ISSN 2186-7437

NII Shonan Meeting Report

No. 190

Engineering Dependable Ubiquitous Systems

Christos Tsigkanos
Carlo Ghezzi
Zhenjiang Hu

March 17 - 20, 2025

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Contents

1 Background – Introduction

Today, even small devices – from mobile phones to industrial robots – are internet-connected, are capable of running software and are situated close to end-users or applications, rendering the overall systems induced software-intensive. Commonly understood by evocative terms such as Industry 4.0 or as said to be exhibiting smart functionalities, they are becoming ubiquitous and are increasingly integrated in daily life and are increasingly engineered with autonomy in mind.

The major hurdle to engineering such systems is the inherent complexity of their software, as software is the principal factor defining their overall behavior. This complexity manifests both at design as well as at runtime. At design time, software may be required to be designed and developed for heterogeneous platforms and stacks while ensuring interoperability. Software components may furthermore implement particular types of computation (such as verification or AI functionalities), often in a distributed manner. This is something which needs to be accommodated in both design and runtime cycles – a typical example case is Internet of Things applications, where technological advances have enabled even miniscule devices to run software; as such, a continuum from devices to cloud arises. At runtime, such systems are exposed to changes in different contexts derived from changes in environment, software configuration, execution infrastructure, or other unforeseen issues.

Paradigms and approaches from software engineering and programming languages have shown to be highly effective for such designs – however traditional methods and techniques face challenges from this new domain. Examples range from model-based engineering and code generation to tackle heterogeneity, to verification techniques employed at runtime for requirements validation or AI used to devise control actions. All those require the backing of sound engineering techniques in order to build and operate scalable, usable and efficient applications and systems that operate in a way that is beneficial to individuals, society and the environment. This Shonan meeting aimed at investigating wide themes within software engineering for dependable ubiquitous systems, including:

- *Software-defined Everything.* Software is used to abstract and automate management and control of computational, networking, storage, or hardware resources of various devices. Applications involve various software stacks and exhibit different complexities, with multiple software components being deployed in diverse infrastructures and contexts being a central theme: components may be deployed on (or migrated to) different hosts, ranging from resource-constrained or domain-specific devices to powerful cloud servers. Architecting and developing such systems reliably is a major challenge. Software-defined everything here involves active management of the system by (possibly autonomous) software agents, which set in motion actions to satisfy system objectives. Attributes such as locality, variability in the environment, and distribution of computation (manifested as uncertainty at runtime) pose challenges, calling for additional management and control closer to the software components' operating architectural layer. A further open problem is how to instrument requirements verification and control at runtime for systems that are diverse, are expected to scale (e.g., to avoid resource saturation or central points of failure), ensure con-

formance to new requirements (e.g., privacy, responsiveness), or are highly dynamic.

- *Programming Models for Ubiquitous Computation.* One critical problem in exploring ubiquitous and pervasive systems for greater societal benefits is the lack of a fundamental basis of widely accepted programming models for such systems. The ubiquitous computing scenario brings many new problems such as coping with the limited processing power of mobile devices, frequent disconnections, interoperation among different computing devices, the migration of code and tasks between heterogeneous devices, etc. It requires programming languages and models that can support mobility, interoperability, adaptation, and context awareness.

- *Ethics for Autonomous and Intelligent Systems.* As deployment of autonomous and intelligent systems becomes increasingly pervasive, we need to establish societal and policy guidelines in order for such systems to remain human-centric and ensure that they serve humanity's values and foundational ethical principles. The self-adaptive systems community has tackled mechanisms for attainment of technical goals; autonomous and intelligent systems however raise issues beyond simply addressing technical problems. These systems must be developed and should operate in a way that is beneficial to society and the environment.

2 Overview of the Meeting

The NII Shonan Meeting "Engineering Dependable Ubiquitous Systems" took place in Shonan, Japan from March 17th to 20th, 2025. The seminar was organized by Christos Tsigkanos (University of Athens & University of Bern), Carlo Ghezzi (Politecnico di Milano) and Zhenjiang Hu (Peking University). We hosted 26 participants from all over the world and diverse expertise, who discussed various topics. All participants held lightning talks, while 10 experts gave longer overview talks. All talks were followed by a discussion session, while we allotted sufficient time for working group discussions throughout the days of the seminar. The social event included the traditional seminar outing, including visits of the Kenchoji temple and Kamakura and attending a ZAZEN session.

