# CS445 / ECE451 / CS645 / SE463

## Software Requirements Specification & Analysis

# Introduction

# Whole Course on Requirements?

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."

Fred Brooks, "No Silver Bullet – Essence and Accident in Software Engineering", *IEEE Computer*, 1987

# Successes vs. Flops

## Airbnb

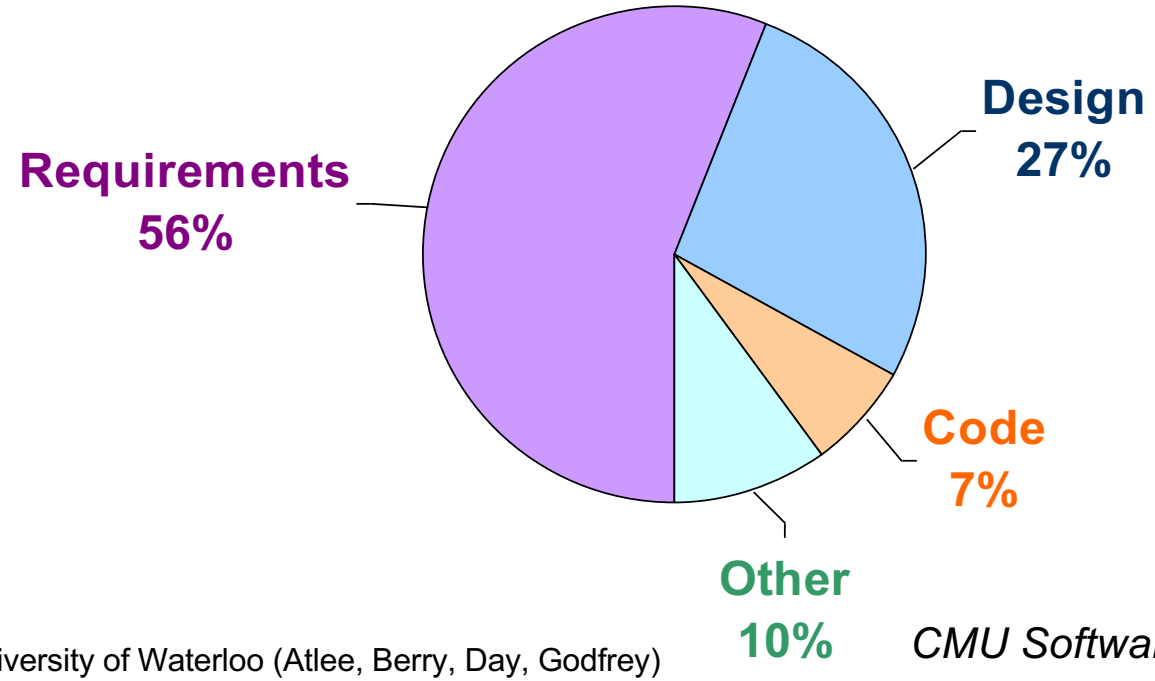
Basic short-term room bookings
payments, booking fees
Craigslist platform hack
Airbnb photography program
Airbnb Social Connections
Wish Lists
$1,000,000 Host Guarantee
Airbnb Neighbourhoods
Localmind
Host Home
Hospitality Standards
Host Groups
Discover Feed
Price Tips

## Facebook

Facebook Notify
Facebook Parse
Facebook Home
Facebook Deals
Facebook Gifts
Facebook Offers
Facebook Credits
Autofill (credit card info) with Facebook
Facebook Inbox
Facebook FBML
Facebook Lite
Facebook Poke
Facebook Slingshot
Facebook Questions
Facebook Places
Frictionless Sharing
Facebook Beacon
Facebook Sponsored Stories
Facebook Phone

