Strategic Research & Innovation Agenda

FOR THE FUTURE OF SOFTWARE

The foundation of Europe's digital transformation

September 2026

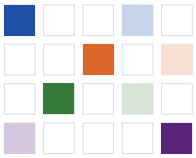
ERCIM
European Research Consortium for Informatics and Mathematics

INFORMATICS EUROPE

Software is the foundation on which Europe's digital future depends.

About this Strategic Research and Innovation Agenda

Developed by the European software community under the auspices of ERCIM and Informatics Europe.



PUBLICATION INFORMATION

Title	Strategic research and innovation agenda (SRIA) for the future of software
Published	23rd of September 2026
DOI	10.5281/zenodo.22685531
Coordinated by	Informatics Europe Software Research Working Group www.informatics-europe.org/research/software-research.html chair: Prof. Dr. Tanja E. J. Vos (tvos@vrain.upv.es) ERCIM Software Working Group ercim-software-wg.github.io/ chair: Prof. Dr. Tijs van der Storm (storm@cwi.nl)
Copyright	Informatics Europe and ERCIM, licensed under CC BY-SA 4.0.
Core writing team	Benoit Combemale, Ina Schieferdecker, Alexander Serebrenik, Tijs van der Storm and Tanja E. J. Vos.
Contributions from	Silvia Abrahão, Emil Alégroth, Bengt Augustsson, Paris Avgeriou, Luciano Baresi, Mikkel Baun Kjaergaard, Domenico Bianculli, Jan Bosch, Lionel Briand, Jean-Michel Bruel, Lola Burgueño, Manuel Carro, Michel Chaudron, Roberto Di Cosmo, Elisabetta Di Nitto, Yvonne Dittrich, Yanja Dajsuren, Sigrid Eldh, Martin Glinz, Jesus M. Gonzalez-Barahona, Alessandra Gorla, Regina Hebig, Timo Kehrer, Sangeeth Kochanthara, Filippo Lanubile, Elena María Navarro, Kim Mens, Tom Mens, Andreas Metzger, Jürgen Mottok, Juan Manuel Murillo Rodríguez, Mikael Ödqvist, Gilles Perrouin, Ernesto Pimentel, Azzurra Ragone, Malin Rosqvist, Bernhard Rumpe, Mikael Sjödin, Kurt Schneider, Ernest Teniente, Jurgen Vinju, Anthony Ventresque, Stefan Wagner, Anton Wijs, Stefano Zacchiroli, Andreas Zeller, Steffen Zschaler

This SRIA is a community-driven strategic agenda intended to support discussion on future European research, innovation, infrastructure, education and policy priorities in software engineering and technologies.

Executive summary

The criticality of software and the urgency to act

PURPOSE

Secure multi-year EU investment in foundational software research and shared infrastructure to ensure the success of Europe's digital transformation.

AUDIENCE

European Commission (DGs CNECT, RTD, GROW, DIGIT), Member States, Industry & Software foundations and associations.

01 SOFTWARE IS THE FOUNDATION

Software is the foundation on which Europe's digital future depends. Software underpins every strategic domain of digital transformation: **artificial intelligence, digital twins, healthcare, mobility, energy, finance, public administration, defence,** and **critical infrastructure**.

02 AI MAKES SOFTWARE ENGINEERING MORE CRITICAL, NOT LESS

The software industry is currently being transformed with generative and agentic AI. AI is changing how software is specified, developed, tested, deployed, maintained, and evolved, while dramatically increasing both the speed and scale of software production.

This transformation creates enormous opportunities for Europe, but many of the existing software engineering challenges this transformation builds on remain unsolved. Trustworthy, explainable, secure, and sustainable AI systems are required for AI and software solutions to be used within healthcare, defence, and industry. But these can only be realized on top of equally trustworthy software engineering foundations. Rather than making software engineering less important, AI makes it more critical than ever.

03 EUROPE FACES A STRATEGIC INVESTMENT GAP

Yet a dangerous strategic gap persists: **foundational software research and public software infrastructure remain chronically underprioritized within EU funding frameworks.**

This is dangerous because trustworthy, efficient, and sustainable AI is impossible without advanced software engineering infrastructure.

Leading academic and research bodies, including Informatics Europe (IE), VERSEN, and ERCIM, have sounded the alarm [1,2,3,4,5] that Europe's investment in foundational software science and the public infrastructure that supports it is significantly lower than that of global competitors. This poses risks to the European economy, public safety, security, democracy and digital sovereignty.

04 EUROPE MUST ACT AT SCALE

If Europe is serious about its digital transformation ambitions, it must invest in software engineering at a scale that matches these ambitions.

The Informatics Europe and ERCIM Working Groups (WG) on Software propose:

ESSIA

European Software Science & Infrastructure Accelerator An 8-year, high-impact initiative with an indicative budget of **€0.5–1 billion**.

SRIA

Strategic Research and Innovation Agenda A shared vision of software engineering capabilities, research priorities, and infrastructure investments.

The need for a regulatory wake-up call

The need for a change in European software investment is widely recognised, and the concern is not new. European organisations have repeatedly warned that software research is structurally under-prioritised:

- VERSEN, the Dutch association for Software Engineering, wrote a manifesto that *Europe Needs Strong Software Research* [1].
- ERCIM, published a postion paper about the *The Importance of Software in Europe* [2].
- Informatics Europe, send an open letter concerning the omission of software in recent funding plans [3].
- At the International Conference on Software Engineering 2026 a position paper was published reclaiming software engineering as the enabling technology for the digital age [4].

These concerns are mirrored in practice. Consultations conducted by the Working Groups reveal a consistent set of challenges across European companies, from start-ups to large multinationals.

! COMMON INDUSTRY PAIN POINTS

Dependence — continued reliance on non-EU cloud and platform providers.

Supply chain — increasing exposure to vulnerabilities in complex, often open-source, dependency ecosystems.

Technical debt — large volumes of legacy systems that are costly and risky to modernise.

Complexity — challenges when managing long-term evolution of large-scale systems, variability, and versioning.

Compliance — with evolving regulatory requirements.

Responsible innovation — responsibly integrating transformative technologies like Generative AI.

Talent — difficulty attracting and retaining a specialised software workforce.

At the same time, the consequences of software fragility are becoming increasingly visible. Recent supply-chain backdoors, large-scale cloud outages and ubiquitous zero-day vulnerabilities demonstrate how local failures can propagate across interconnected systems.

REGULATORY WAKE-UP CALL

The regulatory clock is ticking. New technologies and EU regulations are fundamentally changing how software is built and maintained. The AI Act, NIS2, and CRA are shifting the compliance burden from paper-based procedures to the mandatory generation of technical evidence, requiring Software Bill of Materials (SBOMs), attestation, verifiable provenance, test oracles, and logs, all embedded directly within everyday software engineering processes.

If Europe delays strategic investment, it risks escalating technical debt, severe compliance bottlenecks, talent drain, and deeper strategic dependence on external, often non-European, platforms that control the necessary tooling and infrastructure.

