08-20 02:22 1.1M
grep-2.14.tar.xz.sig       2012-08-20 02:22  836
grep-2.15.tar.xz           2013-10-26 13:15 1.2M
grep-2.15.tar.xz.sig       2013-10-26 13:15  966
grep-2.16.tar.xz           2014-01-01 23:50 1.2M
grep-2.16.tar.xz.sig       2014-01-01 23:50  966
grep-2.17.tar.xz           2014-02-17 22:45 1.2M
grep-2.17.tar.xz.sig       2014-02-17 22:45  966
grep-2.18.tar.xz           2014-02-21 12:00 1.2M
grep-2.18.tar.xz.sig       2014-02-21 12:00  966
grep-2.19.tar.xz           2014-05-23 01:12 1.2M
grep-2.19.tar.xz.sig       2014-05-23 01:12  966
grep-2.20.tar.xz           2014-06-03 10:10 1.2M
grep-2.20.tar.xz.sig       2014-06-03 10:10  966
grep-2.21.tar.xz           2014-11-23 14:11 1.2M
grep-2.21.tar.xz.sig       2014-11-23 14:11  966
grep-2.22.tar.xz           2015-11-01 21:09 1.2M
grep-2.22.tar.xz.sig       2015-11-01 21:09  801
grep-2.23.tar.xz           2016-02-04 19:33 1.3M
grep-2.23.tar.xz.sig       2016-02-04 19:33  801
grep-2.24.tar.xz           2016-03-11 00:41 1.3M
grep-2.24.tar.xz.sig       2016-03-11 00:41  801
grep-2.25.tar.xz           2016-04-22 00:58 1.3M
```

```
grep-2.26.tar.xz           2016-10-02 21:53 1.3M
grep-2.26.tar.xz.sig       2016-10-02 21:53  801
grep-2.27.tar.xz           2016-12-07 02:19 1.3M
grep-2.27.tar.xz.sig       2016-12-07 02:19  801
grep-2.28.tar.xz           2017-02-06 20:20 1.3M
grep-2.28.tar.xz.sig       2017-02-06 20:20  801
grep-3.0.tar.xz            2017-02-09 23:29 1.3M
grep-3.0.tar.xz.sig        2017-02-09 23:29  801
grep-3.1.tar.xz            2017-07-02 16:24 1.3M
grep-3.1.tar.xz.sig        2017-07-02 16:24  833
grep-3.2.tar.xz            2018-12-20 10:41 1.4M
grep-3.2.tar.xz.sig        2018-12-20 10:41  833
grep-3.3.tar.xz            2018-12-20 23:40 1.4M
grep-3.3.tar.xz.sig        2018-12-20 23:40  833
grep-3.4.tar.xz            2020-01-02 16:28 1.5M
grep-3.4.tar.xz.sig        2020-01-02 16:28  833
grep-3.5.tar.gz            2020-09-27 23:15 2.5M
grep-3.5.tar.gz.sig        2020-09-27 23:15  833
grep-3.5.tar.xz            2020-09-27 23:15 1.5M
grep-3.5.tar.xz.sig        2020-09-27 23:15  833
grep-3.6.tar.gz            2020-11-08 23:40 2.5M
grep-3.6.tar.gz.sig        2020-11-08 23:40  833
grep-3.6.tar.xz            2020-11-08 23:40 1.5M
grep-3.6.tar.xz.sig        2020-11-08 23:40  833
grep-3.7.tar.gz            2021-08-14 15:56 2.6M
grep-3.7.tar.gz.sig        2021-08-14 15:56  833
grep-3.7.tar.xz            2021-08-14 15:56 1.6M
```

Chapter1-3. About Developing SW

General issues that affect SW (2)

Security and trust

- As software is intertwined with all aspects of our lives, it is essential that we can **trust** that software.
  - ex) Therac-25: The worst software bugs in history
    - Many patients were killed by the radiation therapy machine.
    - The machine caused radiation overdoses because of software bug.

KILLED BY A MACHINE: THE THERAC-25

by: Adam Fabio

145 Comments

October 26, 2015

ttps://hackaday.com/2015/10/26/killed-by-a-machine-the-therac-25/

Question to understand SE (1)

[What is software?]