# Failures in Software Products

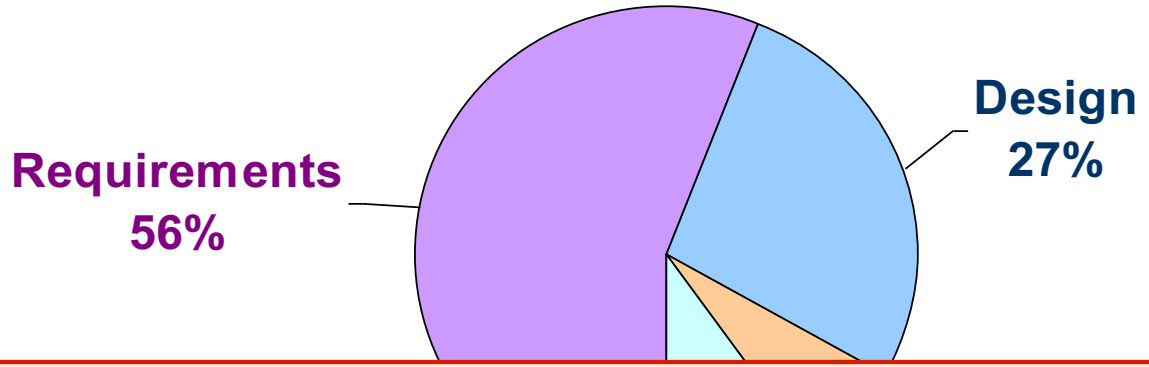
Most software systems fail because they don't meet the needs of their users (rather than because they aren't implemented properly).




*CMU Software Engineering Institute, 2005*

# Failures in Software Products

Most software systems fail because they don't meet the needs of their users (rather than because they aren't implemented properly).

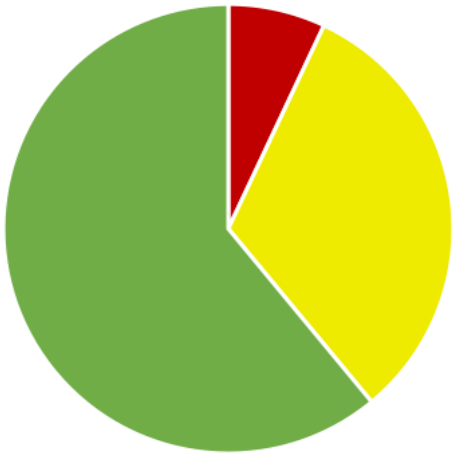



Software developers often don't know what the software is **supposed** to do.

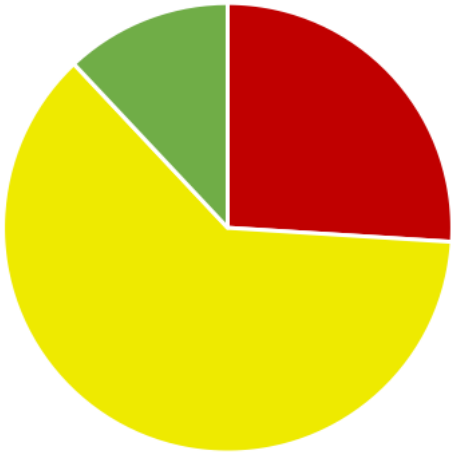
*CMU Software Engineering Institute, 2005*

# Standish Group Report - CHAOS

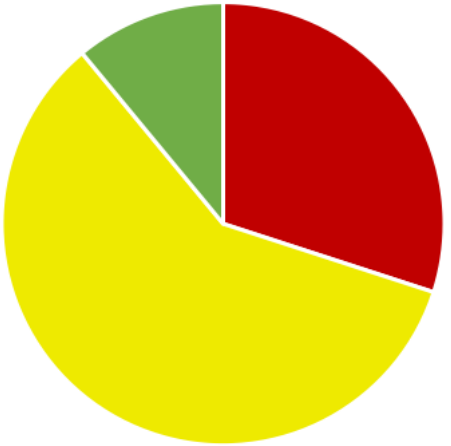
Annual survey of IT executives shows that most software projects are not successful.