3 Overview of Talks

Runtime Monitoring of Ubiquitous Systems

Michele Loreti – University of Camerino

Ubiquitous Systems (US) are a class of systems where computing capabilities are woven into the fabric of everyday life. These are composed by a set of entities that, while interacting with the users, are able to perceive the state of the environment where they operate and that can use actuators to change it. Each of these entities are often programmed to reach local or global goals and operate without any centralised control and to adapt their behaviour according to the changes in the environment where they operate. To support development of resilient US, it is crucial to introduce tools and methodologies that permit describing their behaviour and specifying and verifying the expected requirements. For this reason, if is of outmost importance to identify the right specification language (or primitives) to describe the behaviour of each entity and to use formal languages (temporal logics) that permits describing requirements of a single entity as well as of the full system. In this talk we have first discussed how computation of US can be modelled. Moreover, we have introduced the logic GLoTL that can be used to specify properties of a collection of agents at both local and global levels. At a local level, the properties of the behaviour of single entities are considered. While at the global level, properties refer to the whole system. Finally, the algorithm to support runtime verification of GLoTL is discussed.

An Overview on Reactive Synthesis and Specifications

Shahar Maoz – Tel Aviv University, Israel

Reactive synthesis is an automated procedure to obtain a correct-by-construction reactive system from a given temporal specification. Examples of these systems include the software controllers of robotic systems. Despite recent advancements on the theory and algorithms of reactive synthesis, e.g., efficient synthesis for the GR(1) fragment of linear temporal logic, many challenges remain in bringing reactive synthesis technologies to the hands of software engineers. Over the last several years, we have worked on bridging this gap, addressing challenges that relate to the change from writing code to writing specifications, and the development of tools to support a specification-centric rather than a code-centric development process.

In this talk I'll start with an overview of the SYNTECH project and the Spectra specification language and synthesizer. I will then focus on several recent works dealing with unnecessary assumptions, the specification of triggers, and a user study on the use of the synthesizer. Finally, I will discuss why we should pay more research attention to formal specifications, what are the challenges we will be facing, and present a vision for future work on specification engineering.

Engineering Digital Systems for Humanity: A Research Roadmap

Martina De Sanctis – Gran Sasso Science Institute (GSSI), Italy

As testified by new regulations like the European AI Act, worries about the human and societal impact of (autonomous) software technologies are becoming of public concern. Human, societal, and environmental values, alongside traditional software quality, are increasingly recognized as essential for sustainability and long-term well-being. Traditionally, systems are engineered taking into account business goals and technology drivers. Considering the growing awareness in the community, in this talk, I present a research roadmap for engineering Digital Systems for humanity, where we argue that the engineering of systems should also consider human, societal, and environmental drivers. To this aim, we identified the macro and technological challenges by focusing on humans and their role while co-existing with digital systems. The first challenge considers humans in a proactive role when interacting with digital systems, i.e., taking initiative in making things happen instead of reacting to events. The second concerns humans having a reactive role in interacting with digital systems, i.e., humans interacting with digital systems as a reaction to events. The third challenge focuses on humans with a passive role, i.e., they experience, enjoy, or even suffer digital systems' decisions and/or actions. The fourth challenge concerns the duality of trust and trustworthiness, with humans playing any role. Building on the new human, societal, and environmental drivers and the macro and technological challenges, we identified a research roadmap of digital systems for humanity. The research roadmap is concretized in a number of research directions organized into four groups: development process, requirements engineering, software architecture and design, and verification and validation.

Formal Verification, Vulnerability Analysis, and Attack Detection in Cyber-Physical Systems

Marjan Sirjani – Mälardalen University, Sweden

To explain formal verification, vulnerability analysis, and attack detection in cyber-physical systems I focus on a concrete example, Distributed Redundant Controllers. A potential problem that may arise in the domain of distributed control systems is the existence of more than one primary controller in redundancy plans which may lead to inconsistency. We worked on an algorithm called NRP FD (Network Reference Point Failure Detection), proposed by industry, to solve this issue by prioritizing consistency over availability. I explain how by using modeling and formal verification, we discovered an issue in NRP FD where we may have two primary controllers at the same time. We then provide a solution to mitigate the identified issue, thereby enhancing the robustness and reliability of such systems. In the same context, I also show how we used model checking for making informed decisions in designing test cases for fault-tolerant systems, and how our timing analysis helps in making timing configuration decisions. I show how Time Rebeca and Lingua Franca languages and tools are used for this work hand in hand. I add a discussion on how this approach may be generalized in different contexts.

Resource provisioning with emphasis on ML

Luciano Baresi – Politecnico di Milano, Italy

Many modern dependable systems exploit virtual execution environments (e.g., virtual machines or containers) and embed some machine-learning (ML) based components. Oftentimes, the dependability of these systems does not only require the correctness and robustness of their components but calls for a proper management of computational resources: too few resources may hamper the proper operation of these systems, while too many resources would be a waste and would impact their sustainability.