REFERENCES CITED ON THIS PAGE

[1] VERSEN EU manifesto, *Europe Needs Strong Software Research*, 2022. ir.cwi.nl/pub/31390

[2] ERCIM, *The Importance of Software in Europe*. Position paper, 2025. ercim.eu/news/559-the-importance-of-software-in-europe

[3] Informatics Europe, *The Critical Omission of Software Technology in the EIC Work Programme 2025*. zenodo.org/records/15906491

[4] Vos, T.E.J., et al., *Reclaiming Software Engineering as the Enabling Technology for the Digital Age*, ICSE 2026, Rio de Janeiro, 2026. hal.science/hal-05468508

The Strategic Research and Innovation Agenda

The SRIA translates Europe's software engineering challenge into a structured agenda of capabilities. It is organised around **four strategic pillars** and **five cross-cutting perspectives**. Their intersection defines **twenty capability needs** for Europe.

4 PILLARS × **5 CROSS-CUTTING PERSPECTIVES** = **20 CAPABILITY NEEDS**

THE FOUR STRATEGIC PILLARS define the four core areas in which Europe needs to strengthen its software capabilities.

01

TRUSTWORTHY, SECURE AND COMPLIANT SOFTWARE SYSTEMS

Ensuring that software can be trusted, secured and shown to meet regulatory and societal requirements.

02

EVOLVABLE LARGE-SCALE SOFTWARE SYSTEMS

Enabling complex software systems to be maintained, adapted and evolved safely over long lifetimes.

03

SOVEREIGN AND RESILIENT DIGITAL INFRASTRUCTURE

Building the shared software foundations and infrastructure Europe needs for resilience and strategic autonomy.

04

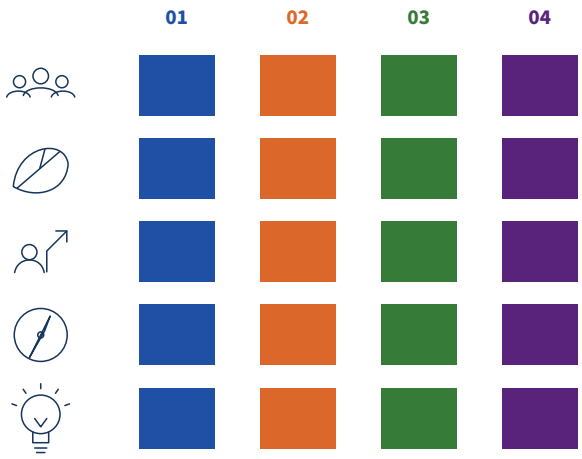
HUMAN AND ORGANISATIONAL CAPABILITIES

Strengthening the people, skills, organisations and practices needed to engineer software effectively.

FIVE CROSS-CUTTING PERSPECTIVES provide complementary lenses to examine every pillar.

Societal impact and public value	Who are we doing this for?
Green transition	How do we ensure the planet doesn't suffer?
Skills, workforce and development	Who will build and maintain this?
Strategy and governance	How should Europe organise and govern this?
Innovation	How do we create new opportunities?

THEIR INTERSECTION DEFINES THE SRIA



20 capability needs

Each intersection defines a concrete capability need, expressed as a **"We need..."** statement and developed through **Why do we need this?** and **How do we get there?**

The complete matrix on the following page presents all twenty capability needs.

The European Software Engineering SRIA at a glance

Four pillars × five cross-cutting perspectives = twenty capability needs. Each statement is repeated verbatim in its corresponding detailed section.

PILLARS / CROSS-CUTTING PERSPECTIVES	Trustworthy, secure and compliant software systems	Evolvable large-scale software systems	Sovereign and resilient digital infrastructure	Human and organisational capabilities
Societal impact and public value – Who are we doing this for?	We need citizens to be able to determine whether software is trustworthy.	We require software systems that can be dynamically adapting to changing societal and regulatory requirements.	We need public digital infrastructures that ensure citizens, businesses and governments can rely on European digital capabilities.	We must unleash the full potential of software in society by strengthening software awareness, literacy, inclusive participation, and organisational readiness.
Green transition – How do we ensure the planet doesn't suffer?	We need to achieve trustworthy, secure and compliant software without harming our planet.	We need software systems that can be running/adapted to more sustainable platforms and architectures as technologies and energy/resource requirements evolve.	We need public storage, compute, and service infrastructures with high ambition and low footprint for software including data and AI.	We need people and organisations capable of recognising the environmental consequences of software-related decisions and embed sustainability into everyday practices, incentives, and choices.
Skills, workforce and development – Who will build and maintain this?	We need professionals capable of assessing, verifying and assuring the trustworthiness of increasingly AI-generated and AI-empowered software systems.	We need new skills and curricula to prepare software engineers for the development, maintenance and modernization of software systems in the era of AI.	We need to attract, develop and retain the talent required to build resource-efficient software infrastructures that are also used for digital solutions for sustainability.	We need to develop, retain, renew, and transfer software expertise and strengthen the skills needed for continuous learning, knowledge transfer, and effective human–AI work.
Strategy and governance – How should Europe organise and govern this?	We need the capability to govern, regulate and assure the trustworthiness of (increasingly AI-generated) software systems.	We need European strategies that support the long-term maintenance, evolution and reuse of software as a strategic asset.	We need to a European strategy for critical software engineering resources, ensuring that Europe can build, maintain, and evolve its digital infrastructures independently and resiliently.	We need organisational and governance capabilities that enable institutions to adopt, procure, govern, and steward software responsibly throughout its lifecycle.
Innovation – How do we create new opportunities?	We need the capability to turn trustworthiness into a competitive advantage for European innovation.	We need software engineering methods that turn software evolution into a driver of innovation.	We need to expand the availability and ease the access to and use of European software infrastructures that support continuously evolving digital systems.	We need organisational capabilities that enable experimentation with software-enabled ways of working, learning from them, and turning successful experiments into sustained innovation.

From SRIA to European Action

The SRIA identifies the software capabilities Europe needs to safeguard its digital transformation. Turning this agenda into reality requires more than individual projects and isolated calls. It requires institutional ownership, strategic programming and coordinated funding.

Restore software's institutional home

Give strategic software clear ownership within the European Commission.

Software once had a clear institutional home and dedicated research programming within the European Commission. Today, software is more strategic than ever, yet responsibility is fragmented across programmes and organisational structures.

Turn the SRIA into a sustained European programme

ESSIA – European Software Science & Infrastructure Accelerator

We propose ESSIA as an 8-year European initiative bringing together the European Commission, Member States, industry and the research community to maintain the strategic agenda, set shared priorities and guide a coordinated €0.5–1 billion European funding programme.

ESSIA could be implemented as a long-term European Partnership for Software or equivalent instrument under FP10, coordinating European funding across four areas:

Research – *Develop Europe's future software capabilities*

Dedicated **research funding** to develop the software capabilities identified in the SRIA, from foundational research to applied innovation.

