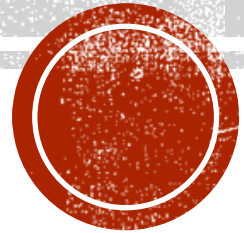
SOFTWARE ENGINEERING

Spring 2024



COURSE STRUCTURE

- Lectures 16 Weeks
- Assignment(3) 10%
- Mid Term 20%
- Final 45%
- Project(3 milestone) 15%
- Quizzes(3) 10%

- 1st project-related assignment: think about the ideas for the project during the shopping period

ACADEMIC INTEGRITY

- Expected to cooperate on projects

  … but not on Quizzes and Assignment!

- Default penalty: failing the Quiz or Assignment

COURSE MATERIAL

- All class material will be available on the Google Classroom

  For Section E-clcrydk

  For Section F- 2gorxj6

- Lecture notes, handouts, papers to read, homework, project announcements, etc.

Important: Check the GCR for every course announcements

COURSE MATERIAL

- Books to read:
  - Roger Pressman: "Software Engineering: A Practitioner's Approach", ISBN-10: 0073375977
  - Ian Sommerville: "Software Engineering", ISBN-10: 0137035152
  - Steve McConnell: "Code Complete: A Practical Handbook of Software Construction", ISBN-10: 0735619670
  - Frederick Brooks: ‘‘The Mythical Man-Month’’, ISBN 0-201-83595-9

THE PROJECT

- The only way to learn "software engineering" is by writing a large piece of code in a group
- A BIG project solving a real-world problem
  - Can be (almost) anything
- Done in teams of 3-4 students
  - You do everything
  - Gather requirements, design, code, and test in several assignments
  - This class should be very close to a startup experience

PROJECT TIMELINE

- Project nominations
  - Start thinking about the project proposal already today
  - Project nomination will be due in a week after the shopping period
- Project selection, team assignments
- Projects will be reviewed and then approved

THE IDEAS BEHIND THE PROJECT STRUCTURE

- We will simulate the "real world"
- In the real world, you often spend a lot of time maintaining/extending other people's code
  - This is where specifications, interfaces, documentation, etc. pays off
  - Shows the importance of institutional knowledge.

WHAT THIS COURSE IS (NOT) ABOUT?

- Do not expect to learn a new language
- Do not expect to learn programming tricks
  - But you'll learn techniques by yourself for "programming in the large"
- Do not expect to learn management skills from the lectures
  - Some things you learn by doing, not through lectures!

WHAT THIS COURSE IS ABOUT?

- Learn how to build a large software system in a team
  - Learn how to collect requirements
  - Learn how to write specification
  - Learn how to design
- Reliability is central to software engineering: This constitutes significant part of the course
  - Version Control
  - Testing
  - Debugging
  - Dynamic Analysis

WHAT IS SOFTWARE ENGINEERING?

- As defined in IEEE Standard 610.12:
  - *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*

SOFTWARE ENGINEERING MYTHS: MANAGEMENT

- "We have books with rules. Isn't that everything my people need?"
  - Which book do you think is perfect for you?
- "If we fall behind, we add more programmers"
  - "Adding people to a late software project, makes it later" – Fred Brooks (The Mythical Man Month)
- "We can outsource it"
  - If you do not know how to manage and control it internally, you will struggle to do this with outsiders

SOFTWARE ENGINEERING MYTHS: CUSTOMER

- "We can refine the requirements later"
  - A recipe for disaster.
- "The good thing about software is that we can change it later easily"
  - As time passes, cost of changes grows rapidly

SOFTWARE ENGINEERING MYTHS: PRACTITIONER

- "Let's write the code, so we'll be done faster"
  - "The sooner you begin writing code, the longer it'll take to finish"
  - 60-80% of effort is expended after first delivery
- "Until I finish it, I cannot assess its quality"
  - Software and design reviews are more effective than testing (find 5 times more bugs)
- "There is no time for software engineering"
  - But is there time to redo the software?

HOW DO WE KNOW THE SYSTEM WORKS?

- Buggy software is a huge problem
  - But you likely already know that
- Defects in software are commonplace
  - Much more common than in other engineering disciplines
- This is not inevitable---we can do better!