- **Computer programs and associated documentation.**

[What are the attributes of good software?]

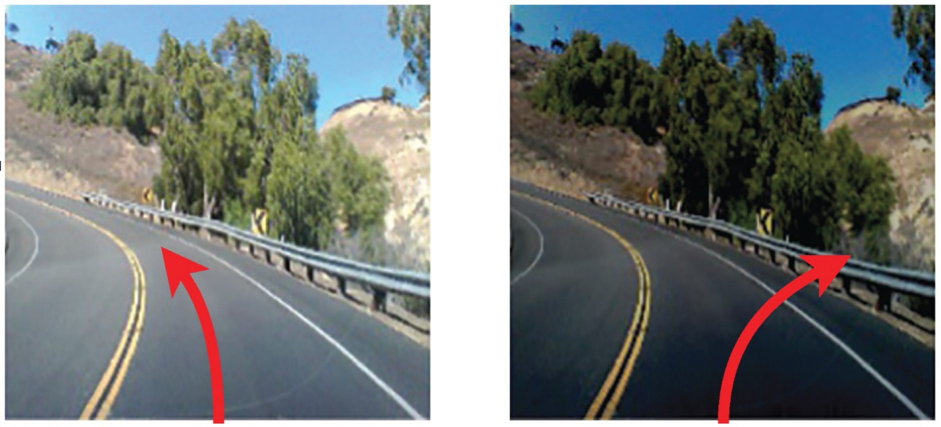
- Good software should deliver the **required functionality** and **performance** to the user and should be **maintainable**, **dependable** and **usable**.

[What is software engineering?]

- Software engineering is an **engineering discipline** that is concerned with all aspects of software production.

Chapter1-4. Details of SW Engineering

Question to understand



(a) Input 1 (b) Input 2 (darker version of 1)

DeepXplore: Automated Whitebox Testing of Deep Learning Systems

[What is software?]

- Computer programs and associated documentation.

[What are the attributes of good software?]

- Good software should deliver the **required functionality** and **performance** to the user and should be **maintainable**, **dependable** and **usable**.

[What is software engineering?]

- Software engineering is an **engineering discipline** that is concerned with all aspects of software production.

Question to understand SE (2)

[What are the fundamental software engineering activities?]

- Software specification, software development, software validation and software evolution.

[What are the key challenges facing software engineering?]

- Coping with increasing diversity, demands for reduced delivery times and developing **trustworthy software**.

[What are the costs of software engineering?]

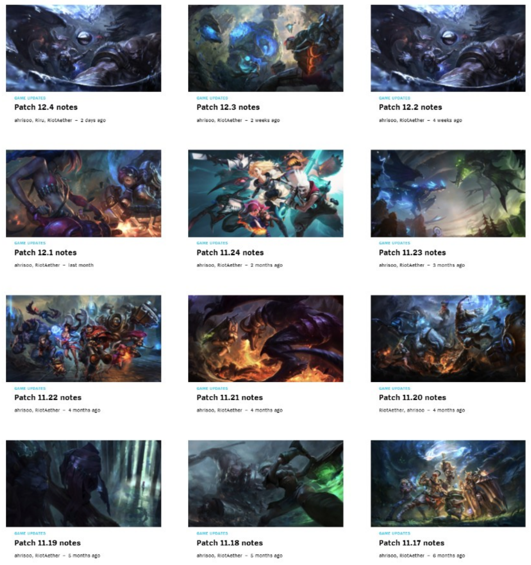
- Roughly **60%** of software costs are development costs, **40%** are testing costs.
- For custom software, evolution costs often exceed development costs.

Chapter1-4. Details of SW Engineering

Question to understand SE (3)

[What are the best software engineering techniques and methods?]

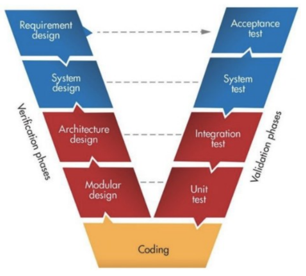
- Different techniques are appropriate for different types of system.