Infrastructure – *Build the foundations Europe needs*

Dedicated **infrastructure funding** for shared and sovereign European software capabilities such as federated software forges, dataset archives for AI-on-code, benchmarks, reproducible build clusters, and testing and verification environments.

Experimentation and compliance – *Enable innovation in a trusted regulatory environment*

Dedicated **sandboxes and real-world testing environments** to develop and validate tools and methods for compliance with the AI Act, Cyber Resilience Act and other European regulations.

Industrial adoption – *Turn research breakthroughs into European industrial capability*

Dedicated **innovation and adoption funding** through Innovation Actions (IA), Testing and Experimentation Facilities (TEFs), demonstrators and large-scale pilots to mature and validate European software technologies in real-world environments and accelerate their industrial adoption.

01

TRUSTWORTHY, SECURE AND COMPLIANT SOFTWARE SYSTEMS

Europe needs software systems whose trustworthiness, security and compliance can be understood, assured and governed throughout their lifecycle.

PILLARS / CROSS-CUTTING PERSPECTIVES	Trustworthy, secure and compliant software systems
Societal impact and public value – Who are we doing this for?	We need citizens to be able to determine whether software is trustworthy.
Green transition – How do we ensure the planet doesn't suffer?	We need to achieve trustworthy, secure and compliant software without harming our planet.
Skills, workforce and development – Who will build and maintain this?	We need professionals capable of assessing, verifying and assuring the trustworthiness of increasingly AI-generated and AI-empowered software systems.
Strategy and governance – How should Europe organise and govern this?	We need the capability to govern, regulate and assure the trustworthiness of (increasingly AI-generated) software systems.
Innovation – How do we create new opportunities?	We need the capability to turn trustworthiness into a competitive advantage for European innovation.

CROSS-CUTTING PERSPECTIVE

SOCIETAL IMPACT AND PUBLIC VALUE

Who are we doing this for?

WE NEED

We need citizens to be able to determine whether software is trustworthy.

WHY DO WE NEED THIS?

Society is increasingly reliant on software as the infrastructure underpinning all aspects of public and private life from e-Governance and e-Health to e-Democracy and e-Economy. As software becomes the primary means for civic participation and public services, the "black box" nature of current digital systems poses a systemic risk to social cohesion, individual autonomy and self-determination, and, ultimately, democracy.

Providing value in this context means moving beyond mere functional reliability; it means establishing **meaningful trust**. Citizens should not only be passive consumers of digital services but should be empowered to verify the integrity, security, and ethical alignment of the software they interact with. By fostering an ecosystem of trustworthy software, we uphold the European commitment to fundamental rights, prevent digital exclusion, and ensure that the digital transition strengthens, rather than erodes, the democratic contract.

HOW DO WE GET THERE?

- Advancing research on transparency and explainability techniques to move toward a standard where technical documentation and provenance (such as expressed via a Software Bill of Materials – SBOM) are readable, accessible and (widely) intelligible to all including the civil society.
- Developing public trust mechanisms and technical infrastructure, allowing citizens and regulators to verify compliance and security without compromising commercial IP.
- Advancing "Trustworthiness by Design" for consumer protection including new mechanisms for software liability and end-user recourse when software systems fail or act opaquely.
- Translating digital rights and ethical considerations into technical constraints to embed principles like data minimization, non-discrimination, and privacy into the software development lifecycle through advanced policy enforcement.
- Shifting the AI discourse from pure performance metrics to "societal alignment" for human-centred trustworthy AI, thereby ensuring that AI-driven software remains accountable to human oversight and remains aligned with European human rights standards.

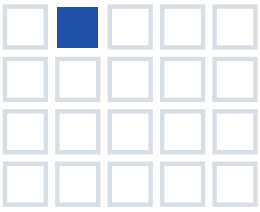
CROSS-CUTTING PERSPECTIVE

GREEN TRANSITION

How do we ensure the planet doesn't suffer?

WE NEED

We need to achieve trustworthy, secure and compliant software without harming our planet.



WHY DO WE NEED THIS?

As we strive for a more trustworthy, secure, and compliant digital ecosystem, we face a critical environmental paradox: the very processes designed to ensure software quality such as continuous security monitoring, rigorous verification and validation, assuring compliance, or the training and use of robust AI models are inherently resource-intensive.

If left unchecked, the "assurance tax", e.g. the energy required to make software safe, secure and trustworthy, will grow in tandem with the complexity of our software-based systems. Achieving a truly sustainable digital future requires us to optimize our assurance frameworks, to ensure that the pursuit of digital resilience and trustworthiness does not come at the expense of our climate goals, transforming "Green Software Engineering" from a niche practice into a foundational pillar of European digital sovereignty.

HOW DO WE GET THERE?

- Re-engineering Continuous Integration/Continuous Deployment (CI/CD) pipelines to minimise energy footprint, including new testing strategies that reduce redundant computations or new energy-optimised verification methods.
- Advancing AI-assistance for trust methods that provide high-assurance guarantees with a fraction of the current computational overhead.
- Researching on "cost of trust" metrics to quantify the environmental impact of compliance and security operations, incentivising the adoption of lean, eco-efficient assurance processes.
- Advancing certification processes toward real-time, distributed, and efficient certification methods that reduce the massive overhead of traditional, human- and resource-heavy certification cycles.

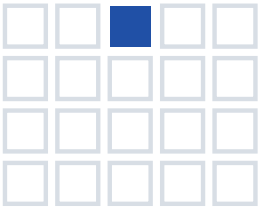
CROSS-CUTTING PERSPECTIVE

SKILLS, WORKFORCE AND DEVELOPMENT

Who will build and maintain this?

WE NEED

We need professionals capable of assessing, verifying and assuring the trustworthiness of increasingly AI-generated and AI-empowered software systems.



WHY DO WE NEED THIS?

The integration of generative and agentic AI into software engineering marks a fundamental shift in the software production industry. As AI systems take over the rote task of code generation, roles of software engineers are shifting from manual coding tasks to the critical tasks of designing, verifying, assuring, and governing complex AI generated and AI empowered software-based systems.

To maintain software resilience and trustworthiness, the software engineering workforce must evolve. We require a new breed of professionals who possess the “critical literacy” to audit AI-generated software and the technical expertise to manage the SDLC and assurance pipelines, but also who are able to work effectively in hybrid human-AI software development teams. Investing in these skills is essential to avoid a “technical debt crisis,” where society relies on software whose underlying architecture, logic and potential failure modes are no longer fully understood by humans.

HOW DO WE GET THERE?

- Developing approaches for rigorous critical analysis of AI-generated software, identifying e.g. hidden biases, security vulnerabilities, and logic flaws.
- Developing training programs to learn how to effectively supervise AI agents (as part of hybrid human-AI teams) when using them to accelerate routine tasks while maintaining final human accountability for system integrity.
- Researching modular, life-long learning frameworks that enable the workforce to keep pace with rapid technological evolution, including AI.
- Advancing curricula to develop interdisciplinary and transdisciplinary professionalism, reducing the gap between software engineering, legal aspects and ethics.