Small Projects



Medium Projects



Large Projects

Success – satisfactory functionality, on-time, on-budget

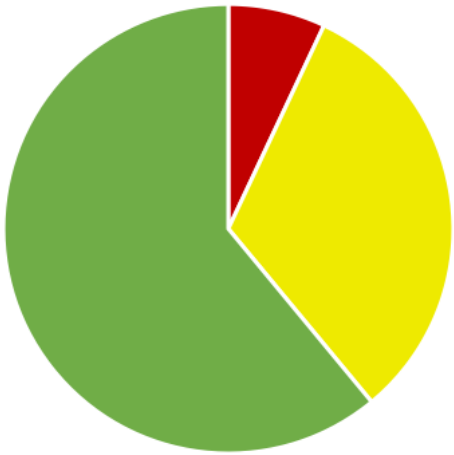
Challenged – “completed”, but over-budget, over-time, missing key functionality

Failed - cancelled

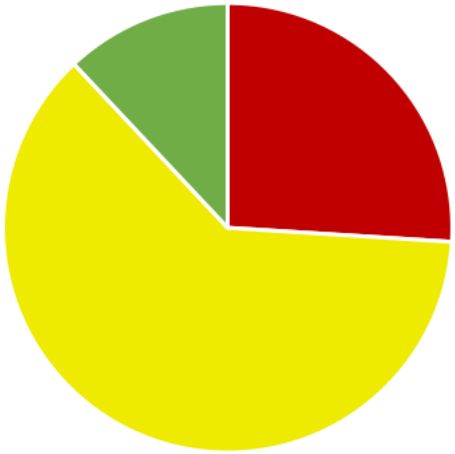
https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

# Standish Group Report - CHAOS

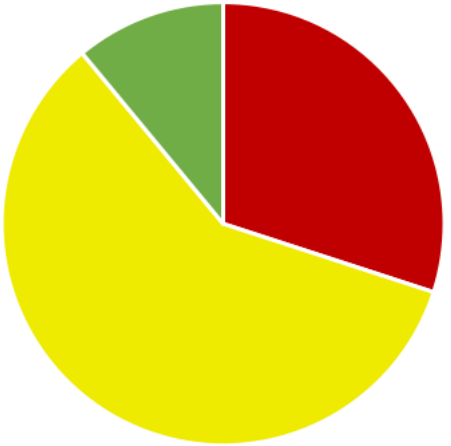
Annual survey of IT executives shows that most software projects are not successful.



Small Projects



Medium Projects



Large Projects

For every 100 project starts, there are 94 restarts

Failed - cancelled

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

# Standish Group Report - CHAOS

**CHAOS RESOLUTION BY AGILE VERSUS WATERFALL**

| SIZE | METHOD | SUCCESSFUL | CHALLENGED | FAILED |
|---|---|---|---|---|
| **All Size Projects** | Agile | 39% | 52% | 9% |
| | Waterfall | 11% | 60% | 29% |
| | | | | |
| **Large Size Projects** | Agile | 18% | 59% | 23% |
| | Waterfall | 3% | 55% | 42% |
| **Medium Size Projects** | Agile | 27% | 62% | 11% |
| | Waterfall | 7% | 68% | 25% |
| **Small Size Projects** | Agile | 58% | 38% | 4% |
| | Waterfall | 44% | 45% | 11% |

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

# Factors Behind Problem Projects

| Factors Underlying Challenged Projects | % of Responses | Factors Underlying Failed Projects | % of Responses |
|---|---|---|---|
| Lack of User Input | 12.8% | Incomplete Reqs and Specs | 13.1% |
| Incomplete Reqs and Specs | 12.3% | Lack of User Input | 12.4% |
| Changing Reqs and Specs | 11.8% | Lack of (Skilled) Resources | 10.6% |
| Lack of Executive Support | 7.5% | Unrealistic Expectations | 9.9% |
| Technology Incompetence | 7.0% | Lack of Executive Support | 9.3% |
| Lack of (Skilled) Resources | 6.4% | Changing Reqs and Specs | 8.7% |
| Unrealistic Expectations | 5.9% | Lack of Planning | 8.1% |
| Unclear Objectives | 5.3% | Didn't Need It Any Longer | 7.5% |
| Unrealistic Time Frames | 4.3% | Lack of IT Management | 6.2% |
| New Technology | 3.7% | Technological Illiteracy | 4.3% |
| Other | 23% | Other | 9.9% |

