

Preserving for Eternity, Coding for Today: The Role of Pseudo-Developers in Cultural Heritage Institutions

Authors

Annelies Cosaert*, Sustainability Unit, Royal Institute for Cultural Heritage (KIK-IRPA), Brussels, Belgium.
Email: annelies.cosaert@kikirpa.be, Orcid: <https://orcid.org/0009-0004-1269-4465>

Bhavesh Shah, English Heritage, Rangers House, Chesterfield Walk, Blackheath, London SE10 8QX, UK.
Email: bhavesh.shah@englishheritage.org.uk, Orcid: <https://orcid.org/0000-0001-8673-0589>

Co-Authors**

Clara Penagos, Dendrochronology lab, Laboratory department
Email: clara.penagos@kikirpa.be, Orcid: <https://orcid.org/0000-0001-7962-9415>

Wim Fremout, Research data management Unit
Email: wim.fremout@kikirpa.be, Orcid: <https://orcid.org/0000-0002-1684-377X>

Nicolas Nadisic, FED-tWIN Professor-Researcher (UGent/KIK-IRPA)
Email: nicolas.nadisic@ugent.be, Orcid: <https://orcid.org/0000-0002-5993-7194>

Roald Hayen, Monuments lab, Laboratories department
Email: roald.hayen@kikirpa.be

Sebastiaan Godts, Monuments lab, Laboratories Department
Email: sebastiaan.godts@kikirpa.be, Orcid: <https://orcid.org/0000-0003-2189-2995>

Stephanie Buyle, Research data management Unit
Email: stephanie.buyle@kikirpa.be, Orcid: <https://orcid.org/0000-0002-1481-8749>

All: contributors from the Royal Institute for Cultural Heritage (KIK-IRPA), Brussels, Belgium.

Tools and project developed by these colleagues are described at the end of the article.

* and ** All are referred to in the article using their initials,

Abstract

This paper explores the evolving role of domain experts who code — “pseudo-developers” — in cultural heritage institutions. By tracing a historical arc from early information organization systems like Belgium's Mundaneum to today's AI-augmented programming environments, we identify persistent challenges in balancing technical rigor with domain expertise. Through a questionnaire examining coding practices at the Royal Institute for Cultural Heritage (KIK-IRPA), we identify two distinct approaches: Research Data Managers focused on infrastructure and interoperability, and Domain-Specific Coders prioritizing analysis and problem-solving. These profiles reveal tensions between immediate pragmatic needs and long-term preservation goals—a particularly critical balance in a field dedicated to multi-generational timescales. As artificial intelligence tools increasingly reshape scientific practice, we find striking parallels with previous technological transitions, suggesting that while tools evolve, core challenges of documentation, reproducibility, and knowledge transfer remain. We propose institutional adaptations, collaborative frameworks, and research directions to support sustainable computational practices in cultural heritage, ultimately arguing that code itself represents a form of heritage requiring preservation alongside the cultural artifacts it helps document and analyze.

Introduction

Heritage has always been interwoven with record keeping. Researchers collect an increasing amount of records on objects, crafts, religious and other practices. They are formulas, written down, improved and optimized over years. Information about a thing that lives parallel to the thing itself. Together, they are more valuable than when apart and shape the culture of the world people live in today.

This paper explores the parallel evolution of computing practices and cultural heritage (CH) preservation approaches. Building on our previous research addressing tool development challenges within the sector

(Cosaert and Beltran, 2022), we examine the rapidly growing presence of pseudo-developers in cultural heritage institutions. By combining historical perspectives with insights from contemporary interviews, we identify the specific needs and challenges facing today's coding cultural heritage specialists, offering a foundation for future collaboration and methodological development.

Cultural institutions and early information technology: the Mundaneum

As an illustration of early information technology stands a lesser known example from Belgium, the 'Mundaneum', a paper version crossing between a modern search engine and Wikipedia. It was designed as a new type of museum, archive, library and international knowledge center. This utopian idea physically existed from 1920 to 1935 in the buildings in the 'Parc de Cinquantenaire' in Brussels.

The idea was brought to life by two Belgian legal scholars and pacifists, Paul Otlet (1868-1944) and Henri La Fontaine (1854-1943, winner of the Nobel Peace Prize in 1913). At the basis of their Universal Bibliographic Repertory (RBU) lies the usage of the Universal Decimal Classification (UDC, published in 1905, based on the earlier Dewey Decimal Classification, DDC). UDC is a numerical code system to organize all human knowledge across language barriers (Wikipedia, 2025).

The goal of RBU was to make knowledge accessible to the world. In consultation with leading international scholars the institution introduced and combined innovative ideas (Rayward, 2010), which are still recognizable today such as:

- Hypertext-like connections: Otlet's system of connecting related documents through classification numbers anticipated modern hypertext links.
- Modular information: breaking knowledge into smaller, interconnected parts parallels modern information architecture and API-based systems.
- Personal computing workstations: Otlet's concept of the "Mondothèque" with multiple information functions combining text, sound and image, resembles today's personal computers.
- Remote information retrieval: the telegraph-based query service, where users could request information resembles modern database queries.
- Standardized information exchange: The card format (12.5 x 7.5 cm) that was used, is comparable to modern data exchange standards that enable system interoperability.