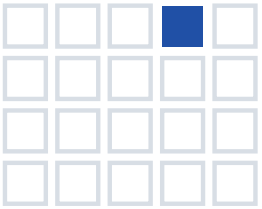
CROSS-CUTTING PERSPECTIVE

STRATEGY AND GOVERNANCE

How should Europe organise and govern this?

WE NEED

We need the capability to govern, regulate and assure the trustworthiness of (increasingly AI-generated) software systems.



WHY DO WE NEED THIS?

Trustworthiness in software is not just a technical requirement, it is a highly strategic issue of Europe's digital sovereignty. With generative and agentic AI, software innovation accelerates, but the risks to system integrity and quality scale proportionally. Also, the possibility for almost anybody to develop software (e.g., vibe coding, prompt or loop engineering) will increase the need for trustworthiness even further.

As the EU AI Act and the Cyber Resilience Act set new standards, we face a critical gap: traditional, manual certification processes cannot keep pace with the velocity of AI-assisted software production. To bridge this, we must evolve toward evidence-native software systems, toward architectures that autonomously and continuously generate the evidence required for regulatory compliance, security, and safety throughout their entire software lifecycle.

HOW DO WE GET THERE?

- Advancing standards and governance frameworks to align AI-assisted software engineering practices with European values.
- Developing next-generation assurance frameworks for complex, adaptive, non-deterministic software systems (benchmarks, testing methodologies, reliability guarantees, certification).
- Researching evidence-native software systems with automated compliance evidence generation (AI Act, CRA, NIS2).
- Encode legal compliance, ethical constraints, and safety requirements as formal guardrails in machine-readable formats (e.g., by using "compliance specification languages") that can be used for automated compliance verification at run-time.
- Providing rigorous, AI-supported mechanisms to ensure full traceability and security across the entire software supply chain, protecting against vulnerabilities in fragmented component ecosystems.
- Developing new quantitative and qualitative metrics for trustworthiness, robustness, and assurance that remain valid even when software systems are partially or fully AI-generated.

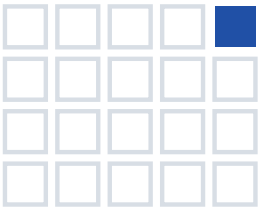
CROSS-CUTTING PERSPECTIVE

INNOVATION

How do we create new opportunities?

WE NEED

We need the capability to turn trustworthiness into a competitive advantage for European innovation.



WHY DO WE NEED THIS?

Europe must shift the narrative of software trustworthiness and sustainability: instead of viewing it as a regulatory or political burden, we must cultivate it as a core driver of innovation, entrepreneurship, and economic growth. As AI lowers barriers to software creation, the market will increasingly reward "trusted by design" and "green" software solutions. Trustworthy software engineering will be essential for enabling broader participation in innovation while maintaining security, compliance and quality. Trustworthiness should become an enabler of software innovation rather than a constraint.

HOW DO WE GET THERE?

- Developing human-AI innovation systems, methods for humans and AI agents to collaboratively create, validate, and evolve trustworthy and sustainable software innovations including methods for maintaining system-wide integrity while leveraging the creative potential of generative and agentic AI.
- Advancing democratized software creation, methods that enable citizens, startups, entrepreneurs, and domain experts to safely innovate using AI-powered software tools.
- Accelerating the adoption of trustworthy and sustainable software innovations, strengthening testbeds, regulatory sandboxes, shared infrastructures and collaboration models that help validate the trustworthiness of software technologies to move faster from research to deployment.

02
EVOLVABLE LARGE-SCALE SOFTWARE SYSTEMS

Europe needs software systems that can be maintained, adapted and evolved safely and efficiently as technologies, requirements and contexts change.

PILLARS / CROSS-CUTTING PERSPECTIVES	Evolvable large-scale software systems
Societal impact and public value – Who are we doing this for?	We require software systems that can be dynamically adapting to changing societal and regulatory requirements.
Green transition – How do we ensure the planet doesn't suffer?	We need software systems that can be running/adapted to more sustainable platforms and architectures as technologies and energy/resource requirements evolve.
Skills, workforce and development – Who will build and maintain this?	We need new skills and curricula to prepare software engineers for the development, maintenance and modernization of software systems in the era of AI.
Strategy and governance – How should Europe organise and govern this?	We need European strategies that support the long-term maintenance, evolution and reuse of software as a strategic asset.
Innovation – How do we create new opportunities?	We need software engineering methods that turn software evolution into a driver of innovation.

CROSS-CUTTING PERSPECTIVE

SOCIETAL IMPACT AND PUBLIC VALUE

Who are we doing this for?

WE NEED

We require software systems that can be dynamically adapting to changing societal and regulatory requirements.

WHY DO WE NEED THIS?

Society depends on large-scale software systems that evolve. Critical infrastructure, such as energy supply, mobile communication, emergency call services, payment and banking systems, traffic control, and government infrastructure (e.g., tax filing) form the foundation of the digital society. Many of these systems have been operational for years and will continue to be so for many years to come. As a result, it is crucial that these systems are continuously modernized and maintained in response to new requirements, changes in the environment (new platforms, languages, API integrations, service providers etc.), technological shifts (e.g., desktop to web, web to mobile, mobile to smart devices), or changes in legislation. As an example consider the extensions needed to address the European AI Act, Data Act, and Cyber-Resilience Act. Software is never finished since it has to keep pace with changing societal, technological and regulatory needs.

Furthermore, software grows fast. A typical hybrid electric car already contains 200 to 300 million lines of program code. Estimates from before the arrival of generative AI suggest that the global code footprint in lines of code exhibits exponential growth, with some estimates proposing a doubling period of approximately every 3.5 years. The impact of generative and agentic AI on software development is only exacerbating this trend. Software systems are already beyond the human scale in terms of comprehension, so without new and scalable techniques to aid the software engineers of the future this will cause societal bottlenecks.

To transcend the limitations of manual, error-prone maintenance, Europe must advance the automation of software evolution by synergizing generative and agentic AI, and rigorous methods. This will enable the needed shift toward continuous, self-optimizing software life cycles.

HOW DO WE GET THERE?

- Developing advanced techniques to reconstruct architectural models and extract domain knowledge, facilitating higher levels of abstraction and sustainable maintainability.
- Implementing methods for large-scale refactoring, re-platforming, and predictive evolution to ensure systems remain agile and adaptable throughout their lifecycle.
- Developing AI-based frameworks that specialize in complex maintenance tasks beyond mere code generation, i.e. modernization, software “shrinking,” code churn reduction, and technical debt remediation.
- Leveraging AI-driven refactoring for the secure migration of legacy codebases to modern programming languages and cloud-native platforms.
- Developing robust (and incremental) verification and validation (V&V) frameworks that ensure the reliability, safety, and trustworthiness of adaptive, probabilistic, and AI-integrated software systems, quantify uncertainty, and explain the quality level reached and risks mitigated.
- Deploying AI-assistants for advanced program analysis, automated documentation, and deep code understanding, thereby reducing the cognitive load on software engineering teams.

