Carsten Knoll, Robert Heedt
Faculty of Electrical and Computer Engineering, Institute of Control Theory

“Automatic Control Knowledge Repository” – A Computational Approach for Simpler and More Robust Reproducibility of Results in Control Theory

ICSTCC, Sinaia, Romania (remote participation), 2020-10-08

Motivation

Reproduction of Scientific Results (1)

- Scientific communication: traditionally based on natural language (+ mathematics + diagrams)
- Growing importance of computational methods
  - Simulations
  - Numerical linear algebra
  - Symbolic calculations with CAS

Proceedings in Applied Mathematics and Mechanics, 6/10/2020

Trajectory Planning for Closed Kinematic Chains Applied to Cooperative Motions in Health Care

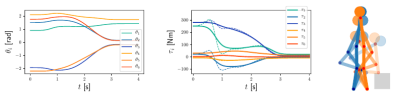
Carsten Knoll[1,*], **Xuehua Jia**[1], and **Robert Heedt**[1]

[1] Institut für Regelungs- und Steuerungstheorie, Fakultät Elektrotechnik und Informationstechnik, Technische Universität Dresden, 01062 Dresden

Physical interaction of humans can be a challenging. For some practically relevant situations a reasonable model is given by a closed kinematic chain of a planar rigid body mechanism. From a control perspective a cooperative motion, e. g. assisted standing-up, can be seen as an optimal control problem (OCP) with input and state restrictions. Due to the algebraic constraints and the comparably high number of joints involved, proper formulation of such a problem is not trivial. In this contribution we compare two different modeling approaches: discretization of the full dynamical problem and iterative solution of consecutive stationary problems. The latter turns out to be significantly faster while still providing – in some sense – optimal solutions.

1 Introduction: Modelling of Cooperative Standing-Up Motion

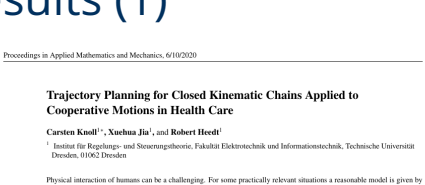
Motivated e. g. by [1] and references therein we consider the basic mechanical aspects of cooperative standing-up motion, which takes place e. g. in geriatric care. The goal of this contribution is not to provide realistic numbers but rather a methodology to simplify further research. Following [1] we model each of the two persons (care giver and receiver) as chains of $\bar{n}_L = 3$ links (rigid bodies) connected by $\bar{n}_J$ actuated revolute joints and assume that both chains are fixed to the same basis (i. e. the ground) and the motion is entirely planar. We further assume that the torque which can be applied to each joint is bounded. The connection of the two kinematic chains (i. e. persons) is also modeled as an (unactuated) revolute joint. The resulting mechanism thus has $n_L = 2\bar{n}_L = 6$ links and $n_J = 2\bar{n}_J + 1 = 7$ joints. We denote the configuration coordinates (actuated joint angles) with $(\theta_1, ..., \theta_{2\bar{n}_J})^T =: \boldsymbol{\theta}$ and the respective torques with $(\tau_1, ..., \tau_{2\bar{n}_J})^T =: \boldsymbol{\tau}$. The degree of freedom of this single closed loop is $F = n_J - 3 = 4$, cf. [2, Section 1.2.1]. The closing condition of the loop – enforcing that both independent link sequences have the connection joint as common point – introduces a constraint expressable as two scalar algebraic equations.



Motivation

Reproduction of Scientific Results (1)

- Scientific communication: traditionally based on natural language (+ mathematics + diagrams)
- Growing importance of computational methods
  - Simulations
  - Numerical linear algebra
  - Symbolic calculations with CAS
- Understanding and reproduction depends on access to these methods (sources and executables)



Proceedings in Applied Mathematics and Mechanics, 6/10/2020

Trajectory Planning for Closed Kinematic Chains Applied to Cooperative Motions in Health Care

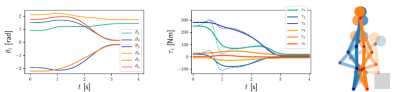
Carsten Knoll[1,*], **Xuehua Jia**[1], and **Robert Heedt**[1]

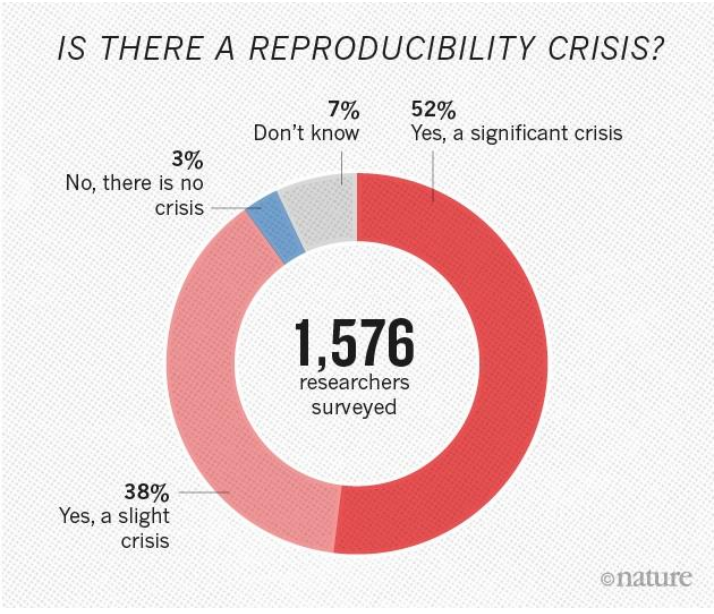
[1] Institut für Regelungs- und Steuerungstheorie, Fakultät Elektrotechnik und Informationstechnik, Technische Universität Dresden, 01062 Dresden