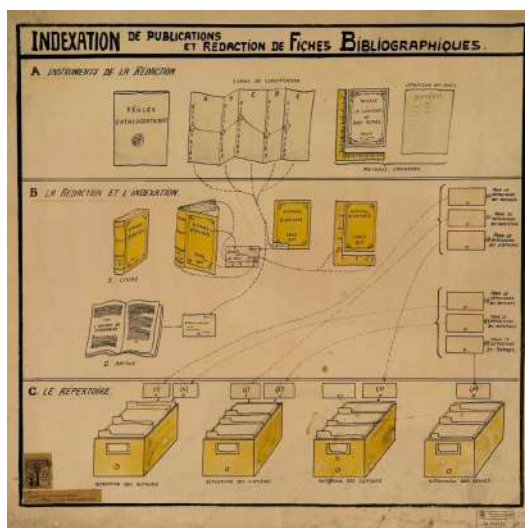
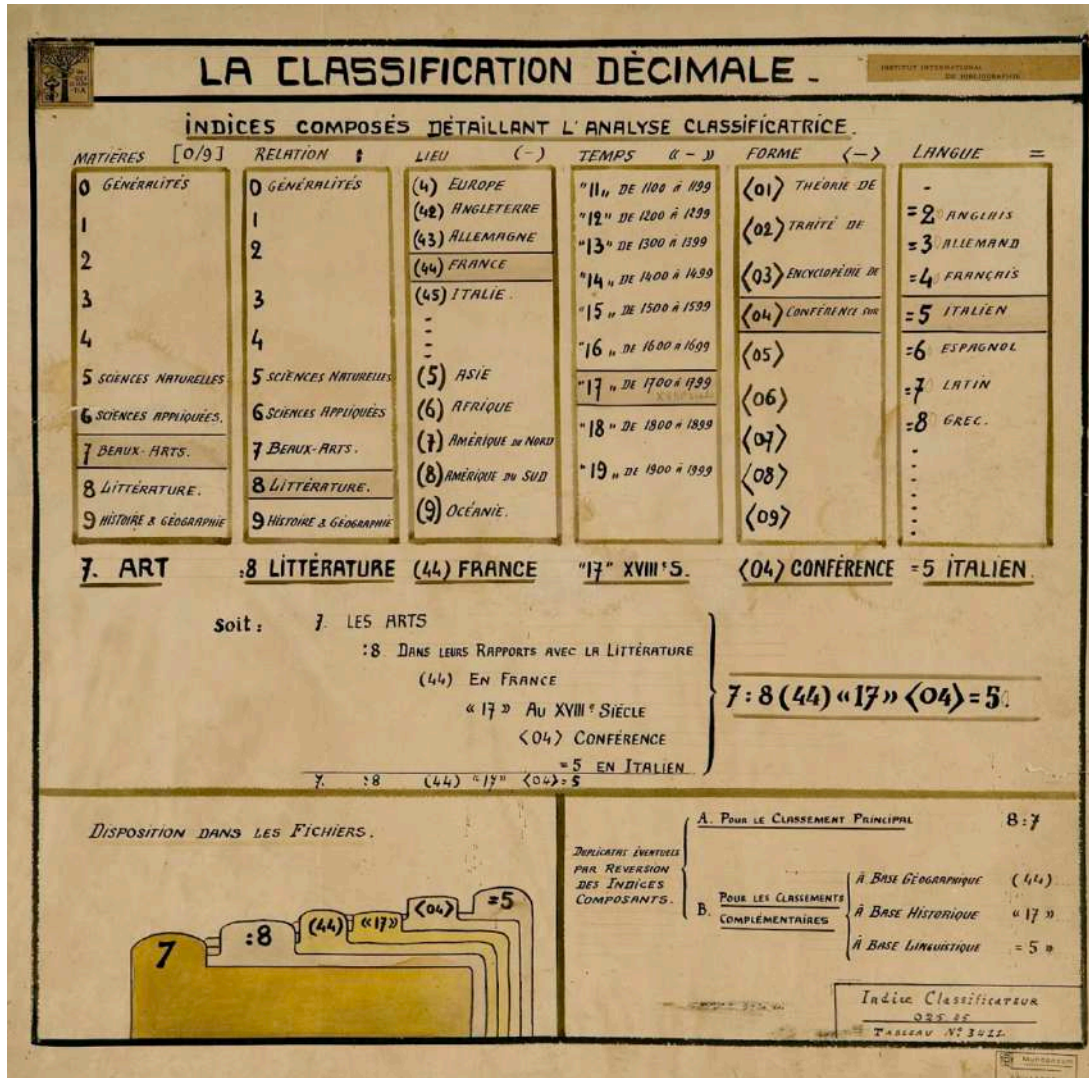
These innovations represent an early model of what can now be seen in heritage institutions: specialists creating systems to organize knowledge without being trained as information technologists. The Mundaneum's creators were effectively early pseudo-developers or domain experts who created technical solutions despite lacking formal training in system design. This pattern continues today as heritage specialists adapt digital tools to preserve cultural knowledge.

"Mankind is at a turning point in its history. The mass of data acquired is astounding. We need new instruments to simplify it, to condense it, or intelligence will never be able to overcome the difficulties imposed upon it or achieve the progress that it foresees and to which it aspires." (Translated to English from French, from Otlet, 1934)

Otlet's century-old observation resonates powerfully today in heritage preservation. One hundred years after the interbellum (the period between the First and Second World War), the world finds itself in turbulent times again. The volume of digital heritage data now being generated — from high-resolution scans of artifacts to environmental monitoring data in collection spaces — combined with new technologies, will be a challenge for years to come. The question remains how heritage professionals, often lacking formal

Figure 1 and 2: A representative scheme for the Universal Decimal Classification - Index formation (top), and a diagram that illustrates the protocols for the indexing of publications and editing bibliographic records (bottom left). Located at the Mundeneum in Mons, Belgium. Source: Fédération Wallonie Bruxelles de Belgique,

Figure 3 (bottom right): Some larger concepts were summarized in the 'Encyclopedia Universalis Mundaneum', 1920. This was a collection of so-called 'planches' creating visual synthesis of knowledge through standardized plates and diagrams, resembling a Wikipedia page of today. Source: Mundaneum asbl, Author: Paul Otlet



technical training but now needing to engage with code and data science, can "provide instruments to condense this information" and "achieve the progress to which it aspires" as Otlet envisioned.

Many of the Mundaneums records got lost due to international political pressure, war and a lack of preservation efforts. However, renewed interest led to the registration as UNESCO World Heritage in 2013. Today, visitors can see the remaining part of the collection in the Mundaneum in Mons, Belgium. The museum exhibitions focus on themes such as anarchism, pacifism and feminism, paying homage to the fundamental interests of its creators (Veldhoen, 2023).

Continued evolution: the role of science, data science, developers and pseudo-developers

Just as the Mundaneum represented an innovative approach to organizing information in the pre-digital age, today's heritage professionals must navigate increasingly technical approaches to preservation. This evolution requires understanding the changing relationships between domain expertise and programming skills.

The following pages explore the history, future and challenges of the role of non-trained researchers using code, specifically in the heritage sector and within preventive conservation. For this purpose some concepts are defined:

Within data science, two roles are highlighted:

- A regular developer, who has enjoyed a formal education or extensive training in computer science or programming, resulting in a deep understanding of software architecture and development principles. They have the ability to build complex systems from the ground up. They are used to working in teams or in tandem and integrate proper version control and source code management in their working processes.
- A pseudo-developer is a researcher, often with a background in exact sciences but possibly also in humanities. They are experts in their field, but have no or limited training in active code development. They know one or more coding languages and write code to automate some of their processes. They generally work alone and write code in a goal-oriented way, where reproducibility is not the main focus. They understand the logic of programming and can modify existing code, but may not build complex systems from scratch.

Data science itself has evolved alongside programming practices. As Donoho (2017, p. 746) notes in his comprehensive paper '50 years of data science', the distinction between traditional statistics and modern data science is largely one of scale and approach rather than fundamental difference. For heritage professionals, this evolution has meant adapting statistical methods developed for small, carefully collected datasets to the massive, messy data generated by digital preservation efforts.

The book 'The fourth paradigm – data-intensive scientific discovery' (Hey et al., 2009, p.xviii-xix, based on a talk by Jim Gray, 2007), describes how scientific approaches have evolved through four paradigms: empirical description (thousands of years ago), theoretical modeling (hundreds of years ago), computational simulation (decades ago), and most recently, data-intensive methods that unify theory, exploration and simulation.