- Games (e.g., LoL) vs Safety critical systems (e.g., KHNP)



(A more flexible software process that accommodates rapid change)

VS

KHNP
KOREA HYDRO & NUCLEAR POWER CO., LTD



V model
(A complete specification to be developed)

Chapter1-4. Details of SW Engineering

Software products

Generic products

- **Stand-alone systems** that are marketed and sold to any customer who wishes to buy them.
- Examples – PC software such as word processors, drawing packages, and project management tools.

Customized products

- Software that is **commissioned by a specific customer** to meet their own needs.
- Examples – air traffic control software, traffic monitoring systems, embedded control systems.

Product specification

Generic products

- The specification of what the software should do **is owned by the software developer** and decisions on software change are made by the developer.

Customized products

- The specification of what the software should do **is owned by the customer** for the software and they make decisions on software changes that are required.

The distinction between these product types is becoming increasingly blurred.

Essential attributes of good Software (1)

Maintainability

- Software should be written in such a way so that it **can evolve** to meet the **changing needs** of customers.
- This is a critical attribute because **software change is an inevitable requirement** of a changing business environment.

Dependability and security

- Software **dependability** includes a range of characteristics including reliability, security and safety.
- Dependable software **should not cause** physical or economic **damage** in the event of **system failure**.
- **Malicious users** should **not be able to access** or damage the system.

Essential attributes of good Software (2)

Efficiency

- Software should **not make wasteful use of system resources** such as memory and processor cycles.
- **Efficiency** therefore includes **responsiveness, processing time, memory utilisation**, etc.

Acceptability

- Software must be **acceptable** to the type of **users** for which it is designed.
- This means that it must be **understandable, usable** and **compatible** with other systems that they use.

Software engineering ethics

- ✧ Software engineering involves **wider responsibilities** than simply the application of technical skills.

- ✧ Software engineers must **behave in an honest and ethically responsible way** if they are to be respected as professionals.

- ✧ Ethical behavior is **more than simply upholding the law** but involves following a set of principles that are morally correct.

Issues of professional responsibility (1)

Confidentiality

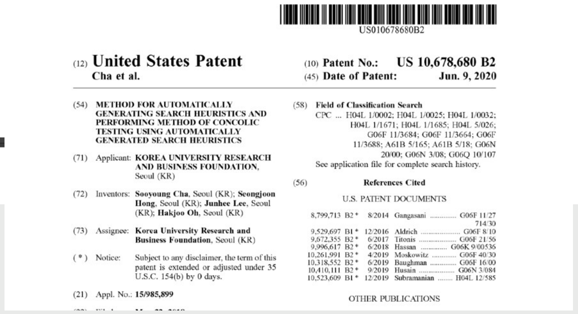
- Engineers should normally respect **the confidentiality of their employers** or clients irrespective of whether a formal confidentiality agreement has been signed.

Competence

- Engineers **should not misrepresent their level of competence**.
- They should not knowingly accept work which is outwith their competence.

Chapter 1-5. SW engineering ethics

Issues of professional responsibility



US010678680B2

(12) **United States Patent**
Cha et al.