Besides framing the problem, the talk discussed the work carried out in the last years to tackle these issues. The presentation started by sketching the key ingredients of the first solutions we developed to control CPU core allocation for the operation of interactive web applications. The use of containers and control theory (PI controllers) allowed for the fine-grained and runtime adaptation of provisioned cores to meet set response times. We used similar enablers to control the operations of batch applications with proper deadlines, that is, big-data applications run on top of Spark. These applications are not interactive and there no response times to meet; deadlines are used to constrain batch execution. CPUs are not enough when one wants to control the training of ML models, which impose that GPU cores be allocated, and their peculiar characteristics be managed properly. This is why designed controller provision resources in a different way.

Thorough evaluation demonstrates that developed solutions work properly and can scale to manage realistic applications. Fine-grained GPU provisioning and distributed settings, for example, federated learning frameworks, need further work and are still part of our research agenda.

Towards Neuro-Symbolic Formal Reasoning

Xiaoxing Ma – Nanjing University, China

This talk proposes Neuro-Symbolic (NeSy) formal reasoning as a promising approach for integrating the power of AI, particularly Large Language Models (LLMs), into software engineering.

Software engineering has achieved tremendous success in building large-scale systems, making "software-defined everything" nearly a reality. The power of software lies in its unique capability to manage complexity through unprecedented compositionality and reusability. These strengths stem from the fact that traditional (symbolic) software is a purely logical artifact, free of uncertainty.

However, the introduction of ML models brings uncertainty due to their statistical nature, reliance on incomplete induction, and the randomness involved in training processes. This inherent uncertainty challenges foundational software engineering principles. Simply using ML models as components or as unverified code generators undermines software's traditional strengths. While powerful, LLMs often produce incorrect or "hallucinated" outputs.

The core proposal is a NeSy approach that leverages the strengths of both paradigms: the LLM's ability to "guess" or generate potential solutions (like

code or mathematical proofs) and symbolic methods' ability to rigorously "check" or verify the correctness of these generations. This "Guess and Check" method aims to produce trustworthy outputs from potentially untrustworthy LLMs. The idea is to automate the "how" (implementation and proof generation) while humans focus on the "what" (requirements and goals), ensuring LLM-generated artifacts are formally verified by symbolic tools.

The talk also presents initial work exploring the feasibility of this NeSy approach, yielding encouraging results:

The Olympiad Inequality Proving (LIPS) system combines LLM intuition for suggesting proof steps with symbolic solvers for verifying and pruning these steps. This method significantly outperformed not only LLMs or symbolic solvers alone on various datasets but also human Gold medalists on a benchmark of competition-level inequalities. It even discovered novel proofs and identified errors in existing expert solutions.

The Loop Invariant Inference (LaM4Inv) framework uses LLMs to generate candidate loop invariants. These candidates are then verified and refined using symbolic tools like SMT solvers and Bounded Model Checking, incorporating counterexamples back into the LLM prompt for iterative improvement. This NeSy approach demonstrated significant improvements over existing baseline methods for loop invariant generation.

Environment Modeling-driven Software Requirements Engineering

Zhi Jin – Peking University, China

Requirements engineering is to observe the real world, locate system problems, and decide the system capabilities. There are different perspectives on the problem observation. Ubiquitous Systems directly interact with their operative environment i.e. sensing the environment and actuating the environment. It is more than important to identify the features and issues of the environment and to deal with the concerns initiated by the environment. The environmental model-based approach is taking the environment as the first-class citizen when conducting the requirement engineering. Through modeling the environment, and analyzing the environment static and dynamic characteristics, the behaviors that the systems should have and the constraints that need to be met can be derived systematically. This talk first presents what the environment is, how to model the environment, and how to determine the system dependability enhancement capabilities from the perspective of the environment. Second, it presents a study to identify the preferred way of expressing end users' requirements in a typical kind of ubiquitous systems, the smart home. We categorize the needs of smart home into three levels of abstraction and propose a multi-level requirements description language which can illustrate the progressive refinement of smart home requirements. Third, a set of problem decoupling strategies is proposed to separate system problem concerns and transform complex problems into a set of relatively simpler problems which can be specified by a set of specification patterns. Finally, the talk is trying to explore the possibility of generating the code for the specification patterns.