This evolution is particularly visible in heritage preservation, where conservators have moved from empirical observation of deterioration, to theoretical models of decay processes, to computational simulations of preservation environments, and now to data-intensive approaches that monitor collections in real-time across multiple variables.

At present, artificial intelligence (AI) may represent a fifth paradigm, fundamentally changing how scientific questions are approached by automatically identifying patterns and generating insights that might otherwise remain hidden. For heritage professionals, this could mean AI systems that predict deterioration before it's visible or identify connections between collection items that reveal new historical contexts.

The evolution of programming practices can be understood through four distinct periods (Abbate, 2012; Ceruzzi, 2003):

- Coding pioneers (1950s-1970s): direct hardware interaction requiring specialized skills
- Language development (1970s-1990s): higher-level languages with structured programming concepts
- Tool-assisted programming (1990s-2020): development environments supporting collaboration
- AI-augmented programming age (2020-Present): large language models (LLMs) generating and assisting with code creation

For heritage professionals operating as pseudo-developers, each of these transitions has simultaneously created new opportunities and challenges. While programming has become more accessible with each age, the growing complexity of systems means that the gap between casual coding and professional development practices continues to widen. The following sections examine how these tensions appear specifically in conservation science and how new AI tools might bridge this divide or is enlarging it as we speak.

1. The pseudo-developers journey: from DIY to specialized and back again?

1.1. The coding pioneers: from hardware to software

Let's go back to a time before developers and pseudo-developers. People programming computers were simply scientists, often mathematicians, physicists, electrical engineers, linguists and philosophers. They had no choice but to code their own solutions, creating groundbreaking tools despite limited or no formal computing education.

"The activity known as computer programming was not foreseen by the pioneers of computing. During the 1950s they and their customers slowly realized: first, that it existed; second, that it was important; and third, that it was worth the effort to build tools to help do it." (Ceruzzi, 2003, p.108)

The job description of the women calculating ballistic trajectories in 1946, later working on the 'ENIAC' program (fig. 4 and 5), was simply 'computer'. While the creation of the electro-mechanical hardware was left to men, the mathematical computing was left to women. They introduced subroutines, nesting and debugging practices as they went along, translating their experiences to train a new generation (Abbate, 2012) and collaborating on early language development such as COBOL and FORTRAN.

Did you know? The word 'computer' in French means "to calculate." When IBM introduced their computer in France, the local branch found it too similar to 'calculatrice' (handheld calculator). They rebranded their machines as 'ordinateurs' from 'ordonnateur,' meaning "to put things in order." This linguistic reasoning mirrors how early scientific programmers viewed their work: not just calculation, but creating order from chaos through algorithms and structured data analysis. Early computing was a pioneering approach that required both domain knowledge and the ability to impose logical structure on complex problems.

It's important to understand that early programming was strongly based on computer hardware architecture. There is a gradual evolution from physical programming (involving plugging and unplugging of cables) such as for the ENIAC, over the use of switches for the input binary code, to the introduction of assemblers, using more telling commands such as ADD or STORE instead of a string of 1's and 0's. For coding pioneers, there was no real difference between 'coding' and 'using' a computer.

The time stamp of early computer development, at the beginning of the Second World War, and followed by the Cold War is unfortunate for many computing pioneers in the USSR, Japan and Germany. The U.S. post-war dominance and intact infrastructure leading to commercial success in the west made them achieve dominance in the long run. However, it was Konrad Zuse that built the Z3, in 1941's Germany, the first functional programmable computer. He envisioned computing to be an integrated whole of hard and software. He wrote the first high-level programming language (between 1942-1945) called Plankalkül, including things as: arrays and records (data structures), hierarchical program structures, loop control structures, conditional statements and subroutines. This concept is significantly more flexible since, unlike assemblers (1950s), it is designed to be machine independent.

Ceruzzi (2003) starts 'the early history of software' in 1952. With some predecessors, early languages such as FORTRAN (IBM, 1957) and ALGOL (EU collaboration, 1958-60) still have a strong mathematical basis. The basic goal was to facilitate easier computing. Languages like COBOL (1959) and SNOBOL (1962) on the contrary tried to focus on respectively business and textual data, focussing on domain-specific solutions. COBOL was deliberately moving away from numerical commands and shifted toward English for its commands.

During the 60's, the fields of data science and software engineering emerged. American statistician John Tukey stated in his 1962 influential paper 'The Future of Data Analysis' (p.4-5) that he was interested in a future as a data scientist, predicting many of the challenges the field might face and requirements it might need in the future. The term software engineering was first coined at the 1968 NATO Software Engineering Conference in Garmisch, Germany. This conference addressed what was becoming known as the "software crisis" - the growing difficulty of writing correct, understandable, and verifiable computer programs.

By the end of the 1960s there were about 6000 general purpose computers installed in the U.S. predominantly being present in businesses, government agencies and universities. The hardware is present and commercially available, specific purpose languages and software are created, and courses in computer engineering exist. At this pivotal moment, using a computer and programming a computer began to diverge as distinct activities. As specialized software applications emerged, many users could accomplish their tasks without understanding programming. This separation created a growing divide between computer scientists and other domain scientists who increasingly became consumers rather than creators of computing tools.

The ACM's (Association for Computing Machinery, founded in 1947) embrace of Dijkstra's ideas helped transform his somewhat radical views into mainstream practice, accelerating the transition from ad-hoc programming approaches to more formal software engineering methodologies.

For pseudo-developers, this trade-off is particularly relevant. While the degree of structure varies by programming language — with Python encouraging structure through indentation and older languages like C and FORTRAN allowing more freedom — structured approaches generally require more initial investment in learning proper programming practices but ultimately lead to more sustainable, maintainable solutions (Dahl et al., 1972).

From the 1980s through the 1990s, programming languages proliferated to address specialized domains. C++ (1985) extended C with object-oriented features for systems programming, while Visual Basic (1991)

made building Windows applications accessible to casual programmers. Domain-specific languages emerged to serve particular communities — MATLAB (commercially released in 1984) for mathematical computing, R (1993) for statistical analysis, and later Python (gaining prominence in the late 1990s) for scientific computing.