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

# Waterloo Software

## WaterlooWorks

- Can no longer search for jobs by selecting programs of interest
- Cancellations and rejections do not decrement application counter
- No “quick apply”
- Customizing resumes/letters is harder, more time consuming, more clicks
- Employment history now appears at the end of an application package
- Employment history listed in chronological order
- No Jobmine glitch (which CECA claims to have asked for)
- Can’t view job postings if not in co-op cycle for next term
- Cannot remove a job from the short list after you’ve applied to it
- Match results not released on the same day as ranking deadline
- On employer side – cannot download batch of applications (either download one at a time or download single PDF)
- Takes 10 mins to figure out how to view past coop jobs
- UI doesn’t fit on one screen
- More clicks needed to do most tasks
- Horizontal bar in the job table
- Crashes

# Pain of Bad Requirements

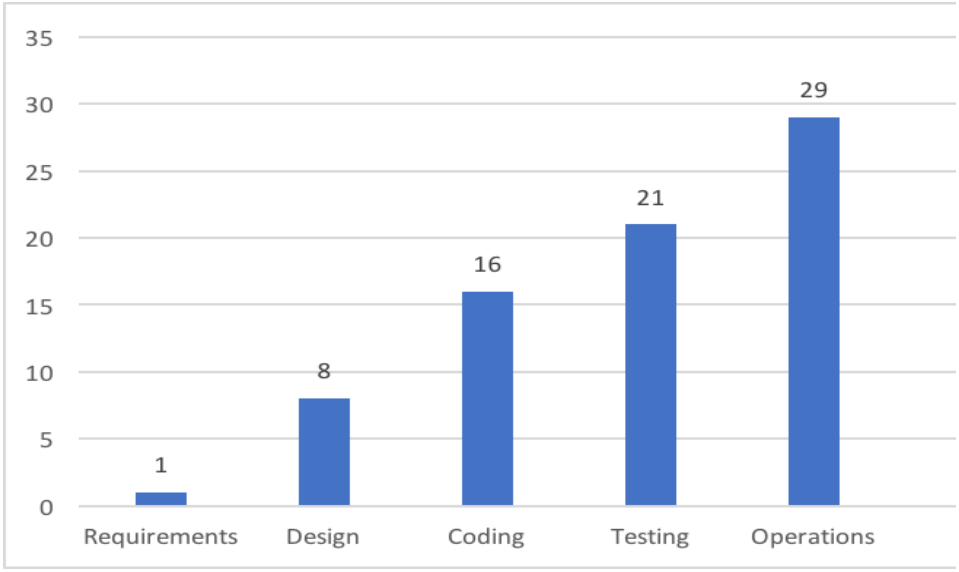
Rework – time and money spent on fixing problems rather than adding value to the software.

- Rework accounts for 30%-50% of development cost
- Requirements errors account for 70%-85% of rework cost

# Pain of Bad Requirements

Rework – time and money spent on fixing problems rather than adding value to the software.

- Rework accounts for 30%-50% of development cost
- Requirements errors account for 70%-85% of rework cost




Cost to fix req error once discovered

Stecklein, et al. “Error Cost Escalation Through the Project Life Cycle”, *INCOSE*, 2004

# Pain of Bad Requirements

- Rework – time and money spent on fixing problems rather than adding value to the software.
  - Rework accounts for 30%-50% of development cost
  - Requirements errors account for 70%-85% of rework cost
- Unstable, changing requirements
  - Reported by 60%-70% of developers
- Ill-understood requirements and priorities – keeps team from focusing on the highest-value features
  - 45% percent of developed software features are never used