Physical interaction of humans can be a challenging. For some practically relevant situations a reasonable model is given by a closed kinematic chain of a planar rigid body mechanism. From a control perspective a cooperative motion, e. g. assisted standing-up, can be seen as an optimal control problem (OCP) with input and state restrictions. Due to the algebraic constraints and the comparably high number of joints involved, proper formulation of such a problem is not trivial. In this contribution we compare two different modeling approaches: discretization of the full dynamical problem and iterative solution of consecutive stationary problems. The latter turns out to be significantly faster while still providing – in some sense – optimal solutions.

1 Introduction: Modelling of Cooperative Standing-Up Motion

Motivated e. g. by [1] and references therein we consider the basic mechanical aspects of cooperative standing-up motion, which takes place e. g. in geriatric care. The goal of this contribution is not to provide realistic numbers but rather a methodology to simplify further research. Following [1] we model each of the two persons (care giver and receiver) as chains of $n_L = 3$ links (rigid bodies) connected by $\bar{n}_J$ actuated revolute joints and assume that both chains are fixed to the same basis (i. e. the ground) and the motion is entirely planar. We further assume that the torque which can be applied to each joint is bounded. The connection of the two kinematic chains (i. e. persons) is also modeled as an (unactuated) revolute joint. The resulting mechanism thus has $n_L = 2\bar{n}_L = 6$ links and $n_J = 2\bar{n}_J + 1 = 7$ joints. We denote the configuration coordinates (actuated joint angles) with $(\theta_1, ..., \theta_{2\bar{n}_J})^T =: \boldsymbol{\theta}$ and the respective torques with $(\tau_1, ..., \tau_{2\bar{n}_J})^T =: \boldsymbol{\tau}$. The degree of freedom of this single closed loop is $F = n_J - 3 = 4$, cf. [2, Section 1.2.1]. The closing condition of the loop – enforcing that both independent link sequences have the connection joint as common point – introduces a constraint expressable as two scalar algebraic equations.





source: https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970 (2016)

source: https://www.nature.com/articles/467775a/box/2 (2010)

Outline

✓ **Motivation: Reproducibility Challenge**
☐ Existing Approaches
☐ Our Approach
☐ Implementation Status & Demo
☐ Conclusion & Outlook

Outline

✓ Motivation: Reproducibility Challenge
☐ **Existing Approaches**
☐ Our Approach
☐ Implementation Status & Demo
☐ Conclusion & Outlook

Collaboration via Distributed Version Control Systems

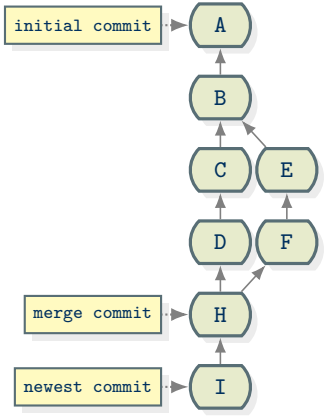
- DVCS: Keeping track of changes
  (Who? When? What?)

Collaboration via Distributed Version Control Systems

- DVCS: Keeping track of changes
  (Who? When? What?)
- Defacto standard: *git* repository

Collaboration via Distributed Version Control Systems

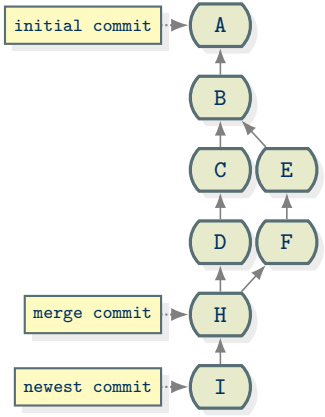
- DVCS: Keeping track of changes
  (Who? When? What?)
- Defacto standard: *git* repository
- Incremental snapshots of working directory +meta data
- stored in a *hash tree*



Collaboration via Distributed Version Control Systems

- DVCS: Keeping track of changes
  (Who? When? What?)
- Defacto standard: *git* repository
- Incremental snapshots of working directory +meta data
- stored in a *hash tree*

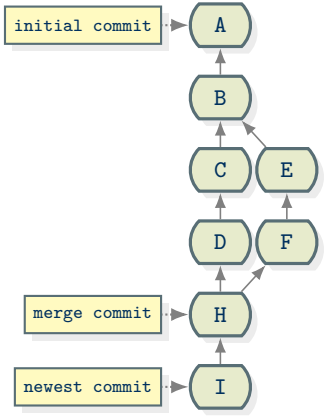
- Allows temporary parallelism (*fork* → *merge*)



Collaboration via Distributed Version Control Systems

- DVCS: Keeping track of changes
  (Who? When? What?)
- Defacto standard: *git* repository
- Incremental snapshots of working directory +meta data
- stored in a *hash tree*

- Allows temporary parallelism (*fork* $\rightarrow$ *merge*)
- Mechanisms to resolve merge conflicts

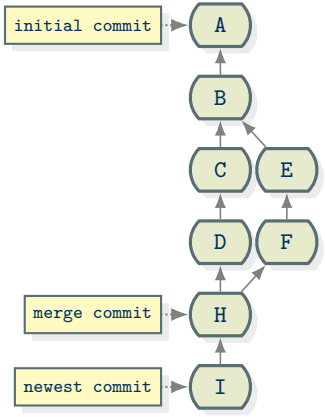


Collaboration via Distributed Version Control Systems

- DVCS: Keeping track of changes
  (Who? When? What?)
- Defacto standard: *git* repository
- Incremental snapshots of working directory +meta data
- stored in a *hash tree*

- Allows temporary parallelism (*fork* $\rightarrow$ *merge*)
- Mechanisms to resolve merge conflicts

- $\exists$ platforms for public repository-hosting
  (github, gitlab, codeberg, ...)