(10) **Patent No.:** **US 10,678,680 B2**
(45) **Date of Patent:** **Jun. 9, 2020**

(54) **METHOD FOR AUTOMATICALLY GENERATING SEARCH HEURISTICS AND PERFORMING METHOD OF CONCOLIC TESTING USING AUTOMATICALLY GENERATED SEARCH HEURISTICS**

(71) Applicant: **KOREA UNIVERSITY RESEARCH AND BUSINESS FOUNDATION,** Seoul (KR)

(72) Inventors: **Sooyoung Cha**, Seoul (KR); **Seongjoon Hong**, Seoul (KR); **Junhee Lee**, Seoul (KR); **Hakjoo Oh**, Seoul (KR)

(73) Assignee: **Korea University Research and Business Foundation**, Seoul (KR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **15/985,899**

(58) **Field of Classification Search**
CPC ... H04L 1/0002; H04L 1/0025; H04L 1/0032; H04L 1/1671; H04L 1/1685; H04L 5/026; G06F 11/3684; G06F 11/3664; G06F 11/3688; A61B 5/165; A61B 5/18; G06N 20/00; G06N 3/08; G06Q 10/107
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,799,713	B2 *	8/2014	Gangasani	G06F 11/27 714/30
9,529,697	B1 *	12/2016	Aldrich	G06F 8/10
9,672,355	B2 *	6/2017	Titonis	G06F 21/56
9,996,617	B2 *	6/2018	Hassan	G06K 9/00536
10,261,991	B2 *	4/2019	Moskowitz	G06F 40/30
10,318,552	B2 *	6/2019	Baughman	G06F 16/00
10,410,111	B2 *	9/2019	Husain	G06N 3/084
10,523,609	B1 *	12/2019	Subramanian	H04L 12/585

OTHER PUBLICATIONS

Intellectual property rights

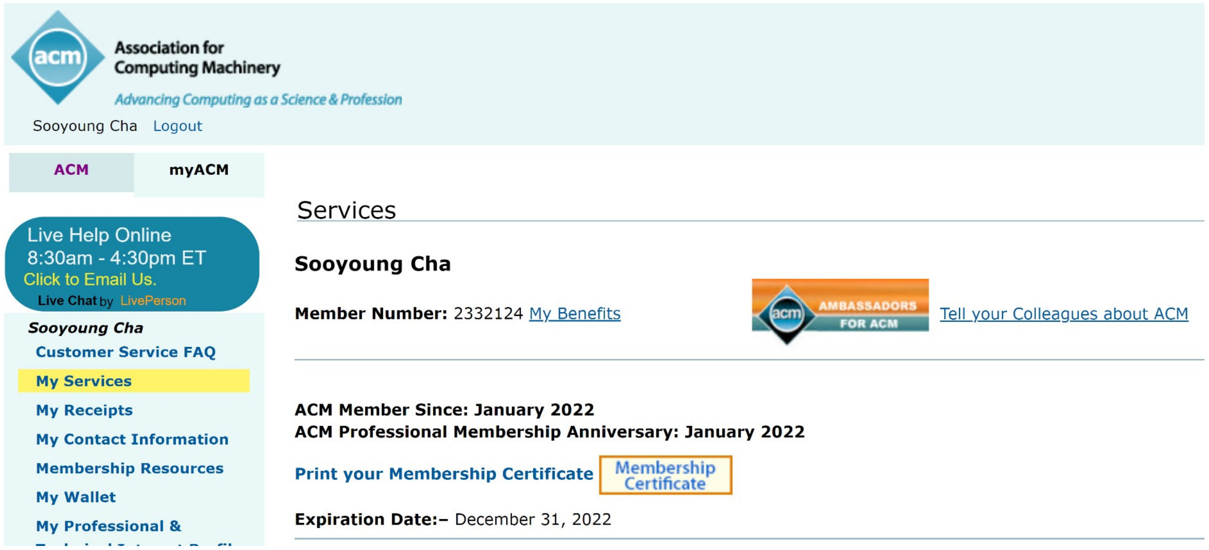
- Engineers **should be aware of local laws** governing the use of intellectual property such as patents, copyright, etc.
- They should be careful to ensure that the intellectual property of employers and clients is protected.

Computer misuse

- Software engineers should not use their technical skills to misuse other people's computers.
- Computer misuse ranges from relatively trivial (**game playing on an employer's machine, say**) to extremely serious (dissemination of viruses).

ACM / IEEE Code of Ethics

- Members of these organisations such as ACM, IEEE sign up to the code of practice when they join.
  - ACM: Association for Computing Machinery
  - IEEE: Institute of Electrical and Electronics Engineers
- Software engineers shall adhere to the eight principles.
  - ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices


Association for Computing Machinery
Advancing Computing as a Science & Profession
Sooyoung Cha Logout

ACM myACM

Live Help Online
8:30am - 4:30pm ET
Click to Email Us.