Engineering DSLs for Constructing Dependable Ubiquitous Systems

Zhenjiang Hu – Peking University, China

Safe and user-friendly domain-specific languages (DSLs) are gaining traction in ubiquitous computing for their ability to specify domain-specific computation directly, enabling non-programmer experts to write code using familiar terminology and operations. However, crafting DSLs tailored to these experts remains challenging. Traditional embedded DSLs offer rapid prototyping but require intimate knowledge of the host language, creating barriers for domain users unfamiliar with their syntax or semantics. In this talk, we introduces a novel approach to streamline DSL development and corresponding IDE construction. It consists of three key components: (1) an extensible general-purpose core language, (2) a DSL definition framework leveraging syntactic sugars to simplify domain abstraction, and (3) a language-lifting process that automatically generates DSL implementations alongside an IDE optimized for DSL workflows. This approach decouples DSLs from host-language intricacies, allowing domain experts to focus on problem-solving rather than technical implementation. To validate this method, we developed Osazone, a system used to create multiple DSLs across diverse domains (e.g., healthcare, finance). The results demonstrate flexibility in adapting to domain-specific needs, efficacy in reducing development time, and practicality in empowering domain experts to independently design and deploy DSLs.

Why Ethical Thinking must Inspire Development of Autonomous Systems

Carlo Ghezzi – Politecnico di Milano

The world in which we live relies on digital technologies, and in particular on software, which operates and interacts with the physical world and humans. In the digital era, software engineers are the demiurges who are creating a new cyber-physical world, where humans, autonomous agents powered by AI, and physical entities live together in a new kind of society. Already in the late 1990's constitutionalist L. Lessig said that software defines the laws that govern the world and asked for reflection and action, because of the potential disruptive consequences. This is even more urgent today, due to to the phenomenal progress of AI and AI-generated software, which led to an increasing pervasiveness of software-enabled functions, with more and more intimate relation with humans and society. This raises the urgent need for re-thinking the way we do research, the competences and responsibilities of technologists who conceive and develop software, and the skills they should acquire through education. Rethinking should start by asking questions like: Should software engineers care about the human values involved while conceiving/developing new applications? About possible future uses and ethical implications? Can they do it by themselves? What kind of skills would they need?

The talk mainly aims at setting the stage for opening a much needed and urgent discussion, which should involve software researchers and educators and has to be broad and open, especially to social sciences and humanities.

Formal Modelling of Socially-aware Autonomous Systems

Livia Lestingi – Politecnico di Milano

Autonomous systems are increasingly deployed in complex contexts where interaction with humans is required. When such contexts are also safety-critical, as is the case with emergencies, guaranteeing safety is paramount. This talk presents FormIDEAble, a framework for the formal modelling of socially-aware autonomous systems and the automated synthesis of safe cooperation strategies with humans. The formal model, a Priced Timed Markov Decision Process (PTMDP) network, incorporates social identity as a source of uncertainty when synthesising a cooperation strategy at runtime. While existing work uses social identity theory and game theory to engineer socially-aware autonomous systems software, it does not provide safety guarantees. The FormIDEAble framework uses Uppaal Stratego, which is a verification tool for PTMDPs, to synthesise strategies optimising performance metrics while guaranteeing safety properties (such as actions within a given time bound). Experimental results show that FormIDEAble balances optimisation and safety guarantees and that improves the autonomous system's performance by 24% on average. Moreover, FormIDEAble provides those safety guarantees while introducing minimal overhead with an average running time of under 2.5s.

Failure Analysis in Cyber-Physical Systems

Ezio Bartocci – TU Wien

Cyber-physical systems (CPS) combine continuous physics elements with discrete control logic, making the diagnosis of faults very complex: physical anomalies may masquerade as logical failures, propagate across components and time scales, and evade detection until they breach high-level safety requirements. In this talk I surveyed four complementary research efforts we have developed as a toolbox for systematic failure analysis in Simulink/Stateflow models.

CPSDebug [1] combines specification mining, differential testing and trace-based reasoning to automatically construct human-readable explanations of observed failures. By separating continuous from discrete variables during invariants generations, CPSDebug localises the root cause of a violation and visualises its temporal–spatial propagation.

FIM (Fault Injection & Mutation) [2] provides an open-source infrastructure for injecting transient or persistent faults at arbitrary model locations. Fine-grained activation flags and parameterised mutation operators enable controlled experimentation and large-scale robustness studies without manual model editing.

To precisely localise the components that trigger a failure, we combine [3] formal requirements with search-based test generation. Given a failing test and its requirement, our method synthesises a closely related passing test; contrasting the two executions isolates the minimal set of suspicious blocks and variables—even under multiple simultaneous faults.

Finally, we extend classical mutation testing with a property-based perspective [4]. A mutant is considered relevant only if it can influence the satisfaction of a given requirement, and it is meaningfully killed only when that requirement

is violated. This approach provides metrics that are both more informative for safety assurance and better aligned with requirement-driven test generation.