Automated Tests and Continuous Integration Services

- **Automated Test**: Software that runs other software
- Compare result to (hardcoded) expected results

Automated Tests and Continuous Integration Services

- **Automated Test**: Software that runs other software
- Compare result to (hardcoded) expected results
- (-) Unproductive code (additional effort)

Automated Tests and Continuous Integration Services

- **Automated Test**: Software that runs other software
- Compare result to (hardcoded) expected results
- (-) Unproductive code (additional effort)
- (+) Identify unwanted behavior (bugs)

Automated Tests and Continuous Integration Services

- **Automated Test**: Software that runs other software
- Compare result to (hardcoded) expected results

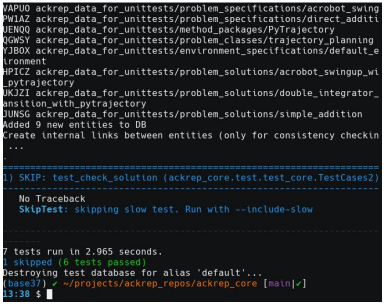
- (-) Unproductive code (additional effort)
- (+) Identify unwanted behavior (bugs)
- (+) Support different execution environments
  (operating systems, library versions, ...)

Automated Tests and Continuous Integration Services

- **Automated Test**: Software that runs other software
- Compare result to (hardcoded) expected results

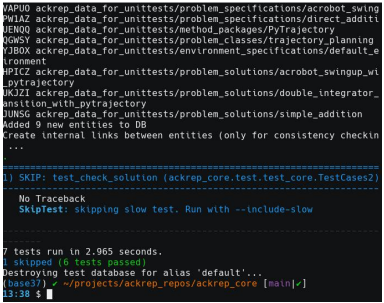
- (-) Unproductive code (additional effort)
- (+) Identify unwanted behavior (bugs)
- (+) Support different execution environments
  (operating systems, library versions, ...)

  $\Rightarrow \exists$ projects with many test cases ($10^1 \ldots 10^3$)



Automated Tests and Continuous Integration Services

- **Automated Test**: Software that runs other software
- Compare result to (hardcoded) expected results



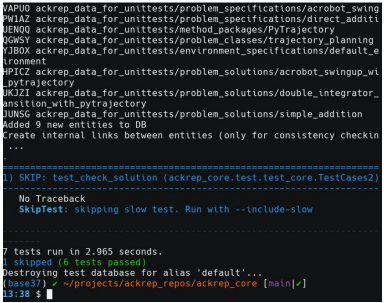
- (-) Unproductive code (additional effort)
- (+) Identify unwanted behavior (bugs)
- (+) Support different execution environments
  (operating systems, library versions, ...)

  $\Rightarrow \exists$ projects with many test cases ($10^1 \dots 10^3$)

- **Continuous Integration**:
  Dedicated service to automated test running and result handling

Automated Tests and Continuous Integration Services

- **Automated Test**: Software that runs other software
- Compare result to (hardcoded) expected results



- (-) Unproductive code (additional effort)
- (+) Identify unwanted behavior (bugs)
- (+) Support different execution environments
  (operating systems, library versions, ...)

  $\Rightarrow \exists$ projects with many test cases ($10^1 \dots 10^3$)

- **Continuous Integration**:
  Dedicated service to automated test running and result handling
- Triggered e. g. by a new commit to the main branch of a repo

Insufficiencies of Existing Approaches

- Low prevalence of source code publication together with scientific results

Insufficiencies of Existing Approaches

- Low prevalence of source code publication together with scientific results
- Even lower prevalence of CI usage

Insufficiencies of Existing Approaches

- Low prevalence of source code publication together with scientific results
- Even lower prevalence of CI usage

Insufficiencies of Existing Approaches

- Low prevalence of source code publication together with scientific results
- Even lower prevalence of CI usage

Plausible reason:

- Target audience: (general) software projects
- Lack of structure and functionality specific to computational science

Insufficiencies of Existing Approaches

- Low prevalence of source code publication together with scientific results
- Even lower prevalence of CI usage

Plausible reason:

- Target audience: (general) software projects
- Lack of structure and functionality specific to computational science

Insufficiencies of Existing Approaches

- Low prevalence of source code publication together with scientific results
- Even lower prevalence of CI usage

Plausible reason:

- Target audience: (general) software projects
- Lack of structure and functionality specific to computational science

Research question:

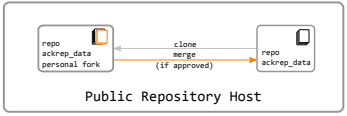
What kind of science specific structure and functionality is suited to mitigate the reproducibility crisis?