SOFTWARE BUGS – SPACE DISASTER

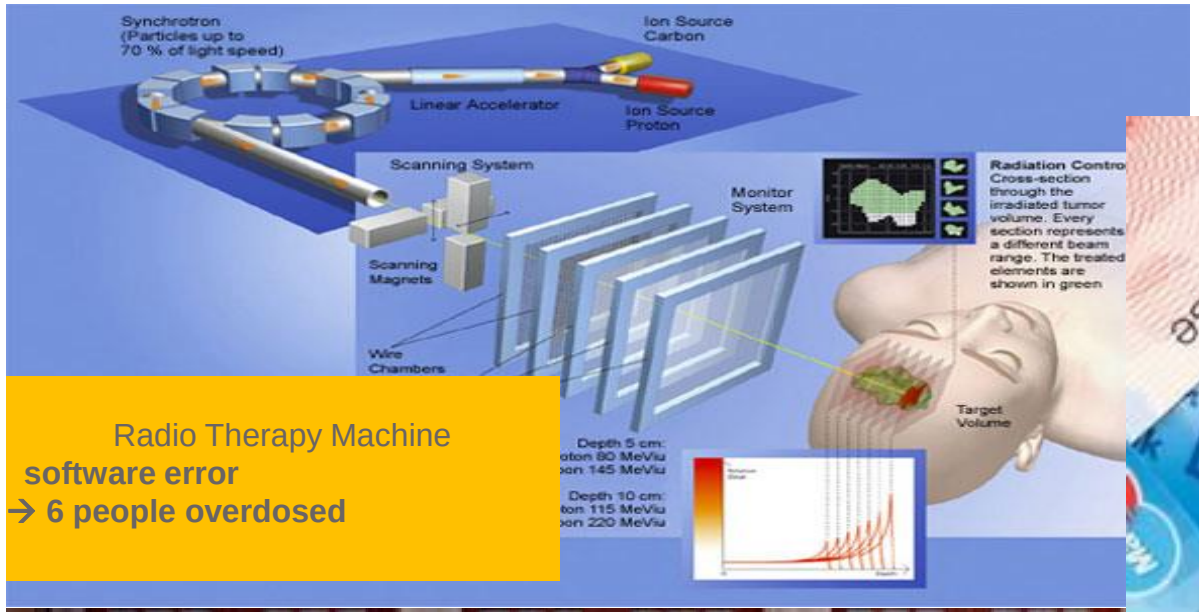
Maiden flight of the Ariane 5 rocket on the 4th of June 1996

- The reason for the explosion was a software error (Attempt to cram a 64-floating point number to a 16-bit integer failed)
- Financial loss: $500,000,000 (including indirect costs: $2,000,000,000)

SOFTWARE BUGS – SPACE DISASTER

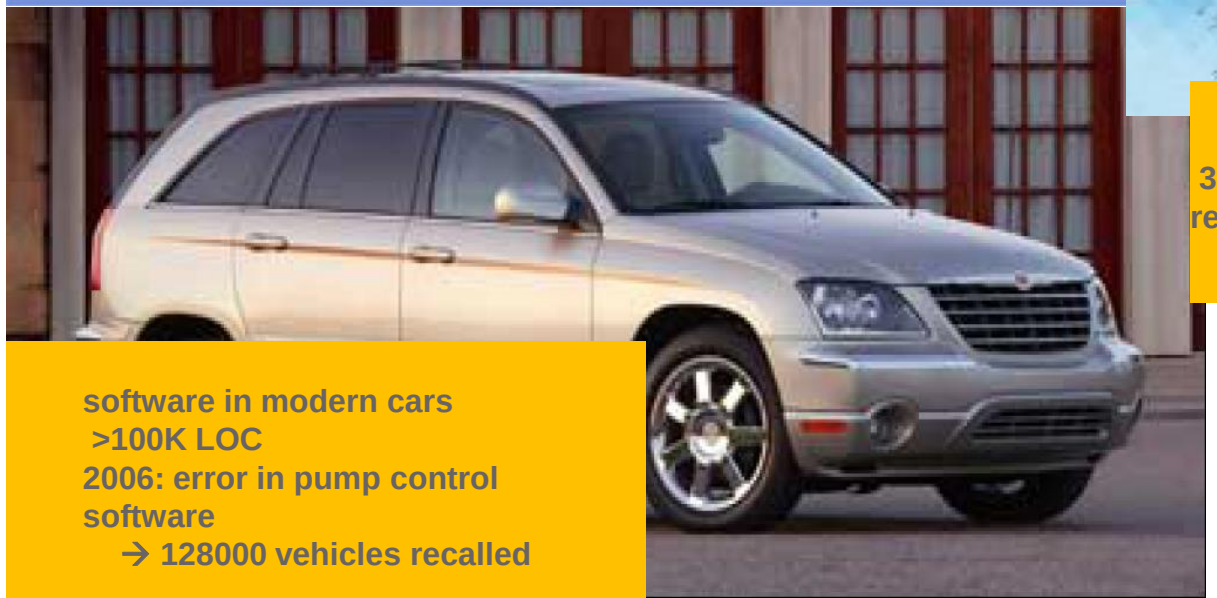
" The failure of Ariane 501 was caused by the complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system. The extensive reviews and tests carried out during the Ariane 5 development programmed did not include adequate analysis and testing of the inertial reference system or of the complete flight control system, which could have detected the potential failure."