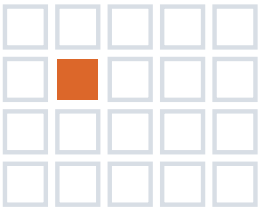
CROSS-CUTTING PERSPECTIVE

GREEN TRANSITION

How do we ensure the planet doesn't suffer?

WE NEED

We need software systems that can be running/adapted to more sustainable platforms and architectures as technologies and energy/resource requirements evolve.



WHY DO WE NEED THIS?

As software systems and AI-enabled development consume increasing amounts of energy and computing resources, the sustainability of our digital infrastructure has become a strategic imperative for Europe and a key concern in software engineering. We propose a paradigm shift towards sustainable software evolvability: by engineering systems that are inherently adaptable to future, energy-efficient architectures and (AI-)optimized hardware, we can lower the footprint, increase its positive environmental impact, and extend the overall software-based system lifespans. Evolvability enables this transition by making software easier and more cost-effective to maintain, modernise and migrate, and reducing the need for redevelopment.

Evolvability is the cornerstone of long-term digital sustainability. It relies on advanced software design techniques, such as variability-intensive systems (software product lines, software reuse, configurable software), the creation of meaningful sustainability-aware software ecosystems, specialised and maintainable software libraries and mastering dependencies across components (supply-chain management). However, the unprecedented scale and velocity of software evolution is challenging existing approaches and human comprehension.

HOW DO WE GET THERE?

- Establishing comprehensive techniques for monitoring, estimating and budgeting the energy consumption of software systems and the software engineering process itself.
- Developing theories and models to quantify the balance between the energy costs of AI-assisted software engineering against the productivity gains, ensuring a net-positive environmental impact.
- Advancing automated techniques to support the migration of software systems toward greener platforms and energy-efficient architectures.
- Advancing energy-aware software paradigms, architectural patterns, and software engineering workflows that integrate sustainability metrics into the core of the software development lifecycle.

CROSS-CUTTING PERSPECTIVE

SKILLS, WORKFORCE AND DEVELOPMENT

Who will build and maintain this?

WE NEED

We need new skills and curricula to prepare software engineers for the development, maintenance and modernization of software systems in the era of AI.

WHY DO WE NEED THIS?

The advent of GenAI is severely disrupting the practice of software engineering. The traditional skill sets taught at universities and vocational schools in the computer science and software engineering curricula are rapidly becoming out of date. A shift is occurring from requirements engineering, software design, writing code, and testing, towards a style of software development characterized as co-creation with AI agents, where AI agents will automate several of these tasks. As a result the focus moves from a developer culture based on code creation ("programming") to a culture that emphasizes scoping and designing combined with checking, reviewing, and verifying the artifacts produced by the AI agents.

Furthermore, the traditional view of software engineering as the development of new software falls short. In practice, long-lived, continuously evolving software systems dominate rather than greenfield software development. In the future, evolvable software will be analyzed, understood, modernized, and further developed over decades through the collaborative efforts of humans and AI agents. The key competency of the next generation of software engineers will no longer be primarily writing code, but rather the responsible design, management, and continuous evolution of complex software systems in collaboration with generative and agent-based AI systems. Software engineering is thus evolving from a discipline of construction to one of intelligent comprehension, evolution and governance of long-lived software systems.

A side effect of the AI adoption is that entry level jobs in software engineering are transforming, since the most basic tasks are automated by AI, and that senior level jobs are changing in nature: from solving key challenges to safeguarding the quality of AI generated code. As a result, enrollment in computer science and software engineering programs is dropping, creating a potential future bottleneck in the workforce. One main challenge is how to modernize software engineering education to allow companies to attract talent, with the right skills for keeping large-scale software systems healthy and evolvable, in the presence of AI.

HOW DO WE GET THERE?

- Developing new onboarding methods for junior software engineers working with large-scale systems.
- Overcoming the chasm from junior software engineers to senior software engineers in the presence of AI replacing many of the junior SE jobs.
- Establishing human-centric software engineering practices that preserve software engineer satisfaction and wellbeing in AI-assisted development, and mitigate burnout and sustain long-term productivity.
- Advancing research into team dynamics and collaborative paradigms within hybrid human-AI software engineering teams.
- Designing next-generation academic and professional training programs that harmonize foundational software engineering principles with state-of-the-art AI-assisted development tools.

CROSS-CUTTING PERSPECTIVE

STRATEGY AND GOVERNANCE

How should Europe organise and govern this?

WE NEED

We need European strategies that support the long-term maintenance, evolution and reuse of software as a strategic asset.

WHY DO WE NEED THIS?

Software Engineering is entering an era of continuous AI-enabled software evolution. Rather than alternating between development and maintenance, future software systems will undergo permanent co-evolution by humans and AI agents. Architectures, code, tests, documentation, security mechanisms, compliance artefacts, and operational configurations will evolve together in response to new technologies, regulations, user needs, and societal expectations. The central challenge is therefore no longer the efficient production of software, but the intelligent stewardship of software assets throughout their entire lifetime.

For Europe, this capability is strategically essential. By treating software as a long-term digital asset and investing in AI-enabled software evolution, Europe can protect its digital investments, strengthen technological sovereignty, reduce systemic inefficiencies, and create the foundations for innovation in emerging fields such as autonomous systems, robotics, smart cities, digital health, sustainable agriculture, and industrial transformation. The goal is not to slow down innovation, but to ensure that Europe's software foundations continuously evolve to support responsible, resilient, and human-centred digital futures.

HOW DO WE GET THERE?

- Establishing formal and methodological guardrails for AI-enabled software engineering that prevent maintenance lock-in and promote good software design, code quality, architecture and long-term maintainability.
- Supporting the adaptation of FAIR-inspired principles for software assets, making software searchable, findable, reusable and extendable under clear usage conditions.
- Developing methods and rigorous metrics to study the impact of AI-generated software on software quality, complexity, maintainability and technical debt.
- Scaling AI-assisted software modernisation and migration frameworks for long-lived software systems to enable the safe and productive transition of legacy systems into modern, energy-efficient, and secure software, thereby extending their utility without the need for costly and resource-intensive redevelopment.

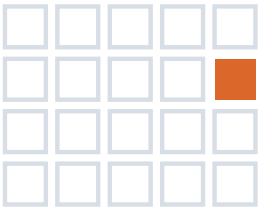
CROSS-CUTTING PERSPECTIVE

INNOVATION

How do we create new opportunities?

WE NEED

We need software engineering methods that turn software evolution into a driver of innovation.



WHY DO WE NEED THIS?

Software evolution is a multidimensional process driven by multiple, diverse stakeholders, who are drivers of innovative software changes. These stakeholders have multiple backgrounds (technical, legal, and domain expertise) and may not even be humans, if we take into account hybrid software engineering teams composed of humans and AI agents. They also have multiple concerns about the system evolution, whether it should be its functionality, performance or continuous adherence to EU regulations. It is key to keep all these stakeholders on the same page to keep innovation growing for the benefit of all. To sustain software innovation, we must bridge the cognitive and communicative gaps between these diverse actors.