# Pain of Bad Requirements

- Rework – time and money spent on fixing problems rather than adding value to the software.
  - Rework accounts for 30%-50% of development cost
  - Requirements errors account for 70%-85% of rework cost
- Unstable, changing requirements
  - Reported by 60%-70% of developers
- Ill-understood requirements and priorities – keeps team from focusing on the highest-value features
  - 45% percent of developed software features are never used

# Value of Better Requirements

More useful software, with less effort

- Helps team focus effort on the most critical functionality – implementing features that are most needed or desired by users
- Enables better project planning – better estimations of the effort and resources needed to develop the software
- Enables better testing – testers develop more accurate test suites; and improved knowledge of risk helps testers to know which functionality needs more scrutiny
- Improves team communication – if documented, then can be shared with new team members to bring them up to speed on project
- Accelerates software development

# Unpacking the Course Title

## Software Requirement

A condition or capability that must be met by a software system (component, extension), for the system to be acceptable (to a user, contract, standard, etc.).

## Analysis

Process of studying and refining the software requirements

## Specification

Documentation that specifies the requirements of a software system (component, extension).

*IEEE Standard Glossary of Software Engineering Terminology*

# Textual Specifications

## 1 Introduction

### 1.1 Purpose

The purpose of this document is to outline the functionality, interfaces, attributes and constraints of four active-safety features for automobiles. These features include blind spot detection (BSD), parking assist (PA), tire pressure monitoring (TPM) and pre/post collision protection (PCP). The models presented in section 3 of this document demonstrate how each system will typically be used, what other systems it interacts with, and the basic behaviour of the system. This information can be used to design a solution that fits the project's requirements.

The intended audience for this document is the stakeholders of the project. This includes but is not limited to the client, developers and regulatory bodies. This specification documents the client requirements of the four active-safety features under design. The clients can use this document to verify the systems under design meet the specified requirements. The developers and designers should use this document to ensure all specified functionality is correct and present in the system. This specification further specifies the systems these features are expected to interface with. Regulatory bodies can use this document to verify compliance with relevant rules and regulations.

The reader of this document is assumed to have basic knowledge of cars and how to operate cars. Knowledge of UML diagrams is also required.