EXAMPLES OF SOFTWARE ERRORS



Radio Therapy Machine
software error
→ 6 people overdosed

Year 2010 Bug
30 million debit and credit cards have been rendered unreadable by the software bug



software in modern cars
>100K LOC
2006: error in pump control software
→ 128000 vehicles recalled

[10 Seriously Epic Computer Software Bugs - Listverse]

FINANCIAL IMPACT OF SOFTWARE ERRORS

Recent research at Cambridge University (2013) showed that the global cost of software bugs is

around 312 billion of dollars annually

Goal: to increase software reliability

PROJECT FAILURE

Increasing System Complexity

As new software engineering techniques help us to build larger, more complex systems, the demands change. Systems have to be built and delivered more quickly; larger, even more complex systems are required; systems have to have new capabilities that were previously thought to be impossible.

Failure to use software engineering methods

It is fairly easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. Consequently, their software is often more expensive and less reliable than it should be.

SOFTWARE AND SOFTWARE ENGINEERING

Software

- Software refers to a collection of programs, data, and instructions that tell a computer how to perform specific tasks. It includes both the programs themselves and the documentation that describes how to use and maintain them.

Software Engineering

- Software engineering is the systematic application of engineering approaches to the development, operation, and maintenance of software. It involves the use of scientific and engineering principles, methods, and tools to create high-quality software products.

Difference

- While software refers to the actual programs and data, software engineering focuses on the process of designing, developing, and maintaining software. Software engineering involves the application of engineering principles and practices to ensure the quality, reliability, and efficiency of software products.

SOFTWARE PRODUCTS

ESSENTIAL ATTRIBUTES OF GOOD SOFTWARE

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.