Live Chat by LivePerson

Sooyoung Cha
Customer Service FAQ
My Services
My Receipts
My Contact Information
Membership Resources
My Wallet
My Professional &

Services

Sooyoung Cha

Member Number: 2332124 My Benefits

AMBASSADORS FOR ACM Tell your Colleagues about ACM

ACM Member Since: January 2022
ACM Professional Membership Anniversary: January 2022

Print your Membership Certificate Membership Certificate

Expiration Date:– December 31, 2022

Ethical principles

1. PUBLIC - Software engineers shall act consistently with the public interest.

2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.

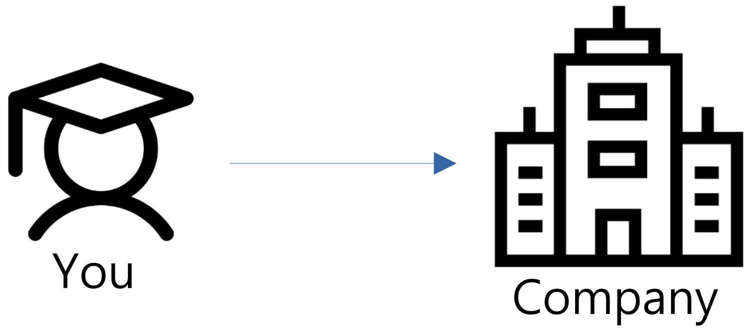
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Dilemma in ethical principles



("Developing a safety-critical system")

"because of time pressure, this company tries to falsify the safety validation records."

Respect the confidentiality of employers?	VS	Alert the customer or media?

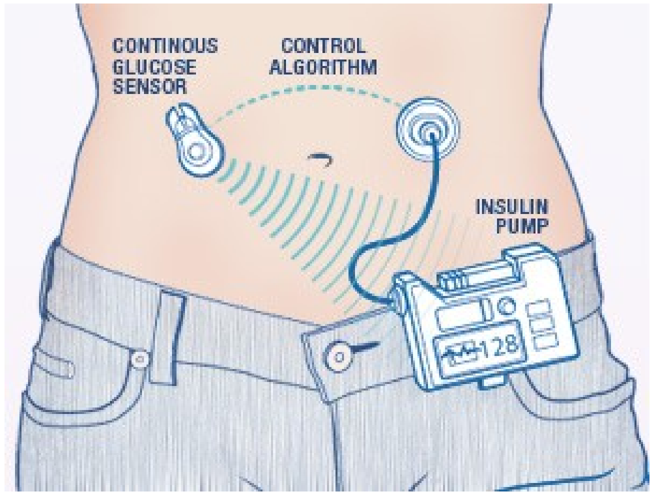
Case studies

- An insulin pump control system
  - An embedded system in an insulin pump used by diabetics to maintain blood glucose control.
- A mental health case patient management system