Figure 4 and 5: programming the ENIAC (Electronic Numerical Integrator And Computer) in Philadelphia, Pennsylvania at the Ballistic Research Laboratory (left). Simmers, holding ENIAC board; Taylor, holding EDVAC board; Beck, holding ORDVAC board; and Stec, holding BRLESC-I board (right). Source: U.S. Army/ARL Technical Library Archives

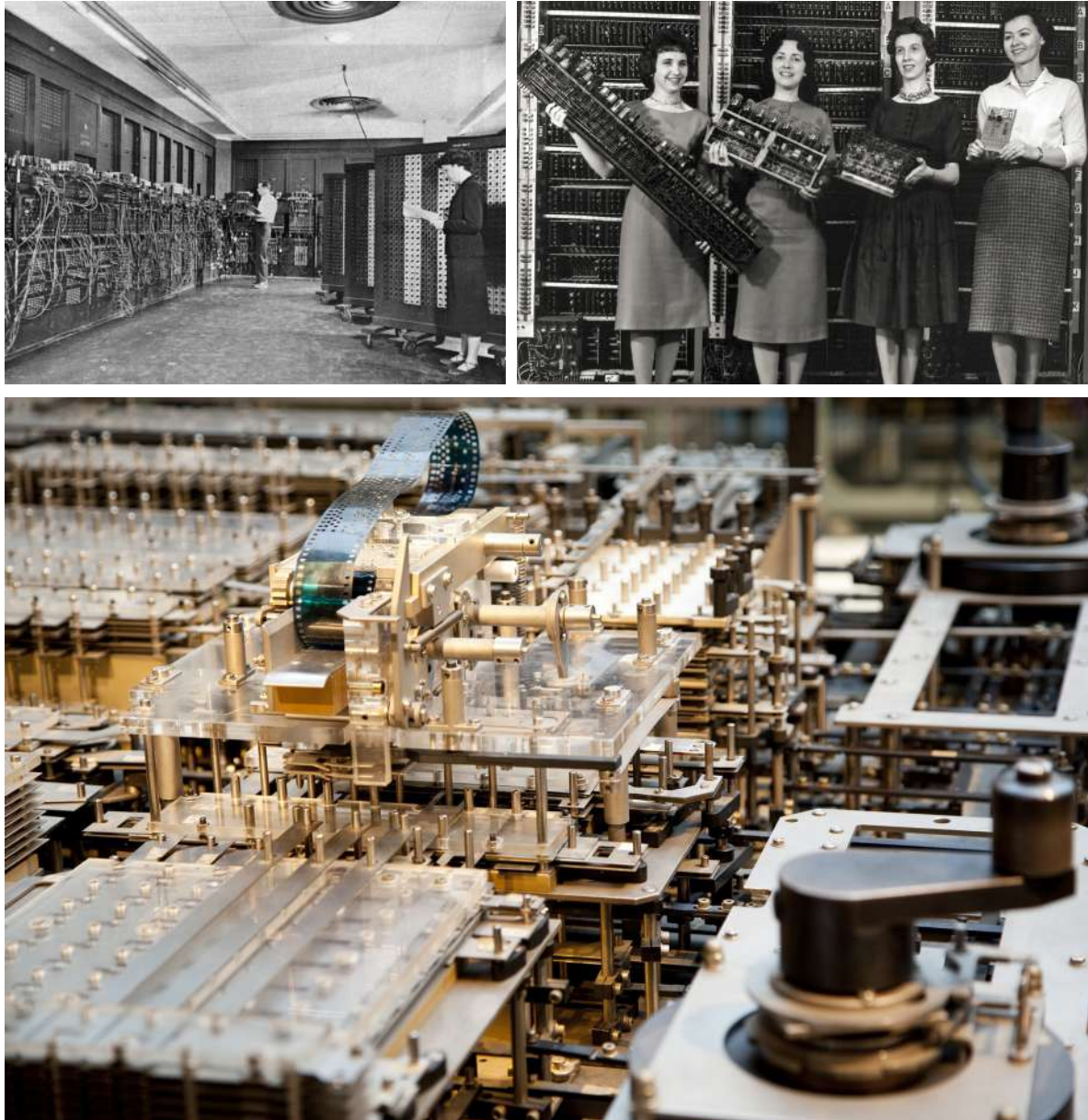


Figure 6: Detail of a replica of the Z1, the first, never properly functioning computer (1936) by Konrad Zuse. The Z1 consists of an arithmetic unit, a memory, an input unit, and an output unit. The computing program is punched into a paper tape. All circuits are binary and process binary numbers. Source: Stiftung Deutsches Technikmuseum Berlin.

As programming became increasingly specialized, a growing gap emerged between professional software developers and domain scientists who needed to code. This gap created new challenges for scientific research that relied on computational methods.

1.3. Tools assisted programming, software carpentry and interdisciplinary collaboration

Recognizing the widening gulf between professional programming practices and scientific coding needs, Greg Wilson became aware of the growing problem of scientists writing code in the late 80's and subsequently formalized training (2006) focussing on teaching this audience how to implement 'good enough' practices in order to improve their writing and testing capabilities. These courses are written for the benefit of the collaborator every researcher cares about most: their future self. The goal is to make the most of time and effort spent, quoting Wilson: "Change is hard, and if researchers don't see those benefits quickly enough to justify the pain, they will almost certainly switch back to their old way of doing things."

The software industry's maturation through the 1990s transformed programming from a general skill into numerous specialized disciplines. Roles diversified into front-end developers, back-end developers, database administrators, systems architects, and quality assurance engineers, each requiring distinct expertise.

When we think about the advocacy for structured and resilient programming and its related practices, many of the principles put forward by Dijkstra are an integrated part of the Software and Data Carpentry courses. The article 'Good enough practices in scientific computing' (Wilson et al., 2017), gives detailed advice centered around the following topics:

- Data management: saving both raw and intermediate forms, documenting all steps, creating tidy data amenable to analysis.
- Software: writing, organizing, and sharing scripts and programs used in an analysis.
- Collaboration: making it easy for existing and new collaborators to understand and contribute to a project.
- Project organization: organizing the digital artifacts of a project to ease discovery and understanding.
- Tracking changes: recording how various components of your project change over time.
- Manuscripts: writing manuscripts in a way that leaves an audit trail and minimizes manual merging of conflicts.

Wilson's 'good enough' practices represent a pragmatic evolution of Dijkstra's structured programming principles, adapted specifically for domain experts rather than professional programmers.

