Software Engineering for Robotics

Introduction

Dr. Alex Mitrevski
Master of Autonomous Systems
Structure

- Robot software development overview
- Software development essentials
Robot Software Development Overview
You might be wondering whether software development is a synonym for programming — it is not.
You might be wondering whether software development is a synonym for programming — it is not.

Programming is simply a process of writing a program for a specific purpose.

Software development is more than writing programs — it is a process of developing individual programs and combining those together into a coherent set that works in unison for achieving a concrete goal.

Particularly in large domains such as robotics, software development is commonly done by teams rather than by individuals. Management is thus an essential element of the software development process.
Software Development vs. Programming

▶ You might be wondering whether software development is a synonym for programming — it is not
  ▶ Programming is simply a process of writing a program for a specific purpose
  ▶ Software development is more than writing programs — it is a process of developing individual programs and combining those together into a coherent set that works in unison for achieving a concrete goal
Software Development vs. Programming

► You might be wondering whether software development is a synonym for programming — it is not
  ► Programming is simply a process of writing a program for a specific purpose
  ► Software development is more than writing programs — it is a process of developing individual programs and combining those together into a coherent set that works in unison for achieving a concrete goal

► Particularly in large domains such as robotics, software development is commonly done by teams rather than by individuals
  ► Management is thus an essential element of the software development process
What is so Special About Robot Software Development?

Hardware

Robots are **hardware platforms that act in the physical world** — this distinguishes them from software-only applications.
What is so Special About Robot Software Development?

**Hardware**

Robots are **hardware platforms that act in the physical world** — this distinguishes them from software-only applications.

**Real-world unpredictability**

Except under special circumstances, **the real world can sometimes be unpredictable**, so robot software needs to take this into account.
What is so Special About Robot Software Development?

**Hardware**

Robots are **hardware platforms that act in the physical world** — this distinguishes them from software-only applications.

**Real-world unpredictability**

Except under special circumstances, **the real world can sometimes be unpredictable**, so robot software needs to take this into account.

**Domain complexity**

**Robotics combines many domains** (e.g. navigation, manipulation, computer vision, etc.) and thus requires the collaboration of a variety of domain experts.
What is so Special About Robot Software Development?

Hardware
Robots are **hardware platforms that act in the physical world** — this distinguishes them from software-only applications.

Real-world unpredictability
Except under special circumstances, the real world can sometimes be unpredictable, so robot software needs to take this into account.

Domain complexity
**Robotics combines many domains** (e.g. navigation, manipulation, computer vision, etc.) and thus requires the collaboration of a variety of domain experts.

Network-based communication
**A modern robot is a distributed system**, which sometimes complicates the use of well-established software paradigms, such as object-oriented programming.
Robot Software and Reusability

- Robot software is typically developed for a specific robot platform.

The question then becomes: Can the developed software be reused on different robots? In some cases, reusability is not possible due to major hardware differences (e.g., software developed for a flying robot is likely to have limited usability for a wheeled robot). More often than not, robots have many physical similarities; it should then, in principle, be possible to adapt the software from one platform for another one. Reusability is often achieved by creating reconfigurable components (at design or runtime).
Robot Software and Reusability

- Robot software is **typically developed for a specific robot platform**
- The question then becomes: Can the developed software be reused on different robots?
Robot Software and Reusability

- Robot software is *typically developed for a specific robot platform*
- The question then becomes: Can the developed software be reused on different robots?
- **In some cases, reusability is not possible due to major hardware differences** (e.g. software developed for a flying robot is likely to have limited usability for a wheeled robot)
Robot Software and Reusability

- Robot software is **typically developed for a specific robot platform**
- The question then becomes: Can the developed software be reused on different robots?
- **In some cases, reusability is not possible due to major hardware differences** (e.g. software developed for a flying robot is likely to have limited usability for a wheeled robot)
- More often than not, robots have many physical similarities though; **it should then, in principle, be possible to adapt the software from one platform for another one**
  - Reusability is often achieved by **creating reconfigurable components** (at design or runtime)
Software Development Essentials
Essential Steps in Software Development

While the concrete details of every software project differ, there are a few general steps that always need to be done:

1. **Requirement elicitation**
2. **Translating requirements into software components**
3. **Software testing**
4. **Component development**

In practice, these are not performed just once in a sequential order, but instead need to inform each other and may need to be performed in an iterative process.
Essential Steps in Software Development

▶ While the concrete details of every software project differ, there are a few general steps that always need to be done:

1. Requirement elicitation
2. Translating requirements into software components
3. Software testing
4. Component development

▶ In practice, these are not performed just once in a sequential order, but instead need to inform each other and may need to be performed in an iterative process.
1. Requirement Elicitation

- When working on your own research projects, the scope of a robotics project is small and you as the developer know what the robot should achieve.