HOW DO WE GET THERE?

- Developing advanced engineering abstractions (e.g., domain-specific languages, models and digital twins) that enable humans and AI agents to jointly understand, visualize, simulate and evolve software-intensive systems.
- Creating trustworthy human-AI engineering interfaces and collaboration mechanisms that enable effective interaction, oversight, and shared decision-making between humans and AI agents, ensuring that software systems remain fit for purpose and achieve the required quality attributes, including usability, reliability, robustness, safety, security, and adaptability.
- Developing AI-aware Software Development Lifecycles (SDLCs) that effectively integrate human expertise and AI throughout software evolution. SDLC must go beyond linear processes, fostering an iterative co-evolution where human expertise and AI's computational speed are seamlessly integrated.
- Implementing robust governance mechanisms that ensure "human-in-the-loop" control establishing verifiable standards for accountability and transparency to ensure that AI-assisted evolution remains aligned with European values, safety standards, and ethical guidelines.

03

SOVEREIGN AND RESILIENT DIGITAL INFRASTRUCTURE

Europe needs resilient and sustainable digital infrastructures that it can build, operate and evolve while maintaining control over critical software capabilities.

PILLARS / CROSS-CUTTING PERSPECTIVES	Sovereign and resilient digital infrastructure
Societal impact and public value – Who are we doing this for?	We need public digital infrastructures that ensure citizens, businesses and governments can rely on European digital capabilities.
Green transition – How do we ensure the planet doesn't suffer?	We need public storage, compute, and service infrastructures with high ambition and low footprint for software including data and AI.
Skills, workforce and development – Who will build and maintain this?	We need to attract, develop and retain the talent required to build resource-efficient software infrastructures that are also used for digital solutions for sustainability.
Strategy and governance – How should Europe organise and govern this?	We need to a European strategy for critical software engineering resources, ensuring that Europe can build, maintain, and evolve its digital infrastructures independently and resiliently.
Innovation – How do we create new opportunities?	We need to expand the availability and ease the access to and use of European software infrastructures that support continuously evolving digital systems.

CROSS-CUTTING PERSPECTIVE

SOCIETAL IMPACT AND PUBLIC VALUE

Who are we doing this for?

WE NEED

We need public digital infrastructures that ensure citizens, businesses and governments can rely on European digital capabilities.

WHY DO WE NEED THIS?

Europe's digital infrastructure remains highly dependent on non-European software technologies, platforms, and engineering toolchains, from cloud services and AI models to forges, CI/CD pipelines, IDEs, copilots, verification tools, and digital-twin technologies. More than 80% of Europe's digital infrastructures and services are currently provided by non-European actors (cf. https://table.media/assets/europe/european-parliament-resolution-of-22-january-2026.pdf), creating structural dependencies across public administrations, research, and industry. These dependencies expose Europe to geopolitical tensions, supply-chain disruptions, extraterritorial regulations, technological lock-in, and loss of control over critical software assets and services. In particular, Europe lacks guarantees that essential source code and software components will remain accessible, auditable, maintainable, and rebuildable if external providers withdraw access.

Achieving digital sovereignty therefore requires more than sovereign cloud, data, and AI infrastructures. Europe must also master the software engineering foundations, platforms, and toolchains through which critical software is built, operated, maintained, and evolved, while ensuring resilience, security, interoperability, trustworthiness, and long-term sustainability.

HOW DO WE GET THERE?

- Developing engineering methods, architectures, and platforms that enable interoperable and vendor-independent digital ecosystems;
- Advancing software technologies for resilient-by-design, secure-by-design, and evolvable critical infrastructures capable of operating under failures, cyberattacks, and changing geopolitical conditions;
- Advancing automated and AI-assisted engineering approaches that increase transparency, verifiability, and human control over complex software systems;
- Strengthening free and open-source software (FOSS) engineering practices and governance models that support European digital commons;
- Developing techniques for continuous assessment, certification, and assurance of large-scale distributed systems throughout their lifecycle;
- Preserving and intrinsically identifying source code as a public good—a sovereign, complete, and permanently referenceable archive of the software Europe depends on, enabling provenance, verification, and independent rebuilding regardless of any platform (e.g., Software Heritage, SWHID / ISO 18670);
- Establishing European sovereignty over the code corpora used to train and evaluate AI for software engineering (e.g., copilots and agents), ensuring that a strategic input to software production remains under European control.

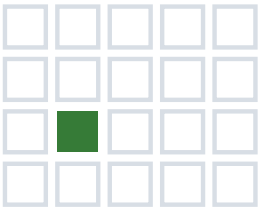
CROSS-CUTTING PERSPECTIVE

GREEN TRANSITION

How do we ensure the planet doesn't suffer?

WE NEED

We need public storage, compute, and service infrastructures with high ambition and low footprint for software including data and AI.



WHY DO WE NEED THIS?

Europe's sustainable and sovereign digital future depends on software infrastructures that are resilient, resource-efficient, and under European control. The shift towards renewable, decentralised, and variable energy requires software systems able to adapt computation in *time* and *space* according to energy and infrastructure constraints. Software engineering is therefore central to both *Green IT* and *IT for Green*, while reducing dependencies on non-European technologies and supply chains. Sustainability, resilience, security, and sovereignty must consequently be treated as interdependent quality attributes, supported by new methods, metrics, and automated toolchains throughout the software lifecycle.

HOW DO WE GET THERE?

- Engineering software for Europe's future green energy system, enabling computation to adapt in time and space to renewable-energy availability, carbon intensity, infrastructure capacity, and resource constraints.
- Advancing sustainability-by-design, with methods and tools to make environmental, economic, and social sustainability measurable and actionable throughout the software lifecycle.
- Developing AI-supported sustainable infrastructures, using automation, observability, prediction, and optimisation to improve energy efficiency, resource use, resilience, and lifecycle management.
- Improving metrics, certification, and evidence frameworks for the continuous assessment of sustainability, resilience, security, and sovereignty properties.
- Advancing human-centric and sovereign software systems, ensuring accessibility, transparency, maintainability, long-term preservation, and European control over critical software assets and infrastructures.
- Engineering trustworthy and continuously evolving systems, combining formal and model-based guarantees with the adaptability and learning capabilities of AI.
- Reusing, preserving, and rebuilding software, reducing environmental and economic waste while ensuring that critical systems remain independently rebuildable and evolvable from sovereign software assets.

CROSS-CUTTING PERSPECTIVE

SKILLS, WORKFORCE AND DEVELOPMENT

Who will build and maintain this?

WE NEED

We need to attract, develop and retain the talent required to build resource-efficient software infrastructures that are also used for digital solutions for sustainability.

WHY DO WE NEED THIS?

Building and operating sovereign infrastructures requires people who can not only develop software but preserve, curate, secure and evolve shared critical resources over decades — a scarce, strategic and undervalued skill set that European industry and academia currently lose to a handful of global players. The knowledge of how the foundational tools of software are built and assembled — forges, build systems, package managers, archives — must be grown and retained in Europe.