In the meantime several articles show that efforts from the education and the scientific community as a whole, have introduced 'good practices' to pseudo-developers, including improved code sharing and reuse (Cadwallader and Hrynaszkiewicz, 2022) in several domains. Coding has become an intrinsic part of science, with significant growth registered from 2008 on (Merali, 2010) and scientists have become essential in the development of specialized software. Kim (et al., 2016), indicates 5 distinct types of working styles for data scientists:

- Insight providers, who work with engineers (or scientists) to collect the data needed to inform decisions that managers (or the field) make
- Modeling specialists, who use their machine learning expertise to build predictive models
- Platform builders, who create data platforms, balancing both engineering and data analysis concerns
- Polymaths, who do all data science activities themselves
- Team Leaders, who run teams of data scientists and spread best practices

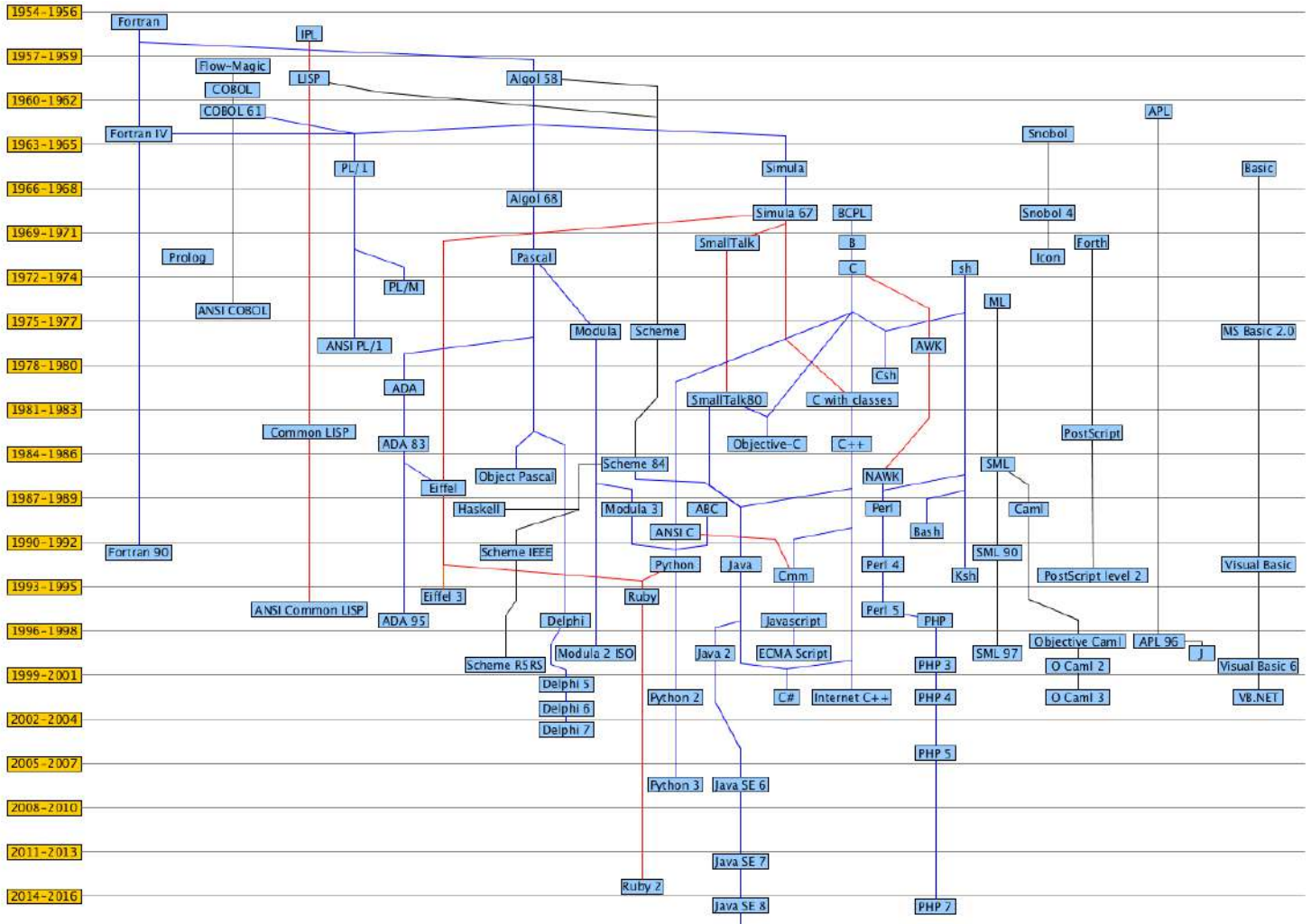
Programming during this period becomes a collaborative engineering discipline with automated testing and deployment pipelines. Scientists will find themselves predominantly in two of these groups: the 'insight providers', sharing the expertise and knowing the needs of their own field, and 'polymaths', simply using code to solve problems as they have always done.

Despite these efforts to improve scientific coding practices, recent global events have revealed how persistent the challenges remain. When Thimbleby reflects back on developments during the COVID crisis,

he notices that we find ourselves in a new crisis "the reproducibility crisis". Even when the field has developed numerous tools to help document and share code such as GitHub, Jupyter notebooks, and containerization solutions, we still find ourselves confronted with similar problems as Dijkstra, Turkey and Wilson and others have already noted from the 60s on.

Figure 7: Programming languages, relation growth and development over time

Source: Silvio Peroni, for his course Computational Thinking and Programming (A.Y. 2018/2019), Second Cycle Degree in Digital Humanities and Digital Knowledge, Creative Commons Attribution 4.0 International License



Furthermore, pseudo-developers often find themselves put under pressure, having to produce in order to answer to, often urgent, societal problems, rather than academic ones (Cadwallader and Hrynaskiewicz, 2022). With insufficient support and interdisciplinary collaboration, this can result badly documented, insufficiently tested code containing hidden methodological flaws. Bad code combined with the trust needed in scientific progress in times of crisis, informing, for example, public health decisions on a large scale, can have significant consequences far beyond academic impact.

As a result, Thimbleby repeats many of Wilson's proposed solutions but supplements them with a need for software engineering boards (SEBs), and a proposed solution in the form of an extension of to the reproducible analytic pipeline (RAP) method, to include code development in scientific workflows. (Thimbleby, 2024). The reproducibility crisis Thimbleby identifies echoes the 'software crisis' recognized at the 1968 NATO conference—both stem from the difficulty of writing correct, understandable, and verifiable computer code, though now at a larger scale and with greater societal implications.



Figure 8: A famous character from the Belgian comic book 'Jommeke' by author and cartoonist Jef Nys. Named 'Professor Gobbelijn' and self declared scientist of everything. Kind hearted but confused, he often says the opposite of what he means. Here he exclaims in blissful happiness "Fail! I mean, Success!". As the ultimate 'confused professor', this image is telling for the reputation scientists might have amongst computer scientists or developers.

Source: Jommeke, by Jef Nys.

As we enter the mid-2020s, scientists find themselves at another pivotal moment in the evolution of computing. Following Hey et al.'s (2009) framework of scientific paradigms — empirical, theoretical, computational, and data-intensive — we may now find ourselves at the height of the fourth or the emergence of a fifth paradigm: computationally enhanced science, introduced by artificial intelligence. However, if we seem so bad at learning from our mistakes, and as science and computing is trusted for large scale decision-making, are we ready to unleash a whole new generation of pseudo-developers? How will we ensure that our computational methods remain transparent and verifiable even as they become more complex and powerful?