Together, these techniques move CPS failure analysis from ad-hoc, manual debugging toward an automated, property-aware workflow that spans fault injection, detection, explanation and quantitative evaluation. The case studies in automotive and avionics domains demonstrate significant reductions in debugging effort and sharper insights into system resilience.

[1] Ezio Bartocci, Niveditha Manjunath, Leonardo Mariani, Cristinel Mateis, Dejan Nickovic: CPSDebug: Automatic failure explanation in CPS models. Int. J. Softw. Tools Technol. Transf. 23(5): 783-796 (2021)

[2] Ezio Bartocci, Leonardo Mariani, Dejan Nickovic, Drishti Yadav: FIM: fault injection and mutation for Simulink. ESEC/SIGSOFT FSE 2022: 1716-1720

[3] Ezio Bartocci, Leonardo Mariani, Dejan Nickovic, Drishti Yadav: Search-based Testing for Accurate Fault Localization in CPS. ISSRE 2022: 145-156

[4] Ezio Bartocci, Leonardo Mariani, Dejan Nickovic, Drishti Yadav: Property-Based Mutation Testing. ICST 2023: 222-233

4 List of Participants

Name	Affiliation	Country
Prof. Christos Tsigkanos	University of Bern & University of Athens	Switzerland
Prof. Carlo Ghezzi	Politecnico di Milano	Italy
Prof. Zhenjiang Hu	Peking University	Japan/China
Prof. Romina Spalazzese	Malmö University	Sweden
Prof. Fuyuki Ishikawa	National Institute of Informatics	Japan
Prof. Luciano Baresi	Politecnico di Milano	Italy
Prof. Marjian Sirjani	Mälardalen University	Sweden
Prof. Xiaoxing Ma	Nanjing University	China
Prof. Soichiro Hidaka	Hosei Univeristy	Japan
Prof. Carla Ferreira	NOVA University Lisbon	Portugal
Prof. Javier Camara	Universidad de Malaga / University of York	Spain
Prof. Ezio Bartocci	Technische Universität Wien	Austria
Prof. Kenji Tei	Institute of Science Tokyo	Japan
Prof. Katinka Wolter	Free University Berlin	Germany
Prof. Shahar Maoz	Tel Aviv University	Israel
Dr. Martina De Sanctis	Gran Sasso Science Institute (GSSI)	Italy
Prof. Yao Guo	Peking University	China
Prof. Zhi Jin	Peking University	China
Prof. Bernhard Rumpe	RWTH Aachen University	Germany
Prof. Michele Loreti	University of Camerino	Italy
Prof. Elisa Gonzalez Boix	Vrije Universiteit Brussel	Belgium
Dr. Livia Lestingi	POLITECNICO DI MILANO	Italy
Dr. Ilias Gerostathopoulos	VU Amsterdam	The Netherlands
Prof. Hiroyuki Nakagawa	Osaka University	Japan
Dr. Hiroyuki Kato	NII	Japan

5 Meeting Schedule

Engineering Dependable Ubiquitous Systems

March 17 - 20, 2025 (Check-in: March 16, 2025)

Monday

Time	Session
09:30-10:00	Welcome
10:00-10:45	Introductions
Break	
11:15-12:00	Introductions (cont.)
Lunch	
14:00-15:30	Introductions (cont.)
Break	
16:00-17:30	Ubiquitous and CPS Systems (I)

- **Shahar Maoz** – An Overview on Reactive Synthesis and Specifications
- **Marjan Sirjani** – Formal Verification, Vulnerability Analysis, and Attack Detection in Cyber-Physical Systems
- Discussion

Dinner

Tuesday

Time	Session
9:00-10:30	Programming Models for Ubiquitous Computation

- **Michele Loreti** – Runtime Monitor of Ubiquitous Systems
- **Xiaoxing Ma** – Towards Neuro-Symbolic Formal Reasoning
- Discussion

Break	
10:45-12:30	Software-defined Everything

- **Ezio Bartocci** – Failure Analysis of Cyber-Physical Systems
- **Luciano Baresi** – Resource provisioning with emphasis on ML
- Discussion