  - Mentcare. A system used to maintain records of people receiving care for mental health problems.
- A wilderness weather station
  - A data collection system that collects data about weather conditions in remote areas.
- iLearn: a digital learning environment
  - A system to support learning in schools

Insulin pump control system

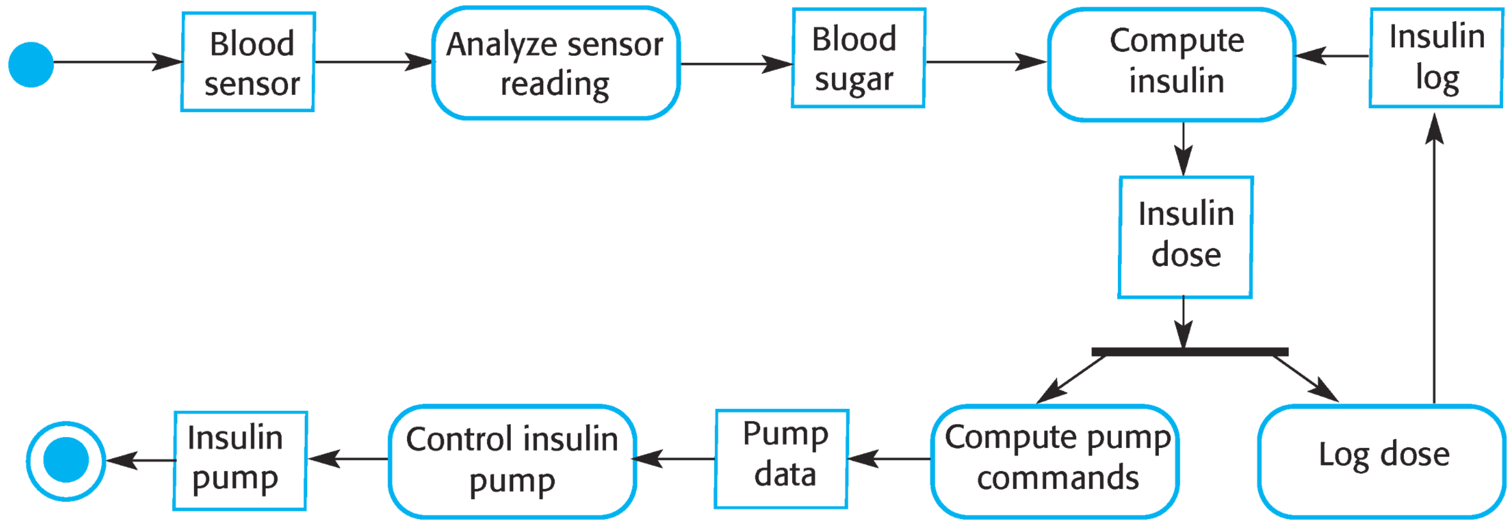
✧ Key roles

- Collect information from a blood sugar sensor.
- Calculate the amount of insulin required to be injected.
- Deliver a controlled dose of insulin to a user.



Insulin pump control system

- A UML activity model of the insulin pump system
  - UML: a Unified Modeling Language

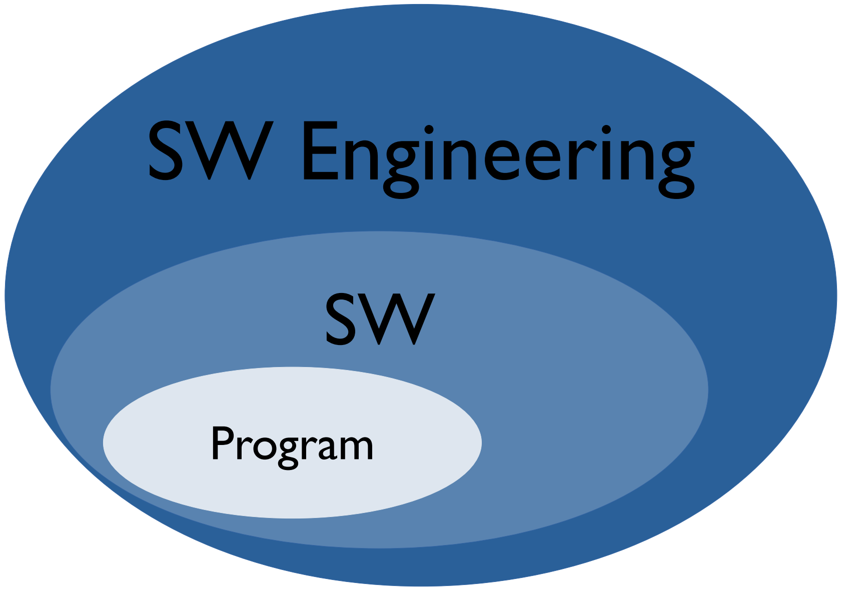


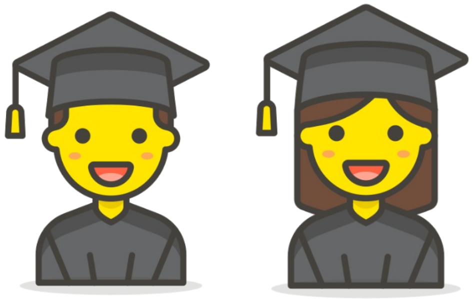
Insulin pump control system

✧ Essential high-level requirements

- The system shall be available **to deliver insulin only when required.**
- The system shall perform reliably and deliver **the correct amount of insulin** to counteract the current level of blood sugar.
- The system must therefore be designed and implemented to ensure that the system **always meets these requirements.**

Summary





Be an honest software engineer.

Thank You