HOW DO WE GET THERE?

- Recognising and rewarding infrastructure and software-stewardship careers—including maintenance, curation, and operation of shared resources—which are essential but often undervalued relative to feature development;
- Building European curricula and doctoral training on software preservation, provenance, reproducibility, and the engineering of resilient and sovereign infrastructures;
- Strengthening recognition of European diplomas and PhDs, as well as academia–industry mobility, to help retain talent within the European ecosystem;
- Providing shared, hands-on training on the operation of European digital commons—including archives, mirrors, and forges—so that expertise is distributed across Member States rather than concentrated.

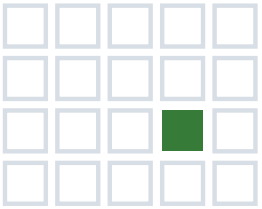
CROSS-CUTTING PERSPECTIVE

STRATEGY AND GOVERNANCE

How should Europe organise and govern this?

WE NEED

We need to a European strategy for critical software engineering resources, ensuring that Europe can build, maintain, and evolve its digital infrastructures independently and resiliently.



WHY DO WE NEED THIS?

The same way chips or precious metals are critical resources for building computers, software components and the tools to make and assemble them (infrastructures, forges, builders, …) are critical resources for software. Among these critical resources, the source code itself — together with the archives, forges and build systems that produce and assemble it — is the most foundational: without guaranteed, verifiable access to it, no infrastructure can be independently maintained or evolved.

Software engineering is the foundation of how software is built. For the sake of resilience and continuity, this knowledge needs to be developed in Europe so that, in the event of international events (embargo, new "Blabla act") from powerful external providers (GAFAM), European activity can not only continue running but also continue evolving.

HOW DO WE GET THERE?

- Identifying and prioritising the critical, most-depended-upon foreign software resources and toolchains, including forges, package and library registries, build systems, compilers, and CI/CD infrastructures;
- Guaranteeing long-term availability through a sovereign European archive of source code: preserving, mirroring, and making permanently referenceable the open-source components and infrastructures Europe depends on, so that activity can continue and evolve even if external providers withdraw—building on Software Heritage as a European digital common;
- Adopting intrinsic, platform-independent software identifiers (SWHID, ISO/IEC 18670) as the reference layer for provenance, traceability, and software supply-chain security, including SBOM and CRA-related requirements;
- Joining forces across Member States to avoid intra-European competition and duplication around these shared resources, including archives, mirrors, forges, and package managers;
- Structuring these shared resources as decentralised and federated services, resilient by design so that the failure of one node does not stop the others, and operating them as long-term-funded public infrastructures with clear stewardship and governance.

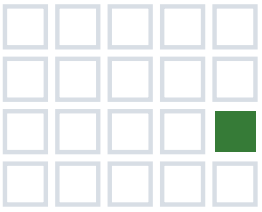
CROSS-CUTTING PERSPECTIVE

INNOVATION

How do we create new opportunities?

WE NEED

We need to expand the availability and ease the access to and use of European software infrastructures that support continuously evolving digital systems.



WHY DO WE NEED THIS?

Europe has largely ceded the initiative on the foundational infrastructures of software — forges, build systems, package registries, copilots — and with it much of its ability to shape how software is produced. But these foundations are now being reinvented (AI-assisted development, agentic pipelines), which reopens the competition: whoever provides the next generation of trustworthy, sovereign software-production infrastructure will set the terms. A sovereign, openly accessible archive of source code is also a unique European asset for AI innovation.

HOW DO WE GET THERE?

- Supporting research and deployment of next-generation European software-production infrastructures—including forges, build and CI systems, package management, and developer tooling—that are open, interoperable, and sovereign by design;
- Leveraging Europe's sovereign source-code archive as a foundation for trustworthy AI for code, including training, evaluation, and provenance-aware assistants, and as an evidence base on the software Europe produces and depends on (e.g., the Public Code Observatory);
- Facilitating access to and reuse of these European infrastructures by startups, SMEs, public administrations, and researchers, lowering the barriers to building on sovereign foundations;
- Strengthening academia–industry collaboration to move innovations from research into operational infrastructures.

04

HUMAN AND ORGANISATIONAL CAPABILITIES

Europe needs the skills, organisations, leadership and engineering capabilities required to build, operate and evolve software-intensive systems responsibly and effectively.

PILLARS / CROSS-CUTTING PERSPECTIVES	Human and organisational capabilities
Societal impact and public value – Who are we doing this for?	We must unleash the full potential of software in society by strengthening software awareness, literacy, inclusive participation, and organisational readiness.
Green transition – How do we ensure the planet doesn't suffer?	We need people and organisations capable of recognising the environmental consequences of software-related decisions and embed sustainability into everyday practices, incentives, and choices.
Skills, workforce and development – Who will build and maintain this?	We need to develop, retain, renew, and transfer software expertise and strengthen the skills needed for continuous learning, knowledge transfer, and effective human–AI work.
Strategy and governance – How should Europe organise and govern this?	We need organisational and governance capabilities that enable institutions to adopt, procure, govern, and steward software responsibly throughout its lifecycle.
Innovation – How do we create new opportunities?	We need organisational capabilities that enable experimentation with software-enabled ways of working, learning from them, and turning successful experiments into sustained innovation.

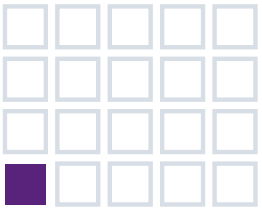
CROSS-CUTTING PERSPECTIVE

SOCIETAL IMPACT AND PUBLIC VALUE

Who are we doing this for?

WE NEED

We must unleash the full potential of software in society by strengthening software awareness, literacy, inclusive participation, and organisational readiness.



WHY DO WE NEED THIS?

Software systems are built by people and for people. Software can only create broad public value when people and organisations understand how it shapes access, participation, wellbeing, and everyday life. Raising software awareness and literacy helps prevent exclusion, makes the societal effects of software more visible, and enables citizens, communities, and organisations to engage with software-enabled change rather than merely undergo it.

Yet these capabilities are unevenly distributed. Many citizens and organisations lack the software literacy needed to understand how software-mediated decisions affect them, while affected groups are often involved only after technologies and processes have already been designed (if at all). Organisations also differ greatly in their capacity to evaluate, adopt and govern software responsibly. Without stronger human and organisational capabilities, digitalisation risks reinforcing existing inequalities and excluding precisely those people and organisations that have the least capacity to influence it.

HOW DO WE GET THERE?

- Improving public and organisational software literacy, inclusive engagement with affected groups.
- Developing organisational readiness and maturity models as well as assessment mechanisms for responsible software adoption, taking into account differences in resources, expertise and governance capacity between SMEs, public organisations, civil-society organisations and large enterprises.
- Developing methods and metrics to understand who benefits, who is excluded, and how software practices affect people with different and intersecting identities.