2. Domain Experts as Developers in Cultural Heritage

2.1. The rise of computational data in Cultural Heritage Institutions

In this section, we examine the growth of computational approaches in cultural heritage institutions and the resulting proliferation of diverse datasets. These developments have created both opportunities and challenges for domain experts working with increasingly technical preservation methods. The Royal Institute for Cultural Heritage (KIK-IRPA) in Belgium serves as one illustrative example. The comparison is an interesting one, as its origins date from around the same time as early computing. The founder of the institute spoke the following words in 1950:

We are happy to live in a time when a mutual and ever growing interpretation of art history and the natural sciences is developing. [...] Physics and chemistry may thus fill an empty space, created by the essentially subjective nature of art criticism and art historical research.
(Coremans, 1950)

This quote does not hide that a solid basis for scientific research had already been laid out, including in some of our neighboring countries such as the private laboratory of Arthur Pillans Laurie (Scottish chemist that eventually settled in London, 1929), The laboratory of the British Museum (created in 1919, installed at the museum in 1932), the Institut Mainini at the Louvre Museum (created 1932), and the Scientific Department at the Courtauld Institute (1934). The Courtauld in this case, is most similar to KIK-IRPA as its mission states that they perform research on the physical constitution of works of art. They also study the influence of the physical environment. Additional areas of focus include the advantages and disadvantages of different methods of restoration and scientific investigations, and the physical, chemical and other characteristics of materials employed in the construction of works of art.

The mission of the Courtauld already indicates that the sciences involved in order to preserve cultural heritage are diverse. It is a small field of study that has benefitted early on from an interdisciplinary approach. At its initial conception in 1934, Paul Coremans (1908-1965), a chemist, was appointed as head of the photographic service and laboratory of the Royal Museums. His first studies would be about humidity control in museums, research that would later be considered within the domain of preventive conservation.

These laboratories were located in the same building complex where the Mundaneum was housed (Vanpaemel, 2018).

In 1946, the interdisciplinary institute was created. After a slight reorganization and a move to a brand new building, KIK-IRPA got the shape and organization of what it is today. A research institution, without an art collection — with the exception of documentation and scientific samples — centered around three departments all tied to different scientific disciplines: physical and microchemical laboratory with scientists and engineers, the photographic archives with historians, art historians, archaeologists and scientific imaging experts, and a conservation department with conservators (Deelstra en Thorburn, 2018). By 1970, scientific development of the field was moving at full speed and documentation by photographs besides textual information was integrated in the way of working. In addition, we start to see larger data collection, registration and analysis efforts and, at a later stage, digitization efforts. While KIK-IRPA provides one example of this evolution, similar transformations were occurring across the cultural heritage sector globally. Let's examine these broader trends in digitization and data management.

From all the galleries, libraries, archives and museums (GLAM) institutions, libraries and archives were leading the way in the digitization efforts. Their function as an institution and their collections are more homogeneous than museums or galleries. From those early documentation systems grew the collection documentation/registration systems. Early systems were inventory numbers with related textual metadata that was searchable. Gradually, the field professionalized with allowing for more data formats to be integrated or linked, and the development of international standards and thesauri.

A pioneering classification system is ICONCLASS, which originated in the Netherlands in the 1950s. ICONCLASS defines codes that are linked to subjects to enable very precise identification of iconographic content across language barriers. It is therefore still similar to UDC and like UDC, still in use. Other early developments originate predominantly in-house or regionally and are published worldwide at a much later stage. The Art and Architecture Thesaurus was initially developed in house at the J. Paul Getty Trust in the late 70s and was only published in 1990, gradually becoming multilingual adding e.g. Spanish (1996), Dutch (2000) and Chinese (2008). Other influential standards and thesauri are: SPECTRUM (1994, UK), CIDOC Conceptual Reference Model (1996, international, ISO 21127:2023), Gemeinsame Normdatei (2012, Germany). This led to the creation of national and international platforms that share our cultural heritage. Noteworthy initiatives are Europeana (2008, EU wide) and Japan Search (2020).

While documentation benefited early from computation and digitization — leading to the development of specialties like information managers or digital archivists and registrars — other specialties within cultural heritage have followed different trajectories. Publication trends clearly demonstrate this evolution, with DCH (Digital Cultural Heritage) research showing three distinct phases: an embryonic stage (1997-2006) with minimal publications, a rising stage (2007-2016) marked by conceptual development, and a prosperity stage (since 2017) with exponential growth in research output (Lian and Xie, 2024). This evolution reflects both increasing specialization within cultural heritage disciplines and growing interconnectedness between them through digital technologies.

Contemporary research in digital cultural heritage focuses on three primary areas: technological innovation through VR/AR and interactive digital storytelling; information management through specialized databases and digital archives; and digitization methodologies for cultural heritage preservation (Lian and Xie, 2024). These areas align with the specialized data domains that have emerged the historical developments of different institutional practices.

The emergence of specialized data domains in cultural heritage has fundamentally transformed preservation practices. These domains solve critical problems by enabling non-invasive analysis of fragile artifacts,

Specialists produce both standard documentation photography and analytical imaging using various parts of the electromagnetic spectrum (X-rays, infrared, UV fluorescence). These visual datasets range from high-resolution documentary photographs to specialized technical visualizations that reveal hidden information about object construction and condition invisible to the naked eye. At KIK-IRPA photographic campaigns, scientific imaging and digitization are different units that are part of the documentation department.

- Spatial Data - role: **spatial documentation specialists** such as building engineers, architects, archeologists, urban and sites specialists and conservators.
These professionals create vectorial representations of physical objects and environments through 2D drawings, 3D models, and spatial datasets. Rather than capturing visual appearance, these more abstract, structured interpretations preserve dimensional and relational properties, enabling measurement, analysis, and virtual reconstruction. Formats include CAD drawings, photogrammetric models, point clouds, and GIS datasets that document everything from small artifacts to architectural complexes. At KIK-IRPA this information is seldom created but the monuments lab (a unit within the laboratories) would be the main user of these type of datasets and able to create or enhance them due to their relevant backgrounds.
- Environmental data - role: **environmental monitoring specialists** such as physicists, preventive conservators and building engineers.
This domain centers on measurements of natural phenomena that affect heritage preservation. Using sensor networks and monitoring systems, these specialists collect time-series data on temperature, humidity, light exposure, and other environmental variables. These continuous datasets provide insights into preservation conditions, detect potential risks to collections, and inform preventive conservation strategies, often utilizing computational approaches to analyze large volumes of environmental measurements. At KIK-IRPA this information is mostly gathered and analyzed by the monuments lab (weather and building facade) and preventive conservation unit (indoor end microclimates, part of the conservation department).
- Material analysis data - role: **analytical science specialists** such as chemists, physicist, and biologist.
Representing the most abstract level of analysis, this domain focuses on laboratory-generated data revealing the fundamental composition, structure, and condition of cultural materials. Using various spectroscopic, chromatographic, and microscopic techniques, these specialists produce datasets that characterize materials at molecular and microstructural levels. These analyses support authentication, provenance studies, degradation monitoring, and treatment planning by translating physical science methodologies into cultural and historical insights. At KIK-IRPA this type of data is mostly produced and analyzed within the laboratories with subsections often related to the use of specific materials (paper/parchment, painting, textiles, wall painting, polychomy, glass, metal and monuments/stone) and specific dating techniques (C14 dating, dendrochonology).