Outline

✓ Motivation: Reproducibility Challenge
✓ Existing Approaches
☐ **Our Approach**
☐ Implementation Status & Demo
☐ Conclusion & Outlook

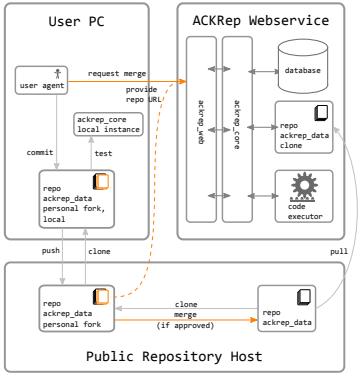
Integration Concept: “Automatic Control Knowledge Repository”

- Establish a (canonical) repository for
  - content-related software (e. g. control algorithms),
  - organizational software (automated tests)
  - and metadata in a **formal structure**



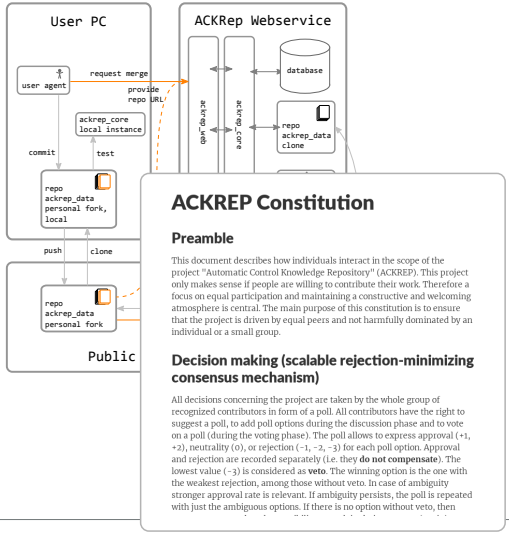
Integration Concept: “Automatic Control Knowledge Repository”

- Establish a (canonical) repository for
  - content-related software (e. g. control algorithms),
  - organizational software (automated tests)
  - and metadata in a **formal structure**
- Establish supportive tools and a web service for that repository
  - providing CI functionality tailored to its structure and to the needs of systems and control engineering



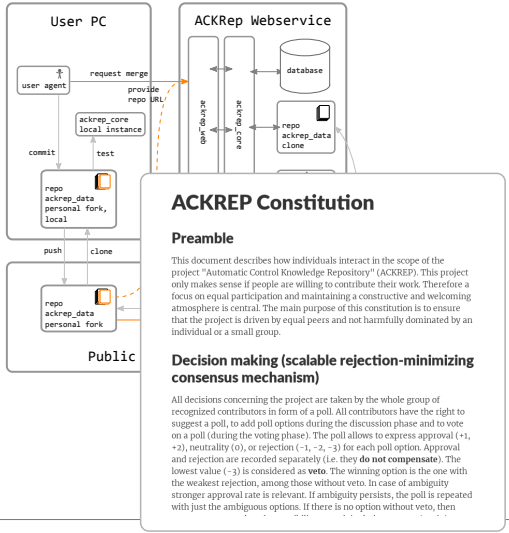
Integration Concept: “Automatic Control Knowledge Repository”

- Establish a (canonical) repository for
  - content-related software (e. g. control algorithms),
  - organizational software (automated tests)
  - and metadata in a **formal structure**
- Establish supportive tools and a web service for that repository
  - providing CI functionality tailored to its structure and to the needs of systems and control engineering
- Establish a set of self-governance rules
  - (Distribute the decision making among active contributors)



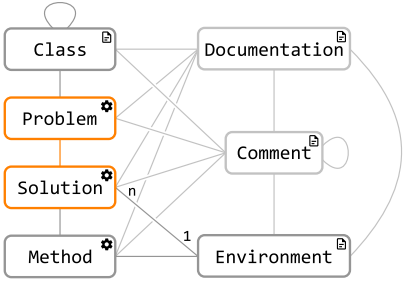
Integration Concept: “Automatic Control Knowledge Repository”

- Establish a (canonical) repository for
  - content-related software (e. g. control algorithms),
  - organizational software (automated tests)
  - and metadata in a **formal structure**
- Establish supportive tools and a web service for that repository
  - providing CI functionality tailored to its structure and to the needs of systems and control engineering
- Establish a set of self-governance rules
  - (Distribute the decision making among active contributors)
- Working title:
  “Automatic Control Knowledge Repository” (ackrep)



Proposed Repository Structure

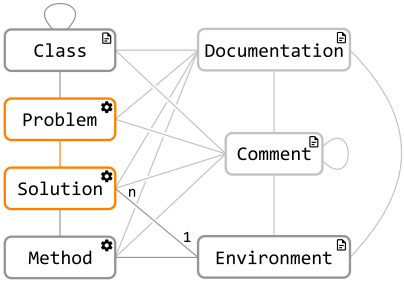
Reproducibility → entity types:



Proposed Repository Structure

Reproducibility → entity types:

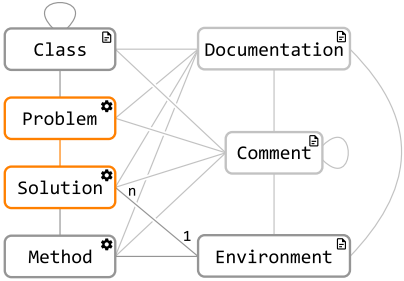
- Is a **Problem** solved by a **Solution**?