### 1.2 Scope

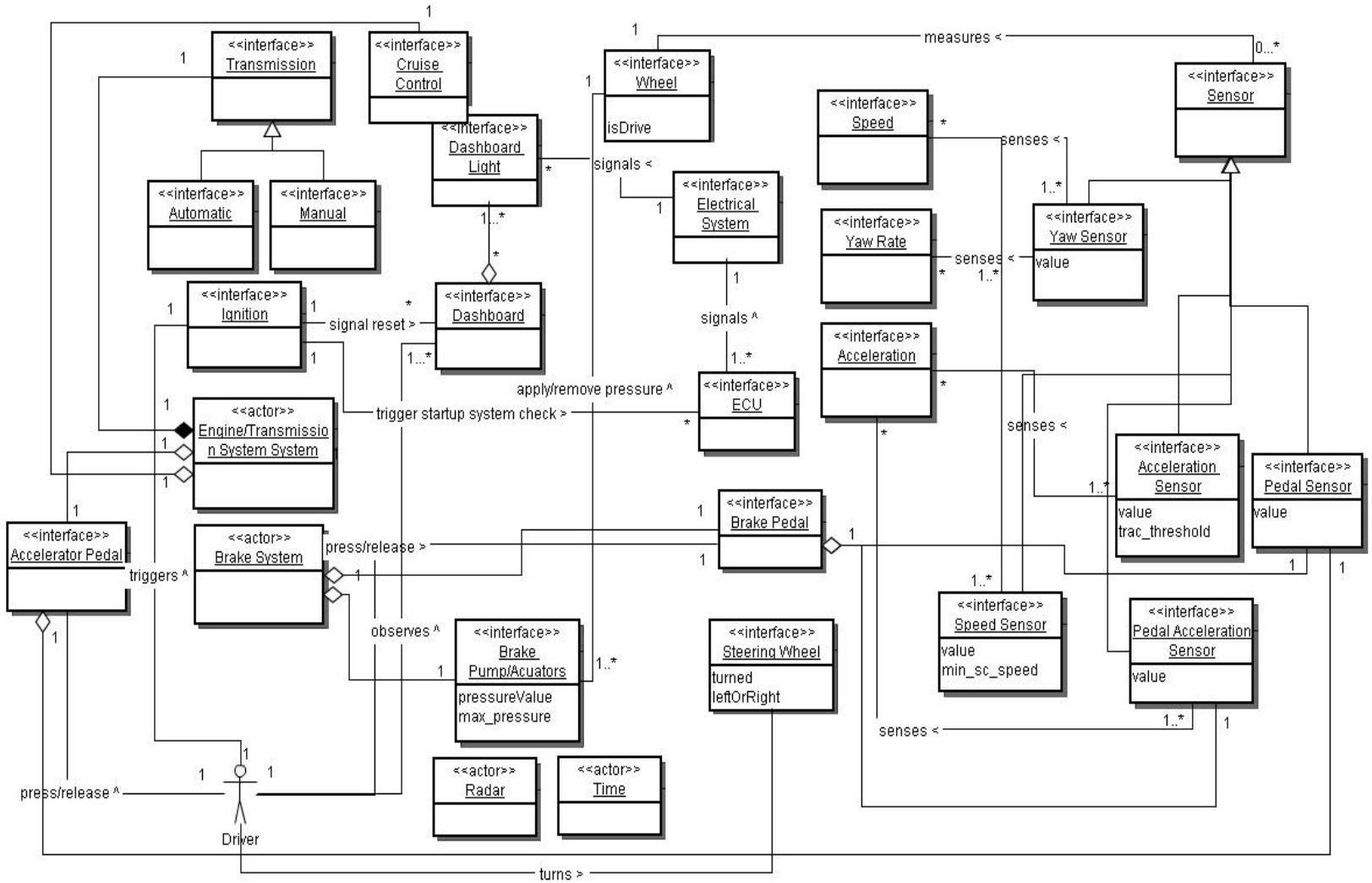
The active-safety system proactively monitors a vehicle's environment and takes action to avoid collisions or mitigate damage if a collision is unavoidable. This system is made up of four subsystems, which include blind spot detection, parking assist, tire pressure monitoring, and pre/post collision protection. These features use software controllers that monitor sensor data, process the sensed data, detect hazards or errors, operate actuators, communicate with other systems and/or log events.

Blind spot detection uses sensors in the car's rear corners to determine if there is an object in the car's blind spot. If an object is detected then the user is alerted visually. If the user signals a lane change, and there is an object in their blind spot then they will be alerted audibly.