Time	Session
Lunch	
13:30-15:30	Ethics for Autonomous and Intelligent Systems
• **Zhi Jin** – Environment Modeling-driven Software Requirements Engineering • **Zhenjiang Hu** – Engineering DSLs for Constructing Dependable Ubiquitous Systems • **Carlo Ghezzi** – Why Ethical Thinking must Inspire Development of Autonomous Systems • Discussion	
Break	
16:00-16:30	**Panel: Bootstrapping Questions**
• **Programming for Ubiquitous Computing:** What programming models and language features are needed to effectively support the development and deployment of dependable ubiquitous systems across diverse platforms and environments? • **Verification and Validation:** How can we effectively model, specify, and verify properties of ubiquitous systems, particularly at runtime, to ensure their dependability and resilience? • **Rethinking Software Engineering:** How can we adapt current principles and practices to better address ethical considerations and human values in the context of dependable ubiquitous systems? • **Engineering Ethical Autonomous Systems:** What are the challenges in designing and building trustworthy AI systems, and how can we ensure they remain human-centric and beneficial to society? • **AI and Machine Learning:** What are the key considerations for developing dependable and self-adaptive ML-based systems, and how can we address the challenges posed by AI-enabled cyber-physical systems?	
16:30-17:30	Discussion and Group Forming
Dinner	

Wednesday

Time	Session
9:00-10:30	Ubiquitous and CPS Systems (II)
• **Livia Lestingi** – Formal Modelling of Socially-aware Autonomous Systems • Discussion	
Break	
10:45-11:30	Group Breakout
11:30-12:00	Intermediary Group Presentations and Feedback
Lunch	
14:00 Onwards	Excursion/Social Event: Visiting Kenchoji Temple with "ZAZEN".

Thursday

Time	Session
9:00-10:00	Group Breakout
Break	
10:30-12:00	Final Group Presentations
• Discussion and Action Plan	
Lunch	

6 Summary of Discussions

The meeting convened experts from the domains of software engineering, programming models, and distributed systems, specifically within the context of ubiquitous computing. The workshop's structure comprised three distinct phases:

- An initial plenary session wherein each participant provided a concise self-introduction, followed by a presentation of their current research and salient research themes. This served as a foundational step towards establishing a shared understanding of the diverse expertise present.
- A subsequent phase involving focused presentations delivered by selected attendees. These presentations were strategically curated by the organizers to serve as catalysts for in-depth discussions on the pre-identified thematic areas. The structuring of these presentations around specific topics was intended to foster the formation of discussion groups.
- The concluding phase consisted of the formation of break-out groups tasked with the interactive exploration and elaboration of specific topics. While the precise topics for these groups were intended to emerge organically during the meeting itself, the organizers prepared a preliminary set of themes to bootstrap initial discourse. These proposed themes critically engaged with the following complex challenges:
  - Advancements in programming language paradigms to effectively address the landscape of software-defined everything.
  - The necessary language features and underlying models for robustly supporting the technical specifications and configuration parameters of diverse execution platforms, and the mechanisms through which seamless deployment and provisioning can be realized across geographically distributed and heterogeneous infrastructure.
  - Methodologies for instrumenting sophisticated control, coordination, and self-healing mechanisms within the software fabric of ubiquitous systems, with a specific emphasis on the integration of regulatory frameworks to ensure ethically-aligned behaviors within such complex distributed environments.
  - Strategies for effectively managing and abstracting core business logic from intricacies of underlying infrastructure capabilities, thereby promoting portability and resilience.
  - Whether non-traditional computational paradigms, such as AI workflows or formal verification processes, necessitate the development of novel programming and system-level models.
  - The identification and rigorous analysis of critical factors and runtime attributes that present significant challenges to the engineering of ubiquitous systems, including, but not limited to, inherent data locality constraints, dynamic environmental variability, and the intrinsic distribution of computational processes.
  - The development of methodologies for the continuous monitoring of runtime aspects and the detection of both anticipated and emer-

gent system behaviors that could potentially compromise dependability, and critically, adherence to established ethical values. Furthermore, the exploration of proactive counteractions through self-adaptive mechanisms was considered paramount.

- The optimal form in which programming support for ubiquitous software systems should be manifested, encompassing conceptual frameworks, specialized middleware solutions, or entirely novel programming model abstractions.
- A critical examination of how ethical considerations can be fundamentally embedded within the engineering lifecycle of autonomous technologies, particularly within the nuanced context of self-adaptive systems.

Discussions were documented to facilitate the subsequent distillation of key workshop findings, the precise articulation of remaining open research problems, and the formulation of proposals for future research directions and collaborative initiatives.

6.1 Programming Languages and Abstraction

The discussion group centered on the challenges and potential solutions for programming dependable ubiquitous computing systems from the programming languages perspective. Participants explored desirable programming language models including event-based, ambient-oriented, data-driven, and context-aware programming, as well as the use of Domain-Specific Languages (DSLs). Key challenges identified included interoperability, privacy, security, safety, the need for domain-expert-friendly abstractions, scalability, and the development of effective debugging tools and programming environments. The group discussed plans to address these issues through a position paper, potential collaborations, and device demonstrations.