Proposed Repository Structure

Reproducibility → entity types:

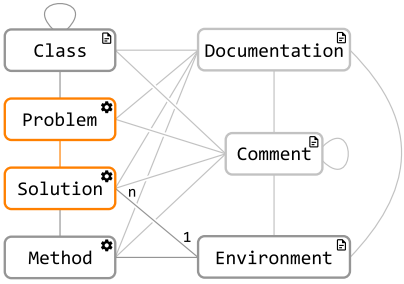
- Is a **Problem** solved by a **Solution**?
  - `ProblemSpecification` (code)
  - `ProblemSolution` (code)



Proposed Repository Structure

Reproducibility → entity types:

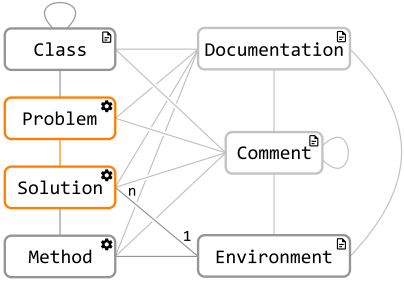
- Is a **Problem** solved by a **Solution**?
  - `ProblemSpecification` (code)
  - `ProblemSolution` (code)
- What **method(s)** does the solution code use?
  - `MethodPackage` (code)



Proposed Repository Structure

Reproducibility → entity types:

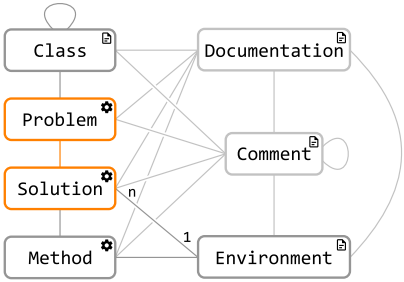
- Is a **Problem** solved by a **Solution**?
  - `ProblemSpecification` (code)
  - `ProblemSolution` (code)
- What **method(s)** does the solution code use?
  - `MethodPackage` (code)
- Which computational **environment** is required?
  - `EnvironmentSpecification` (meta data)



Proposed Repository Structure

Reproducibility → entity types:

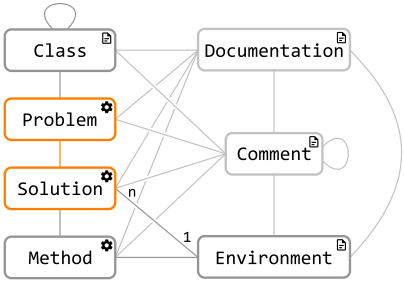
- Is a **Problem** solved by a **Solution**?
  - `ProblemSpecification` (code)
  - `ProblemSolution` (code)
- What **method(s)** does the solution code use?
  - `MethodPackage` (code)
- Which computational **environment** is required?
  - `EnvironmentSpecification` (meta data)
- To which **class** does a Problem belong?
  - `ProblemClass` (meta data)



Proposed Repository Structure

Reproducibility → entity types:

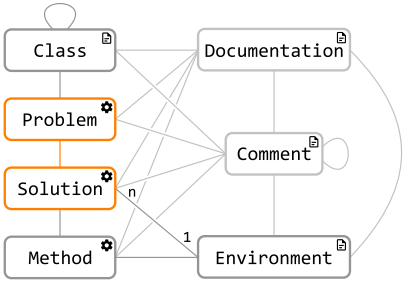
- Is a **Problem** solved by a **Solution**?
  - `ProblemSpecification` (code)
  - `ProblemSolution` (code)
- What **method(s)** does the solution code use?
  - `MethodPackage` (code)
- Which computational **environment** is required?
  - `EnvironmentSpecification` (meta data)
- To which **class** does a Problem belong?
  - `ProblemClass` (meta data)
- Which further **information** exists?
  - `Documentation` (text)
  - `Comment` (text)



Proposed Repository Structure

Reproducibility → entity types:

- Is a **Problem** solved by a **Solution**?
  - `ProblemSpecification` (code)
  - `ProblemSolution` (code)
- What **method(s)** does the solution code use?
  - `MethodPackage` (code)
- Which computational **environment** is required?
  - `EnvironmentSpecification` (meta data)
- To which **class** does a Problem belong?
  - `ProblemClass` (meta data)
- Which further **information** exists?
  - `Documentation` (text)
  - `Comment` (text)

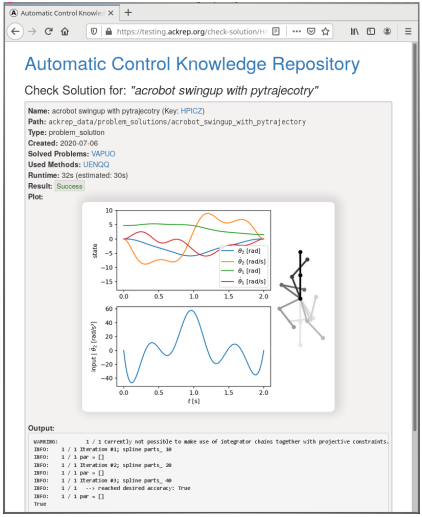


Entities:

- Identified via unique keys
- Stored in (text-based) files
- Managed via DVCS (git)
- Publicly hosted

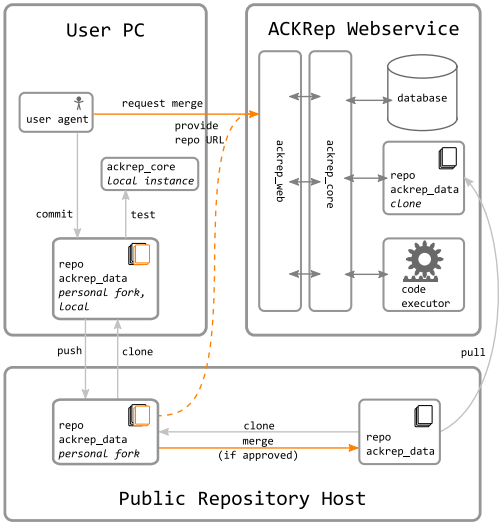
Supportive Webservice and Tools

- Webservice:
  - Execute `Solutions` against `Problems` (≙ Continuous Integration)
  - Present results
  - Make available knowledge accessible