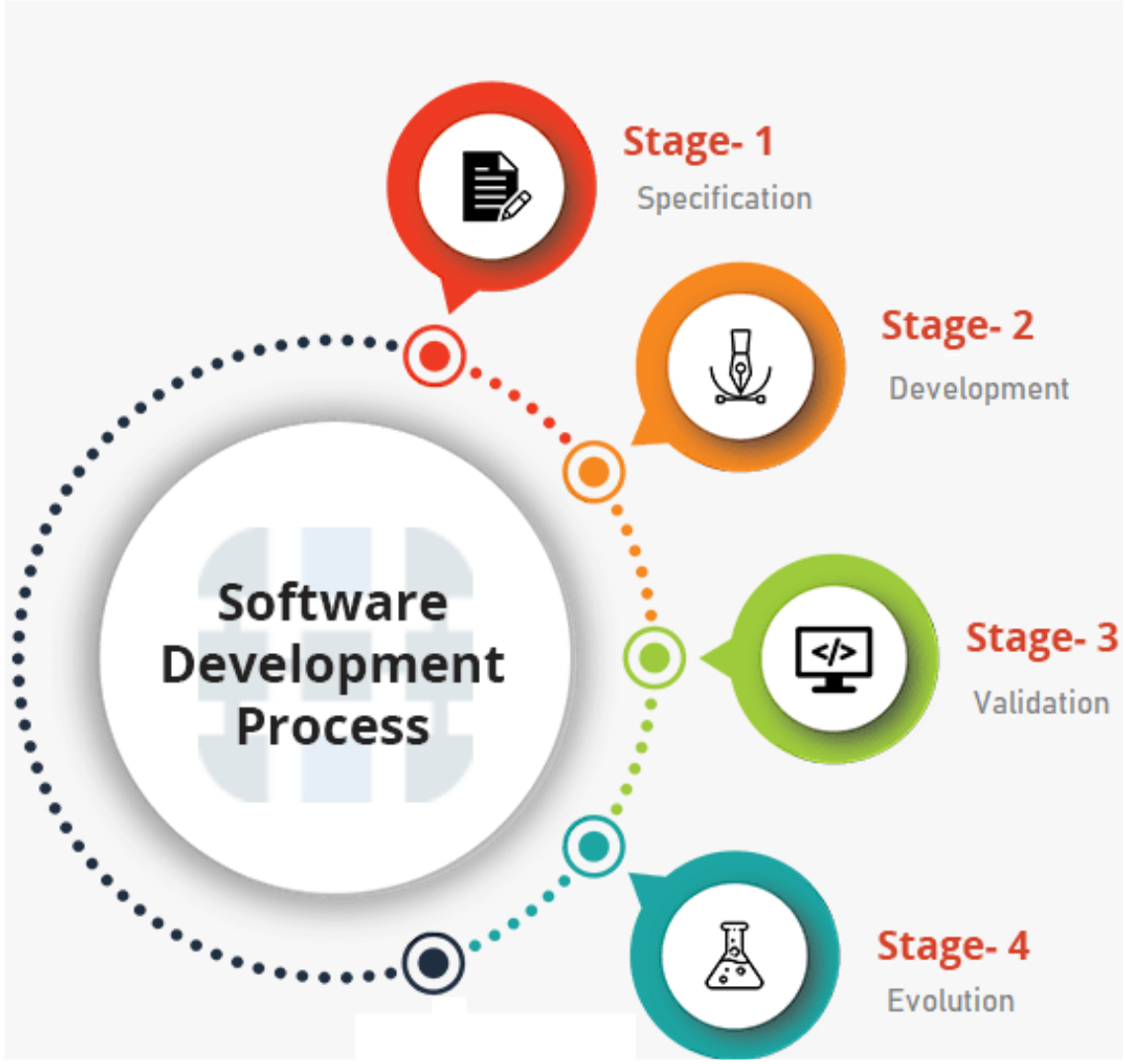
Efficiency

- 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 PROCESS ACTIVITIES



GENERAL ISSUES THAT EFFECT THE SOFTWARE

- **Heterogeneity**
- **Business and social change**
- **Security and trust**
- **Scale**

APPLICATION TYPES

- Stand-alone applications
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
- Interactive transaction-based applications
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.
- Embedded control systems
  - These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system.

APPLICATION TYPES

- Batch processing systems
  - These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.
- Entertainment systems
  - These are systems that are primarily for personal use and which are intended to entertain the user.
- Systems for modelling and simulation
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

APPLICATION TYPES

- Data collection systems
  - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- Systems of systems
  - These are systems that are composed of a number of other software systems.

Case Study: Mental Health care Patient Management System(MHCPMS)

This case study is based on a real system that is in use in a number of hospitals. For reasons of commercial confidentiality, I have changed the name of the system and have not included information about any specific system features.

Background A regional health authority wishes to procure an information system to help manage the care of patients suffering from mental health problems. The overall goals of the system are twofold:

1. To generate management information that allows health service managers to assess performance against local and government targets.

2. To provide medical staff with timely information to facilitate the treatment of patients. The health authority has a number of clinics that patients may attend in different hospitals and in local health centers.

Patients need not always attend the same clinic and some clinics may support 'drop in' as well as pre-arranged appointments. The nature of mental health problems is such that patients are often disorganized so may miss appointments, deliberately or accidentally lose prescriptions and medication, forget instructions and make unreasonable demands on medical staff. In a minority of cases, they may be a danger to themselves or to other people. They may regularly change address and may be homeless on a long-term or short-term basis. Where patients are dangerous, they may need to be 'sectioned' – confined to a secure hospital for treatment and observation. Users of the system include clinical staff (doctors, nurses, health visitors), receptionists who make appointments and medical records staff. Reports are generated for hospital management by medical records staff. Management has no direct access to the system. The system is affected by two pieces of legislation (in the UK, Acts of Parliament). These are the Data Protection Act that governs the confidentiality of personal information and the Mental Health