6.2 Specification and Verification

At the specification and verification group, the discussion revolved around the definition of a problem exemplar that can be used as a vehicle to drive research in the area. Self-adaptive ubiquitous computing systems are increasingly deployed in critical domains that require run time adaptation to changes in dynamic environments affected by multiple sources of uncertainty. One such domain is assistive care, where there is a tight interaction between software, physical elements, and human participants. Unlike other types of self-adaptive systems, self-adaptive ubiquitous systems must continuously balance an array of concerns that include competing technical, ethical, and human-centric requirements.

6.3 Ethics for Autonomous and Intelligent Systems

This group focused on the crucial goal of raising awareness about ethics in the realm of software engineering. The discussion prioritized key concepts, setting aside process and education as secondary concerns for this initial phase. The overarching aims involve not only training and informing technical individuals about ethical considerations but also extending this awareness to professionals

in other disciplines, such as law and human sciences. A significant aspect of this outreach includes addressing legal compliance, exemplified by the contrasting regulations surrounding robocalls in the EU and the US.

Furthermore, the group considered establishing channels with authorities and lawmakers about the technological possibilities and limitations as necessary, particularly concerning enforcement challenges. The discussion also highlighted the need to re-evaluate existing regulations, noting that the EU AI Act, for instance, was formulated before the advent of transformer models and may require updates to account for broader technological impacts. Ultimately, the group sought to encourage a fundamental rethinking of the decision-making processes employed in the development and deployment of software systems in line with ethical guidelines.

Transparency emerged as a central theme, with an emphasis on "accessible transparency" as a guiding principle for designing ethical systems. This entails ensuring that the intents behind a system are clear and explicit, and that the target users are unambiguously defined. Accessibility was recognized as going beyond mere usability, requiring explicit consideration of inclusivity/exclusivity in design choices, acknowledging the potential gap between educated/uneducated users. It is crucial to clearly communicate why and for what purposes a user can reliably depend on a system, and to explicitly identify all relevant stakeholders, including end-users and policymakers. The concept of explainable systems was also discussed, stressing that explanations should be tailored to the specific claims being made by a software system and the intended audience of the system. Moreover, the sustainability implications of systems, encompassing not just financial costs but also time, energy consumption, and carbon dioxide emissions, should be made explicit. The trustworthiness of a system hinges on whether the guarantees it provides are proven, even if probabilistically, acknowledging the often uncertain environments in which these systems operate. Finally, the discussion addressed the growing challenges in digital identity verification due to technologies like AI deepfakes, raising the fundamental question of distinguishing between human and artificial interaction.

The area of *specification and verification* highlighted the complexities arising from differing societal conceptions of ethics (e.g., as illustrated by the trolley problem). The limitations of standard temporal logics for capturing nuanced ethical concepts were noted, suggesting the potential of deontic-like logics in this domain. The group advocated for systems that offer various levels of personalization to accommodate individual values. While acknowledging the existence of global ethical principles, the discussion also recognized the necessity of exceptions, and the inherent difficulties in managing these exceptions within logical frameworks and practical rules. Defining abstract notions like fairness and bias and translating them into precise specifications remains a significant challenge, although ongoing research, including approaches using hyperproperties and metamorphic testing, offers potential avenues. The specific impact of bias in transfer learning was also raised as a critical concern. Ultimately, the view presented by the system must be trusted, trustworthy, provable, and readily explainable to all stakeholders involved.

The interaction between *users and autonomous systems* was also considered as relevant for defining ethical interactions, particularly in the context of robotics. Finally, the discussion acknowledged the continuous evolution of both software systems and the broader ethical landscape, necessitating ongoing

attention and adaptation of ethical considerations.

7 Identified Issues and Future Directions

7.1 Programming Languages and Abstraction

The discussion group participants identified several key challenges in the development of dependable ubiquitous computing (DUC) systems. A primary concern is interoperability, ensuring that diverse components and systems can effectively communicate and collaborate. The need to address privacy, security, and safety is also paramount, alongside enabling autonomous operation of these systems. Furthermore, there's a recognized gap between current tools and the needs of domain experts; programming languages and tools should be friendly to domain experts, allowing them to contribute to development. Effective abstraction is crucial to manage the complexity inherent in DUC systems. Scalability presents another significant hurdle, as these systems must handle increasing amounts of data and devices. The group also highlighted the importance of establishing robust methodologies for developing DUC systems and the difficulties associated with debugging distributed and pervasive systems. Finally, the limitations of current programming environments (IDEs) were noted as an impediment to efficient development.