Supportive Webservice and Tools

- Webservice:
  - Execute `Solutions` against `Problems` (≙ Continuous Integration)
  - Present results
  - Make available knowledge accessible
- Tools:
  - Locally check entities
  - Support debugging process



Self-Governance Rules

Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure

Self-Governance Rules

Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure
  - Technical: public repo or webservice could vanish

Self-Governance Rules

Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure
  - Technical: public repo or webservice could vanish
    $\rightsquigarrow$ uncritical due to DVCS (open sources of data and software)

Self-Governance Rules

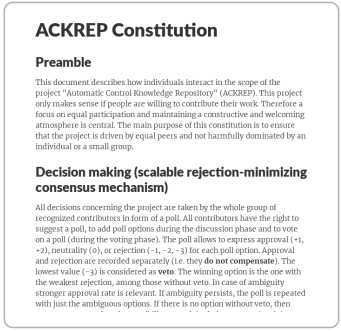
Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure
  - Technical: public repo or webservice could vanish
    $\rightsquigarrow$ uncritical due to DVCS (open sources of data and software)
  - Personal: bad management (rejected merge-request etc.)

Self-Governance Rules

Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure
  - Technical: public repo or webservice could vanish
    $\rightsquigarrow$ uncritical due to DVCS (open sources of data and software)
  - Personal: bad management (rejected merge-request etc.)
- Solution approach: **peer-based decision making**



ACKREP Constitution

Preamble

This document describes how individuals interact in the scope of the project "Automatic Control Knowledge Repository" (ACKREP). This project only makes sense if people are willing to contribute their work. Therefore a focus on equal participation and maintaining a constructive and welcoming atmosphere is central. The main purpose of this constitution is to ensure that the project is driven by equal peers and not harmfully dominated by an individual or a small group.

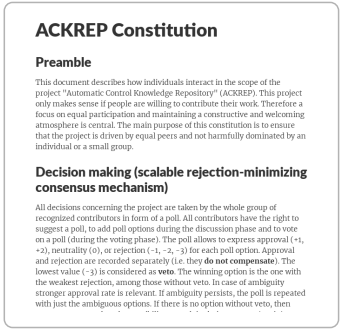
Decision making (scalable rejection-minimizing consensus mechanism)

All decisions concerning the project are taken by the whole group of recognized contributors in form of a poll. All contributors have the right to suggest a poll, to add poll options during the discussion phase and to vote on a poll (during the voting phase). The poll allows to express approval (+1, +2), neutrality (0), or rejection (−1, −2, −3) for each poll option. Approval and rejection are recorded separately (i.e. they **do not compensate**). The lowest value (−3) is considered as **veto**. The winning option is the one with the weakest rejection, among those without veto. In case of ambiguity stronger approval rate is relevant. If ambiguity persists, the poll is repeated with just the ambiguous options. If there is no option without veto, then

Self-Governance Rules

Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure
  - Technical: public repo or webservice could vanish
    $\rightsquigarrow$ uncritical due to DVCS (open sources of data and software)
  - Personal: bad management (rejected merge-request etc.)
- Solution approach: **peer-based decision making**
  - … about contributions (merge-requests)
  - … about the rules themselves



ACKREP Constitution

Preamble

This document describes how individuals interact in the scope of the project "Automatic Control Knowledge Repository" (ACKREP). This project only makes sense if people are willing to contribute their work. Therefore a focus on equal participation and maintaining a constructive and welcoming atmosphere is central. The main purpose of this constitution is to ensure that the project is driven by equal peers and not harmfully dominated by an individual or a small group.

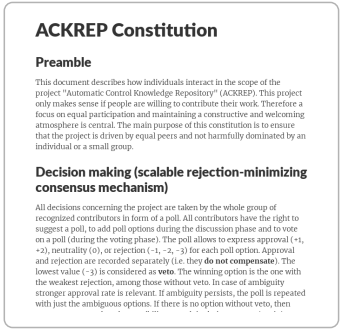
Decision making (scalable rejection-minimizing consensus mechanism)

All decisions concerning the project are taken by the whole group of recognized contributors in form of a poll. All contributors have the right to suggest a poll, to add poll options during the discussion phase and to vote on a poll (during the voting phase). The poll allows to express approval (+1, +2), neutrality (0), or rejection (−1, −2, −3) for each poll option. Approval and rejection are recorded separately (i.e. they **do not compensate**). The lowest value (−3) is considered as **veto**. The winning option is the one with the weakest rejection, among those without veto. In case of ambiguity stronger approval rate is relevant. If ambiguity persists, the poll is repeated with just the ambiguous options. If there is no option without veto, then

Self-Governance Rules

Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure
  - Technical: public repo or webservice could vanish
    $\rightsquigarrow$ uncritical due to DVCS (open sources of data and software)
  - Personal: bad management (rejected merge-request etc.)
- Solution approach: **peer-based decision making**
  - … about contributions (merge-requests)
  - … about the rules themselves
  - Entitled to vote: previous contributors



ACKREP Constitution

Preamble

This document describes how individuals interact in the scope of the project "Automatic Control Knowledge Repository" (ACKREP). This project only makes sense if people are willing to contribute their work. Therefore a focus on equal participation and maintaining a constructive and welcoming atmosphere is central. The main purpose of this constitution is to ensure that the project is driven by equal peers and not harmfully dominated by an individual or a small group.