CONCERNS WITH RESPECT TO MHCPMS

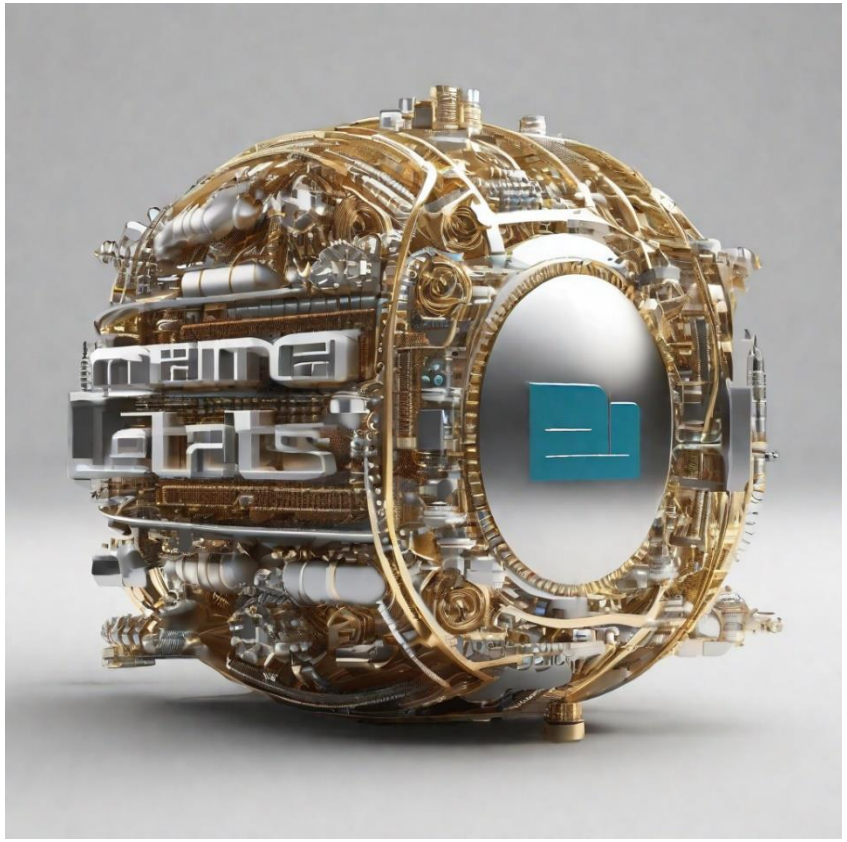
1. Usability. The system must be used by a range of staff, often working under time pressure and with different levels of experience using computer-based information systems.

2. Safety. Patients may cause harm to themselves or others. The provisions of the Mental Health Act must be taken into account.

3. Privacy. Patient privacy must be maintained according to the Data Protection Act and local ethical guidelines.

SOFTWARE ENGINEERING ETHICS

- **Ethical Practices in Software Engineering**
- Ethics play a crucial role in software engineering, ensuring the responsible and ethical development of software systems.
- **Ethical Considerations**
- Software engineers must consider the potential impact of their work on individuals, society, and the environment.
- **Responsibilities of Software Engineers**
- Software engineers have a responsibility to act ethically and uphold professional standards to ensure the safety, privacy, and well-being of users and stakeholders.



PROFESSIONALISM AND PROFESSIONAL ORGANIZATIONS

Professionalism in Software Engineering

- Professionalism in software engineering refers to the conduct, skills, and ethical standards expected of professionals in the industry. It involves a commitment to continuous learning, adherence to professional ethics, and the ability to work collaboratively in teams.

Role of Professional Organizations

- Professional organizations play a crucial role in the software engineering industry. They provide a platform for networking, knowledge sharing, and professional development. These organizations offer various benefits and resources to their members, including access to industry events, training programs, research publications, and career advancement opportunities.

SOFTWARE ENGINEERING CODE OF ETHICS

Importance of the Code of Ethics

- The software engineering code of ethics plays a crucial role in guiding ethical behavior within the field. It provides a set of principles and guidelines that software engineers should adhere to in order to ensure the development of ethical and responsible software.

Principles and Guidelines

- The code of ethics outlines several key principles and guidelines that software engineers should follow. These include:
- Ensuring the safety and well-being of users and the public.
- Maintaining confidentiality and protecting user data.
- Being honest and transparent in all professional interactions.
- Continuing professional development and staying up-to-date with the latest technologies and best practices.

ACM/IEEE CODE OF ETHICS

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

THE ACM/IEEE CODE OF ETHICS

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

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.

ACM ETHICS

- Diane recently started a new industry research job, joining the interactive technologies team. In graduate school, her advisor had collaborated with several members of the team on a number of research projects, involving and highlighting Diane's contributions whenever possible. The team had been impressed by Diane's work and recruited her as she was approaching graduation.
- Max, the team's technical leader had built a reputation as a brilliant yet mercurial expert in augmented reality. His team's contributions were highly cited within the field, with Max typically claiming primary authorship as the team leader. Their work was also highlighted frequently in popular press, always with quotes only from Max. Despite the team's repeated successes, Max would erupt with verbal and personal attacks for even minor mistakes. He would yell at the person and berate them in internal chat forums. On multiple occasions, women team members have found their names removed from journal manuscript submissions as punishment.
- Diane soon found herself the target of one of Max's tirades when she committed a code update that introduced a timing glitch in the prototype shortly before a live demo. Infuriated, Max refused to allow Diane to join the team on stage. Feeling Max's reaction was unprofessional and abusive, Diane approached the team's manager, Jean. Jean agreed that the experience was unpleasant, but that was the price to pay for working in an intense, industry-leading team. Jean's advice to Diane was to "Grow up and get over it."