To address these identified issues, the discussion group outlined several future directions. The first is to draft a position paper that articulates a framework encompassing WASM, OS, DSLs, APIs, event-driven and data-driven programming paradigms, and bidirectional synchronization techniques such as BX, CRDTs, and OT. A key strategy involves fostering collaborations among programming language experts, operating system teams, and the IoT industry to leverage diverse expertise. The group also emphasized the importance of practical demonstrations, with plans to showcase solutions on devices like the M5StickC to provide hands-on evidence of their feasibility and effectiveness.

The discussion group identified several promising programming language models for dependable ubiquitous computing (DUC) systems and also highlighted key challenges in the field. Good programming models for DUC include event-based programming languages, ambient-oriented programming, middleware/APIs, data-driven programming models, context-aware programming languages, and Domain-Specific Languages (DSLs). However, significant challenges remain, such as ensuring interoperability among diverse systems, addressing privacy, security, and safety concerns, creating tools that are user-friendly for domain experts, developing effective abstractions, achieving scalability, establishing robust methodologies, and improving debugging and programming environments.

7.2 Specification and Verification

The specification and verification group centered its discussion on defining a problem exemplar to guide their research efforts and identify challenges. This research focuses on self-adaptive ubiquitous computing systems, which are increasingly used in critical areas demanding real-time adaptation to dynamic and uncertain environments. Assistive care serves as a key domain, highlighting the complex interplay between software, physical components, and human users, requiring a continuous balance of technical, ethical, and human-centric considerations unlike other adaptive systems. Some of the distinctive challenges in the

area include:

- Multiple modalities of human involvement. A fundamental challenge in such systems stems from the tight involvement of human participants in multiple roles. Participants are not only passive recipients of assistance but can also actively participate as monitors, analysts, and decision-makers. They contribute to run time validation processes through continual feedback (Human-on-the-Loop), and may be required to take actions that are difficult for automated systems to perform (Human-in-the-Loop). This deep integration of human and automated system capabilities introduces complexities in decision-making, as the system must accommodate varying levels of human engagement and trust while ensuring that adaptation strategies remain effective.

- *Tradeoffs among technical and social goals and constraints.* Furthermore, the operation of these systems is governed by a delicate balance between technical system goals and broader socio-technical constraints. While system designers typically prioritize goals such as safety, availability, cost efficiency, and timeliness, real-world deployments must also adhere to ethical principles, legal regulations, and compliance standards—referred to as SLEEC (Social, Legal, Ethical, Environmental, and Cultural) considerations. The interplay between these objectives is potentially conflicting, as optimizing one dimension may come at the expense of another. For example, ensuring strict adherence to safety properties (e.g., avoiding collisions of an assistive mobile robot that is sharing the physical space with a human user) might reduce system availability or increase operational costs, requiring trade-offs that must be carefully managed at run time.

- *Local vs. Global Goals.* An additional layer of complexity arises from the potential misalignment between local and global goals. In assistive care settings, system-wide optimization strategies (e.g., minimizing operational costs, maximizing coverage of users receiving service) may conflict with preferences or constraints of individual human users (e.g., maximizing quality of the received service). Unlike other fully automated systems where all components operate in unison, ubiquitous self-adaptive systems must accommodate the autonomy of human participants. Users may choose to disregard system recommendations, or prioritize personal preferences over the system's prescribed actions. These variations in behavior introduce unpredictability and make it challenging to enforce system-wide adaptation strategies effectively.

Given these complexities, the group identified the need for exemplar problems that encapsulate these challenges and provides a foundation for evaluating self-adaptive ubiquitous computing systems. The discussion at the group bootstrapped such a problem, highlighting the critical interdependencies between human actors, technical constraints, and ethical considerations.

7.3 Ethics for Autonomous and Intelligent Systems

The initial discussion highlighted several critical points for fostering ethical considerations in technology. It emphasized the need for strong technical awareness

among developers to build ethically sound systems, alongside robust interdisciplinary engagement to bridge the gap with fields like law and human sciences. The group also stressed the importance of informing legal and regulatory bodies about the nuances of technology, especially regarding enforcement challenges and the timeliness of existing laws. Finally, a fundamental rethinking of decision-making processes was deemed necessary in the age of increasingly autonomous systems.

Based on these discussions, several next steps and recommendations were proposed to advance the goal of raising ethical awareness. These include developing accessible educational materials for diverse audiences, fostering interdisciplinary dialogue through workshops and forums, investigating formal methods for ethical specification, conducting research on bias in machine learning, engaging with regulatory bodies to inform policy evolution, and promoting best practices for achieving transparency in system design and documentation. This report lays the groundwork for future efforts aimed at integrating ethical considerations as a fundamental aspect of software engineering practices and cultivating a wider societal understanding of the ethical dimensions of technology.