As all scientific fields in cultural heritage evolve, there is a distinct development of hybrid techniques that intentionally bridge multiple data categories, requiring interdisciplinary expertise and specialized computational approaches. Two notable examples are macro X-ray fluorescence (MA-XRF) scanning, which merges analytical chemistry with imaging science to create element distribution maps that are simultaneously visual records and chemical analyses, and 3D documentation with integrated analytical data, which combines volumetric spatial recording with material analysis by mapping spectral information directly onto geometric models of cultural objects.



Figure 10: Painted image of a cross section of the of the cloak of St. John, part of the Ghent Altarpiece, 2000x enlarged. Source: KIK-IRPA's scientific collection. Low-tech communication on scientific analysis of a material-technical art composition. Date: suspected to be from the 1950s or 1960s. CC BY 4.0 KIK-IRPA, Brussels (Belgium), cliché X165583

Figure 11 and 12 (below): The use of photogrammetry at the Yungang Grottoes starting in 1986 (left) leading to a 3D-printed replica (right) of Yungang Grotto 18 installed at Beijing University of Civil Engineering and Architecture, November 2018. Data-intens and high tech techniques to share cultural heritage replicas. These 3D capture techniques can also be used as records in case of destruction by natural disaster or confluct. Copyright: The Yungang Grottoes Research Academy.



It is interesting to link these profiles to the specific domains put forward by (Cadwallader and Hrynaszkiewicz, 2022), we can see different profiles emerge, some examples:

- Insight providers → documentation specialists, registrars, and conservators who analyze collection data and translate domain expertise into actionable preservation strategies
- Modeling specialists → conservation scientists and preventive conservators who develop predictive models for material aging, environmental impact assessment, and risk analysis
- Platform builders → digital asset managers and database administrators, who design and implement systems for managing diverse heritage data types
- Polymaths → with different backgrounds but primarily found in research or academic institutions working in highly interdisciplinary domains such as preventive conservation, digitization methodology development, and emerging analytical techniques that bridge multiple knowledge domains
- Team leaders → with different backgrounds but often principal investigators (PIs) of nationwide or international interdisciplinary research campaigns that coordinate specialized knowledge across institutional and disciplinary boundaries to solve complex heritage challenges

In the following sections, we will return to examine the specific role of pseudo-developers at KIK-IRPA. As datasets grow in size and complexity, and as cultural heritage institutions increasingly rely on computational methods for preservation decisions, the need for sound coding practices becomes more important. The professional evolution proposed by Wilson's Software Carpentry and similar initiatives remains as relevant for cultural heritage professionals as for other scientific domains. While some standards and best practices have been established in the field, the challenge lies in knowledge transfer and consistent adherence to these standards.

What distinguishes cultural heritage from many other scientific domains are the time horizons used and the limited size of the field of study. The objects and information being preserved are intended to last well beyond the working life of today's conservation scientists — potentially for centuries, as the Mundaneum example illustrates. This creates unique considerations for pseudo-developers in heritage fields:

- The balance between developing shareable software versus individual code solutions becomes crucial. Limited resources often force heritage professionals to make pragmatic choices about when to invest in software development versus simpler coding solutions. The question is if reusability needs to always be the end goal.
- Unlike in many scientific fields where data may become obsolete as theories advance, cultural heritage data has to maintain its value over time. The historical can remain relevant for centuries, making sustainable data practices particularly critical.

These considerations highlight why cultural heritage institutions must carefully navigate the role of pseudo-developers, balancing immediate practical needs with long-term data stewardship responsibilities.

2.2. KIK-IRPA's domain experts as developers

"Not too bad for a researcher"

"It's messy, but it works"

"Did you set this up with ChatGPT?"

(NN, SB, and SG when asked what an outsider says when looking at your code?)

Based on former research into coding practices and code development (Cosaert and Beltran, 2022), and since the introduction of generative AI within the field, it seemed opportune to launch a small questionnaire to collect data on various coding practices in-house. The goal of this questionnaire was threefold: understand coding practices and profiles at KIK-IRPA in order to understand internal needs and streamline

processes, to establish international collaboration lifting us from our siloes (for some) and to be better informed and establish priorities when applying for future research opportunities.

KIK-IRPA consists of 120 staff, with about 75% working in research. A small but growing subset of researchers use coding, with seven respondents actively integrating it into their workflows. From the respondents, one works in documentation (scientific imaging - NN), three at the laboratories (two from the building department - RH and SG - and one from dendrochronology - CP) and three from supportive services such as the sustainability unit (AC - before under the department of conservation and restoration, preventive conservation) and research data management unit (WF and SB - before part of the documentation department). Participants have diverse backgrounds and all hold master or doctoral degrees: conservator (AC), archaeologist (CP), computer science (NN), civil engineer (RH), archivist (SB), conservator (SG) and chemist (WF). This confirms the interdisciplinary nature of the cultural heritage field and aligns with the historical pattern seen in both early computing pioneers and the resurgence of domain experts adopting programming from the 1980s onward.

The questionnaire employed a mixed-methods approach, combining multiple-choice questions, short answers (limited to 50 words), and extended responses (up to 200 words) across eight structured sections covering personal backgrounds, project descriptions, technical skills, collaboration methods, tools used, future perspectives, training needs, and sustainability practices. While most coders at KIK-IRPA (that are not part of the IT team), have answered these questions, this is only a first attempt to spot some trends in a diversified environment. This is not intended as a broad analytical study.

The questionnaire recorded that the dominant languages are Python for various tasks, for which most are advanced users, and JavaScript for either simple scripts or web tools. Knowledge of SQL is intermediate, but critical for some database tasks. More niche languages are MATLAB and R, which are mainly used for analytical tasks. Other known languages are always known by one individual: C, C++, C#, Julia, TypeScript, React and FastAPI.

"Since the beginning of the discipline, we have collected extensive data in dendrochronology. Coding familiarity is important because our science depends on data acquisition tied to precise references." (CP)

The predominant types of data used are collection management data, archival or documentation data, and chemical and physical analysis data. Environmental and image-related data are slightly less common. Usage of standards and basic data analysis principles are present overall with a high reference to FAIR principles and a more limited use of CIDOC CRM, IIF and Dublin Core. While known, they are not always consciously applied. Code sharing and collaboration are considered important by all, but implemented regularly by a lower number of respondents with GitHub and GitLab being used most.