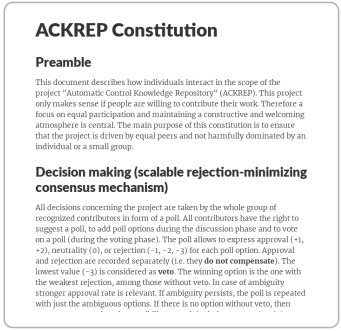
Decision making (scalable rejection-minimizing consensus mechanism)

All decisions concerning the project are taken by the whole group of recognized contributors in form of a poll. All contributors have the right to suggest a poll, to add poll options during the discussion phase and to vote on a poll (during the voting phase). The poll allows to express approval (+1, +2), neutrality (0), or rejection (−1, −2, −3) for each poll option. Approval and rejection are recorded separately (i.e. they **do not compensate**). The lowest value (−3) is considered as **veto**. The winning option is the one with the weakest rejection, among those without veto. In case of ambiguity stronger approval rate is relevant. If ambiguity persists, the poll is repeated with just the ambiguous options. If there is no option without veto, then

Self-Governance Rules

Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure
  - Technical: public repo or webservice could vanish
    $\rightsquigarrow$ uncritical due to DVCS (open sources of data and software)
  - Personal: bad management (rejected merge-request etc.)
- Solution approach: **peer-based decision making**
  - … about contributions (merge-requests)
  - … about the rules themselves
  - Entitled to vote: previous contributors
  - Method: *"Scalable Rejection Minimizing Consensus Mechanism"*



ACKREP Constitution

Preamble

This document describes how individuals interact in the scope of the project "Automatic Control Knowledge Repository" (ACKREP). This project only makes sense if people are willing to contribute their work. Therefore a focus on equal participation and maintaining a constructive and welcoming atmosphere is central. The main purpose of this constitution is to ensure that the project is driven by equal peers and not harmfully dominated by an individual or a small group.

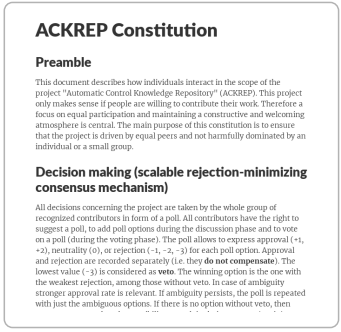
Decision making (scalable rejection-minimizing consensus mechanism)

All decisions concerning the project are taken by the whole group of recognized contributors in form of a poll. All contributors have the right to suggest a poll, to add poll options during the discussion phase and to vote on a poll (during the voting phase). The poll allows to express approval (+1, +2), neutrality (0), or rejection (−1, −2, −3) for each poll option. Approval and rejection are recorded separately (i.e. they **do not compensate**). The lowest value (−3) is considered as **veto**. The winning option is the one with the weakest rejection, among those without veto. In case of ambiguity stronger approval rate is relevant. If ambiguity persists, the poll is repeated with just the ambiguous options. If there is no option without veto, then

Self-Governance Rules

Canonical repository + webservice $\rightarrow$ central infrastructure

- Desirable: makes results/implementations easily findable
- Problems: single point of failure
  - Technical: public repo or webservice could vanish
    $\rightsquigarrow$ uncritical due to DVCS (open sources of data and software)
  - Personal: bad management (rejected merge-request etc.)
- Solution approach: **peer-based decision making**
  - … about contributions (merge-requests)
  - … about the rules themselves
  - Entitled to vote: previous contributors
  - Method: *"Scalable Rejection Minimizing Consensus Mechanism"*
- Fallback option: *hard fork* of software and data (possible due to free license)



ACKREP Constitution

Preamble

This document describes how individuals interact in the scope of the project "Automatic Control Knowledge Repository" (ACKREP). This project only makes sense if people are willing to contribute their work. Therefore a focus on equal participation and maintaining a constructive and welcoming atmosphere is central. The main purpose of this constitution is to ensure that the project is driven by equal peers and not harmfully dominated by an individual or a small group.

Decision making (scalable rejection-minimizing consensus mechanism)

All decisions concerning the project are taken by the whole group of recognized contributors in form of a poll. All contributors have the right to suggest a poll, to add poll options during the discussion phase and to vote on a poll (during the voting phase). The poll allows to express approval (+1, +2), neutrality (0), or rejection (−1, −2, −3) for each poll option. Approval and rejection are recorded separately (i.e. they **do not compensate**). The lowest value (−3) is considered as **veto**. The winning option is the one with the weakest rejection, among those without veto. In case of ambiguity stronger approval rate is relevant. If ambiguity persists, the poll is repeated with just the ambiguous options. If there is no option without veto, then

Outline

✓ Motivation: Reproducibility Challenge
✓ Existing Approaches
✓ Our Approach
☐ **Implementation Status & Demo**
☐ Conclusion & Outlook