- This is different when a robot is developed for a concrete application, in which case the needs of the robot’s prospective users are of primary importance. A robot that does not fulfill the needs of its users is very unlikely to be used by them.

- Particularly in long-running projects, requirements can change over time; elicitation should not be seen as a one-time process.

- NB: In the research projects during your studies, the users will often be your advisors, so make sure that you always understand their requirements.

Requirements elicitation is a process of identifying the objectives and expectations of a software system before the software development process starts.
1. Requirement Elicitation

- When working on your own research projects, the scope of a robotics project is small and you as the developer know what the robot should achieve.

- This is different when a robot is developed for a concrete application, in which case the needs of the robot’s prospective users are of primary importance.
  - A robot that does not fulfill the needs of its users is very unlikely to be used by them.

Particularly in long-running projects, requirements can change over time; elicitation should not be seen as a one-time process.

NB: In the research projects during your studies, the users will often be your advisors, so make sure that you always understand their requirements.

Requirements elicitation is a process of identifying the objectives and expectations of a software system before the software development process starts.
1. Requirement Elicitation

- When working on your own research projects, the scope of a robotics project is small and you as the developer know what the robot should achieve.

- This is different when a robot is developed for a concrete application, in which case **the needs of the robot’s prospective users are of primary importance**.
  - A robot that does not fulfill the needs of its users is very unlikely to be used by them.

- Particularly in long-running projects, **requirements can change over time**; elicitation should not be seen as a one-time process.
1. Requirement Elicitation

- When working on your own research projects, the scope of a robotics project is small and you as the developer know what the robot should achieve.

- This is different when a robot is developed for a concrete application, in which case the needs of the robot’s prospective users are of primary importance.
  
  - A robot that does not fulfill the needs of its users is very unlikely to be used by them.

- Particularly in long-running projects, requirements can change over time; elicitation should not be seen as a one-time process.

- NB: In the research projects during your studies, the users will often be your advisors, so make sure that you always understand their requirements.
1. Requirement Elicitation

- When working on your own research projects, the scope of a robotics project is small and you as the developer know what the robot should achieve.

- This is different when a robot is developed for a concrete application, in which case **the needs of the robot’s prospective users are of primary importance**.
  - A robot that does not fulfill the needs of its users is very unlikely to be used by them.

- Particularly in long-running projects, **requirements can change over time**; elicitation should not be seen as a one-time process.

- **NB:** In the research projects during your studies, the users will often be your advisors, so make sure that you always understand their requirements.

Requirements elicitation is a process of identifying the objectives and expectations of a software system before the software development process starts.
2. Translating Requirements into Software Components

Once the requirements of a software project are known, the next relevant step is to map them to software components that need to be developed.
2. Translating Requirements into Software Components

Once the requirements of a software project are known, the next relevant step is to map them to software components that need to be developed.

The translation of requirements typically includes the design of a software architecture or a plan on how the new components will be integrated into an existing architecture. During this step, the application programming interfaces (APIs) of the components are defined. However, there is no guarantee that there is a one-to-one mapping between requirements and software components.

The translation of requirements into software components is a process of conceptually designing components and their APIs so that the objectives defined by the requirements can be satisfied.
2. Translating Requirements into Software Components

- Once the requirements of a software project are known, the next relevant step is to map them to software components that need to be developed.

- The translation of requirements typically includes the design of a software architecture or a plan on how the new components will be integrated into an existing architecture.
  - During this step, the application programming interfaces (APIs) of the components are defined.

- In the context of collaborative software development, this would then result in a concrete set of development tasks that each developer can work on.

NB: There is no guarantee that there is a one-to-one mapping between requirements and software components. The translation of requirements into software components is a process of conceptually designing components and their APIs so that the objectives defined by the requirements can be satisfied.
2. Translating Requirements into Software Components

- Once the requirements of a software project are known, the next relevant step is to map them to software components that need to be developed.

- The translation of requirements typically includes the design of a software architecture or a plan on how the new components will be integrated into an existing architecture.
  - During this step, the application programming interfaces (APIs) of the components are defined.