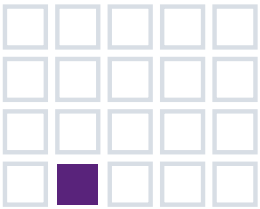
CROSS-CUTTING PERSPECTIVE

GREEN TRANSITION

How do we ensure the planet doesn't suffer?

WE NEED

We need people and organisations capable of recognising the environmental consequences of software-related decisions and embed sustainability into everyday practices, incentives, and choices.



WHY DO WE NEED THIS?

Software has environmental consequences not only through its technical design, but also through the everyday choices of the people and organisations that create, use, and maintain it. Building awareness of these consequences helps teams and organisations treat energy use, resource consumption, longevity, and maintainability as normal concerns in software work, rather than as external or after-the-fact issues.

Yet the environmental consequences of software-related decisions are often invisible to the people making them. Developers, managers and users rarely receive timely information about the resource implications of design, development and usage choices, while organisational incentives continue to emphasise functionality, speed and cost over long-term sustainability. Responsibility for sustainable software is therefore diffuse, and environmentally responsible behaviour is difficult to turn into routine organisational practice.

HOW DO WE GET THERE?

- Developing practical methods that help people recognise and assess the environmental impact of software-related decisions throughout its lifecycle.
- Embedding sustainability criteria, responsibilities and incentives into everyday development, development and maintenance routines, making resource use visible to teams and users, so that environmentally responsible software practices become part of normal organisational decision-making rather than an after-the-fact concern.
- Developing feedback mechanisms, decision-support tools and organisational incentives that make the environmental consequences of software and AI use visible at the point where developers, managers and users make decisions, enabling them to balance expected benefits against energy and resource costs.

CROSS-CUTTING PERSPECTIVE

SKILLS, WORKFORCE AND DEVELOPMENT

Who will build and maintain this?

WE NEED

We need to develop, retain, renew, and transfer software expertise and strengthen the skills needed for continuous learning, knowledge transfer, and effective human–AI work.

WHY DO WE NEED THIS?

Software-intensive organisations depend not only on individual technical skills, but on their ability to develop, retain, renew and transfer expertise over time. Generative and agentic AI are changing roles, career paths, team structures and the division of responsibility between humans and machines. At the same time, critical software knowledge is often tacit, concentrated in a small number of experienced professionals, and difficult to transfer when people change roles or leave organisations.

These capabilities have not developed at the pace of technological change. Training and career structures still largely assume relatively stable roles and technologies, while organisational incentives often prioritise short-term delivery over mentoring, knowledge transfer and long-term capability development. Expertise is frequently treated as an individual asset rather than an organisational resource, and responsibility for reskilling, onboarding and human–AI work design is fragmented across teams and management layers. As AI automates parts of software work, these weaknesses become more consequential: organisations risk losing entry points for junior professionals, weakening opportunities for learning through practice, and concentrating critical knowledge and decision-making in ever smaller groups.

Europe therefore needs to strengthen the human and organisational mechanisms through which software expertise is created and sustained: onboarding and mentoring, continuous learning, knowledge transfer, leadership, career development, inclusive team practices and effective human–AI work design. Without these capabilities, technological advances may increase short-term productivity while weakening long-term organisational expertise, developer wellbeing and resilience.

HOW DO WE GET THERE?

- Developing competence and career frameworks for software-intensive organisations that reflect changing roles, responsibilities and career paths in AI-assisted development.
- Developing mechanisms for onboarding, mentoring, succession and transfer of tacit software knowledge, particularly when routine entry-level tasks are increasingly automated.
- Establishing lifelong organisational learning models that enable software professionals, managers and technical leaders to continuously renew their capabilities.
- Researching effective human–AI work designs for software engineering teams, including task allocation, authority, autonomy, human intervention, accountability, expertise and wellbeing.
- Developing organisational and leadership practices that attract, retain and sustain diverse software-engineering talent and prevent erosion or excessive concentration of critical expertise.

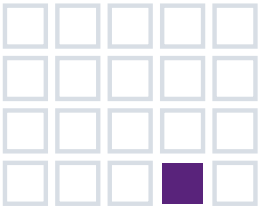
CROSS-CUTTING PERSPECTIVE

STRATEGY AND GOVERNANCE

How should Europe organise and govern this?

WE NEED

We need organisational and governance capabilities that enable institutions to adopt, procure, govern, and steward software responsibly throughout its lifecycle.



WHY DO WE NEED THIS?

Institutions increasingly depend on software, but often make adoption, procurement, and stewardship decisions without sufficient organisational understanding of software's long-term consequences. Building this capability provides value because it helps institutions make responsible choices, avoid dependency on ad hoc technical judgement, and ensure that software decisions remain aligned with their mission, responsibilities, and accountability structures.

Today, however, software is still too often governed as a short-term acquisition or project rather than as a long-lived organisational asset. Responsibility is fragmented across technical teams, procurement, management, legal functions and external suppliers, while many decision-makers lack the software expertise needed to assess long-term consequences. These challenges are particularly acute for smaller organisations, which cannot simply reproduce the specialised governance structures of large enterprises.

HOW DO WE GET THERE?

- Developing organisational decision frameworks for software adoption, procurement and long-term stewardship that explicitly account for maintainability, dependency, skills, sovereignty, sustainability and exit options over the software lifecycle.
- Establishing criteria and assessment methods for evaluating software technologies throughout their lifecycle.
- Clarifying organisational roles and responsibilities across the software lifecycle to strengthen accountability and governance.
- Strengthening software procurement and stewardship practices that support long-term sustainability and organisational resilience.
- Implementing governance mechanisms that enable organisations to monitor, evaluate and adapt software use over time.

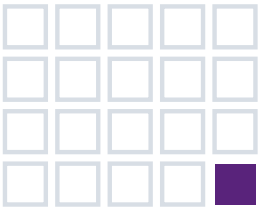
CROSS-CUTTING PERSPECTIVE

INNOVATION

How do we create new opportunities?

WE NEED

We need organisational capabilities that enable experimentation with software-enabled ways of working, learning from them, and turning successful experiments into sustained innovation.



WHY DO WE NEED THIS?

Innovation increasingly emerges from the interaction between people, organisations, and software systems, rather than from technology alone. Understanding these interactions provides value because it helps organisations turn software-enabled possibilities into sustained new practices, products, and services, while making better use of human creativity, organisational knowledge, and software capabilities.

The central challenge is therefore not only generating new ideas with software, but creating organisational conditions in which experimentation can be evaluated, learned from, scaled and embedded into lasting practice.

HOW DO WE GET THERE?

- Studying organisational structures, incentives and leadership practices that enable effective experimentation and collaboration between people, teams, organisations and AI-enabled software systems.
- Developing organisational environments in which new forms of human–AI collaboration can be safely experimented with, evaluated and adapted before being scaled.
- Advancing methods for capturing and transferring knowledge from software-enabled experiments across teams and organisational boundaries, so that successful innovations do not remain isolated local practices.
- Developing methods and metrics to understand which organisational collaboration patterns enable software innovations to move from experimentation to sustained adoption and measurable public or economic value.