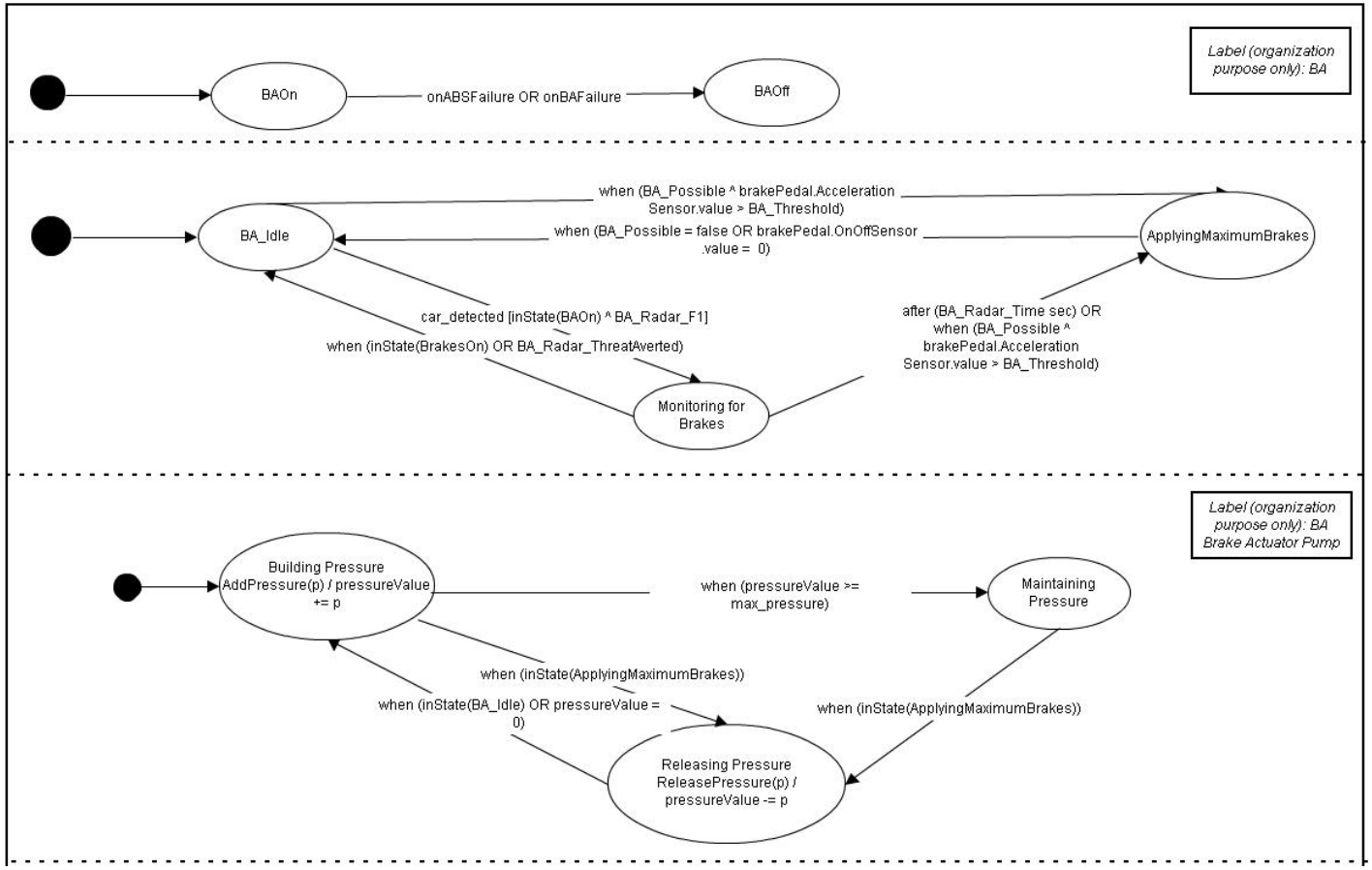
Parking assist uses sensors on the car's front and rear bumpers to alert the driver of near-by objects at low speeds. The system visually and audibly alerts the user of the proximity of objects.

Tire pressure monitoring notifies the user if one or more of their tires has a pressure that varies too greatly from a user desired value. The pressure of each tire in monitored by sensors that are part of the wheels.

# Model Specifications



# Model Specifications



# Best Practices Vary

Best practices vary with the type of the system or project

- Greenfield vs. Brownfield vs. Startup - new product vs. enhancement



- Customer-driven vs. Market-driven


- Web/Mobile App vs. Enterprise System vs. Infrastructure vs. Safety-Critical





- In-house vs. Outsourced


# Customer-Facing Tasks/Roles

Requirements Engineering entails customer-facing activities and roles where you work with people to understand their problems and how they can be addressed with software

- Entrepreneur
- Product/Project Manager
- Ideation roles

Start ups
Tech companies

---

- Business Analyst
- Systems Analyst

Enterprise Software
Consulting Services

---

- Systems Engineer
- Requirements Engineer

Engineered Products
Safety-critical Systems