- In the context of collaborative software development, this would then result in a concrete set of development tasks that each developer can work on.

- NB: There is no guarantee that there is a one-to-one mapping between requirements and software components.
2. Translating Requirements into Software Components

- Once the requirements of a software project are known, the next relevant step is to map them to software components that need to be developed.

- The translation of requirements typically includes the design of a software architecture or a plan on how the new components will be integrated into an existing architecture.
  - During this step, the application programming interfaces (APIs) of the components are defined.

- In the context of collaborative software development, this would then result in a concrete set of development tasks that each developer can work on.

- NB: There is no guarantee that there is a one-to-one mapping between requirements and software components.

The translation of requirements into software components is a process of conceptually designing components and their APIs so that the objectives defined by the requirements can be satisfied.
3. Component Development

- After designing the components, the next step is to actually develop them.

The development process should be done according to concrete development and collaboration standards. Typically, each software project has its own rules and guidelines— it is essential to familiarise yourself with these before starting the development. Even for own smaller projects, it is useful to develop own development guidelines and follow best practices so that consistency and reusability can be ensured. Components need to be developed based on their agreed upon design and by following the development guidelines of the overall software project.
3. Component Development

- After designing the components, the next step is to actually develop them.
- The development process should be done according to concrete development and collaboration standards.
3. Component Development

► After designing the components, the next step is to actually develop them.

► The development process should be done according to concrete development and collaboration standards.
  
  ► **Typically, each software project has its own rules and guidelines** — it is essential to familiarise yourself with these before starting the development.
3. Component Development

- After designing the components, the next step is to actually develop them.
- The development process should be done according to concrete development and collaboration standards.
  - Typically, each software project has its own rules and guidelines — it is essential to familiarise yourself with these before starting the development.
  - Even for own smaller projects, it is useful to develop own development guidelines and follow best practices so that consistency and reusability can be ensured.
3. Component Development

- After designing the components, the next step is to actually develop them.

- The development process should be done according to concrete development and collaboration standards.
  - Typically, each software project has its own rules and guidelines — it is essential to familiarise yourself with these before starting the development.
  - Even for own smaller projects, it is useful to develop own development guidelines and follow best practices so that consistency and reusability can be ensured.

Components need to be developed based on their agreed upon design and by following the development guidelines of the overall software project.
4. Software Testing

While developing components, it is necessary to ensure that they do what you actually intend them to do — this is the purpose of testing.

Testing can be done at different levels:

- at the level of individual components (component testing)
- to verify that multiple components work well together (integration testing)
- at the level of a complete system (system testing)

Testing should accompany the complete development process and is often supported by automated tools.
4. Software Testing

- While developing components, it is necessary to ensure that they do what you actually intend them to do — this is the purpose of testing.

- Testing can be done at different levels:
4. Software Testing

- While developing components, it is necessary to ensure that they do what you actually intend them to do — this is the purpose of testing

- Testing can be done at different levels:
  - at the level of individual components (component testing)
4. Software Testing

► While developing components, it is necessary to ensure that they do what you actually intend them to do — this is the purpose of testing

► Testing can be done at different levels:
  ► at the level of individual components (**component testing**)
  ► to verify that multiple components work well together (**integration testing**)
4. Software Testing

- While developing components, it is necessary to ensure that they do what you actually intend them to do — this is the purpose of testing.

- Testing can be done at different levels:
  - at the level of individual components (component testing)
  - to verify that multiple components work well together (integration testing)
  - at the level of a complete system (system testing)

Testing should accompany the complete development process and is often supported by automated tools.
4. Software Testing

▶ While developing components, it is necessary to ensure that they do what you actually intend them to do — this is the purpose of testing

▶ Testing can be done at different levels:
  ▶ at the level of individual components (component testing)
  ▶ to verify that multiple components work well together (integration testing)
  ▶ at the level of a complete system (system testing)

▶ Testing should accompany the complete development process and is often supported by automated tools
While developing components, it is necessary to ensure that they do what you actually intend them to do — this is the purpose of testing.

Testing can be done at different levels:
- at the level of individual components (component testing)
- to verify that multiple components work well together (integration testing)
- at the level of a complete system (system testing)

Testing should accompany the complete development process and is often supported by automated tools.

Software testing is a process of ensuring that developed components or the system as a whole comply with their actual requirements.