Current Implementation Status

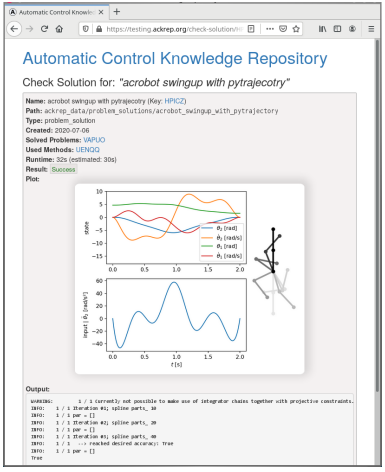
- ∃ `ackrep_core` (GPLv3)
- ∃ `ackrep_web` (GPLv3)
- ∃ `ackrep_data`
  - Default license GPLv3
  - Demonstration examples only
  - Control theoretic content in preparation

Current Implementation Status

- ∃ `ackrep_core` (GPLv3)
- ∃ `ackrep_web` (GPLv3)
- ∃ `ackrep_data`
  - Default license GPLv3
  - Demonstration examples only
  - Control theoretic content in preparation

⇒ Early prototype, proof of concept

Live Demonstration



https://testing.ackrep.org

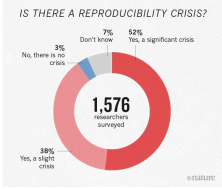
Outline

✓ Motivation: Reproducibility Challenge
✓ Existing Approaches
✓ Our Approach
✓ Implementation Status & Demo
☐ **Conclusion & Outlook**

Conclusion & Outlook

Summary:

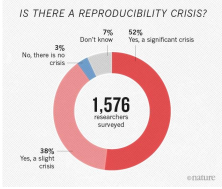
- Reproducibility of computational results is still an unsolved challenge



Conclusion & Outlook

Summary:

- Reproducibility of computational results is still an unsolved challenge
- Existing approaches (public code, CI) not widely enough used

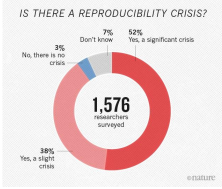


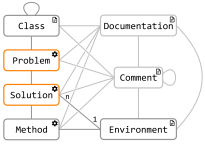
Conclusion & Outlook

Summary:

- Reproducibility of computational results is still an unsolved challenge
- Existing approaches (public code, CI) not widely enough used

- Proposal: Combination of existing components
  - Structured repository



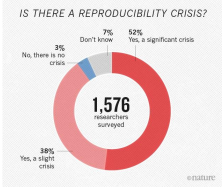


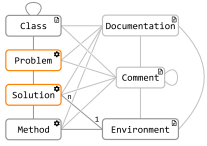
Conclusion & Outlook

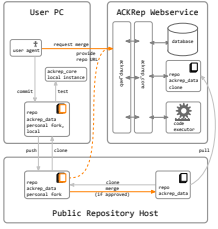
Summary:

- Reproducibility of computational results is still an unsolved challenge
- Existing approaches (public code, CI) not widely enough used

- Proposal: Combination of existing components
  - Structured repository
  - Tailored webservice





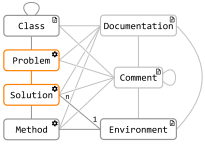


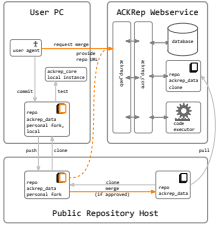
Conclusion & Outlook

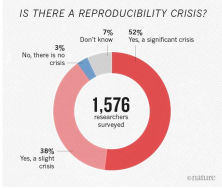
Summary:

- Reproducibility of computational results is still an unsolved challenge
- Existing approaches (public code, CI) not widely enough used

- Proposal: Combination of existing components
  - Structured repository
  - Tailored webservice
  - Peer-based organization structure







ACKREP Constitution

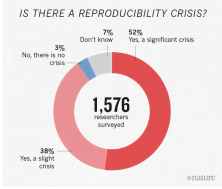
Preamble

This document describes how individuals interact in the scope of the project "Automatic Control Knowledge Repository" (ACKREP). This project only makes sense if people are willing to contribute their work. Therefore a focus on equal participation and maintaining a constructive and welcoming atmosphere is central. The main purpose of this constitution is to ensure that the project is driven by equal peers and not harmfully dominated by an individual or a small group.

Decision making (scalable rejection-minimizing consensus mechanism)

All decisions concerning the project are taken by the whole group of recognized contributors in form of a poll. All contributors have the right to suggest a poll, to add poll options during the discussion phase and to vote on a poll (during the voting phase). The poll allows to express approval (+1, +2), neutrality (0), or rejection (−1, −2, −3) for each poll option. Approval and rejection are recorded separately (i.e. they **do not compensate**). The lowest value (−3) is considered as **veto**. The winning option is the one with the weakest rejection, among those without veto. In case of ambiguity stronger approval rate is relevant. If ambiguity persists, the poll is repeated with just the ambiguous options. If there is no option without veto, then

Conclusion & Outlook



Summary:

- Reproducibility of computational results is still an unsolved challenge
- Existing approaches (public code, CI) not widely enough used

- Proposal: Combination of existing components
  - Structured repository
  - Tailored webservice
  - Peer-based organization structure

Outlook:

- Improve `ackrep_core` (containerization), add more content to `ackrep_data`

Conclusion & Outlook

Summary:

- Reproducibility of computational results is still an unsolved challenge
- Existing approaches (public code, CI) not widely enough used

- Proposal: Combination of existing components
  - Structured repository
  - Tailored webservice
  - Peer-based organization structure

Outlook:

- Improve `ackrep_core` (containerization), add more content to `ackrep_data`

- Incorporate feedback from potential users/contributors

→ https://testing.ackrep.org repository: https://github.com/cknoll/ackrep_data

→ Corresponding author: carsten.knoll@tu-dresden.de