"Few people in the field can code. They use different languages, frameworks or stacks. Most are self-taught and cannot easily switch between them." (WF)

There are two types of developers or coders that emerge from this questionnaire, namely the research data managers (RDM) and domain-specific coders (DSC), with anyone being able to adhere to one or both profiles:

- The RDMs (SB, NN and WF. RH could also fit within this profile when considering language knowledge, but is excluded on the basis of practice since he rarely shares code and collaborates) master multiple languages, tools and standards and have enjoyed formal training in computer science related fields (either as part of their initial training or in the form of a post-master degree). They know the requirements to adhere to software engineering practices and can apply standards and use

advanced tools if required. They can build or contribute to large(r) scale applications, databases or infrastructure. They can focus on technical architecture, interoperability and long term sustainability, if required. They are aligned with the previously defined roles of information specialists (in CH) having a working style comparable to team leaders and/or platform builders (Kim et al., 2016).

- The DSCs (predominantly AC, CP, SG, NN in his current role, and RH) focus on solving domain-specific problems and automating repetitive tasks. They have intermediate skills in one or multiple scripting languages and use limited version control. They often use coding as an extension to basic tools and use generative AI for specific tasks. While they are certainly data-savvy and document their practices, this might be in less established or formal ways. They focus on efficiency and immediate utility over long-term maintainability. They are aligned with the previously defined roles of scientific imaging analyst, environmental monitoring specialists and analytical science specialists (in CH) having a working style comparable to insight providers, polymaths and modeling specialists (Kim et al., 2016).

There is a key difference between the preservation of code directly or indirectly documenting heritage itself, and code used for data analysis within the field. The challenge isn't just coding; it's designing systems that balance the pragmatism of pseudo-developers with the rigor of software engineering.

"Building efficient, fault-proof code is not always easy." (RH)

RDMs practices are aligned with the more classic tasks of the documentalist. Their task is to preserve and share information itself, just as at the Mundaneum, technological changes just being part of the evolution of time. Their focus is data management and interoperability. While most of the RDMs interviewed do not consider themselves developers, they might have become too proficient to still be considered pseudo-developers. For the DSCs, code preservation itself is secondary, since the information that will be saved for future purposes is the results they obtain through their work. Their focus is analysis, automation and visualization. DSCs are, in a sense, better aligned with the earlier definition of a pseudo-developer.

A significant influence on the DSCs work is the niche in which they work. The field of CH is small in itself with the interviewees often working on a data heavy niche within this field. This can lead to siloed workflows and use of AI as a collaborator for testing, debugging and writing code. In contrast, RDMs use AI strategically, to enhance existing engineering practices (debugging, docs) and tackle interoperability challenges.

"More people will learn programming, similar to how professions adopted computers 50 years ago or the Internet 20 years ago. Much of the code will be imperfect, but I don't think that's a major issue." (NN)

The participants' views on their role reveal a nuanced tension between pragmatism and professionalism. There are shared concerns about reproducibility. Both groups highlight the lack of time and training to adhere to software engineering standards, yet their responses diverge in ambition: RDMs advocate for structural changes (e.g., integrated coding support), while DSCs lean on AI and peer networks as stopgaps. Even as AI democratizes coding access, participants like WF and SG warn it may exacerbate the very gaps pseudo-developers seek to close — between quick fixes and enduring solutions. Their perspectives crystallize a central dilemma in heritage informatics: how to balance domain expertise's urgency with computing's demand for rigor.

"In research, we should focus less on writing reusable code and more on 'rewritable' code: code that is easy to understand, copy, and adapt to one's needs." (NN)

"How to keep the code sustainable? Short term: proper documentation, KISS (Keep It Simple, Stupid) principles and active software maintenance. Long term: programming and AI evolution renders code obsolete quickly. Should we try to keep up, or just let software come and go?" (WF)

The question of the need for reproducibility is a valid one. If code is pragmatic and temporary or simple and well documented, is a use for standardization required? The goal of DSCs is often not to create software, but rather to either use their code themselves, with a small team of colleagues with similar expertise or to be an adviser for software development, focusing on bringing technical expertise to the table.

Most participants see training as at least part of the solution with a large interest for AI and ML principles. Some consider formal training about specific subjects, usually to help optimize their workflow. Communal software licenses (including AI tools such as Github Copilot, Chat GPT and Claude) stronger unified institutional practices and, last but not least, better computers, could also be part of the solution.

When asked if KIK-IRPA should take the lead on any of these topics, SG answered: "Yes, because leading these efforts drives innovation, fosters collaboration, and helps us shape the future of technology and best practices."

The patterns observed at KIK-IRPA mirror the broader challenges faced by scientific pseudo-developers across disciplines. Like their predecessors in the software crisis of the 1960s and the reproducibility crisis highlighted during COVID-19, cultural heritage coders struggle with balancing immediacy against sustainability. The decades old question remains valid: can institutions foster a culture where "good enough" code evolves into "sustainable enough" systems, without stifling the ingenuity of domain experts?

AI-generated code may further obscure the distinction between understanding and implementation, potentially creating a new class of "pseudo-pseudo-developers" who rely on AI without grasping the underlying principles. This tension between accessibility and depth, forms the backdrop for the next chapter's exploration of how AI is reshaping the coding landscape in cultural heritage preservation.

3. AI: Democratization, Disruption, or Déjà Vu

Science has often pioneered transformative paradigm shifts throughout history, initiating revolutions that eventually transcend academic boundaries to reshape commerce and industry. When these innovations become mainstream, scientific communities must adapt to new realities they helped create. The current AI developments follow this familiar pattern—scientific breakthroughs becoming commercialized tools that redefine the landscapes they emerged from.

Whether AI represents a fundamental revolution for cultural heritage or merely another evolutionary step remains uncertain at this early stage. While speculation about its ultimate impact would be premature, we can draw valuable lessons from previous technological transitions. By examining how related fields navigated similar transformations, cultural heritage professionals can prepare for the challenges and opportunities that lie ahead, regardless of AI's eventual magnitude of impact.

Fields like data science and software development serve as critical testing grounds for AI's impact on coding, as practitioners both use and evaluate these tools firsthand. Some studies explore how AI enables non-programmers to engage with coding within their own domains (Bien & Mukherjee, 2025), while others view AI as a pervasive collaborator reshaping professional workflows (Wang et al., 2019). At the same time, research also highlights significant challenges, particularly regarding usability and trust (Liang et al., 2024).

Technical assessments consistently point to key weaknesses in generative AI, including low code quality, poor maintainability, debugging difficulties, lack of transparency, and failure to meet functional requirements. These limitations raise concerns about AI-generated code's reliability and long-term viability.

Education is often the first to grapple with the consequences of AI misuse. While some research examines its impact on novice learners under 17 (Kazemitabaar et al., 2023), others focus on how university instructors are responding to AI code generators like ChatGPT and GitHub Copilot (Lau & Guo, 2023). These studies reveal a divided landscape: AI tools present both opportunities and risks in education. While they can enhance learning and improve accessibility, prior coding experience remains highly advantageous. Additionally, concerns about over-reliance on AI, academic integrity, and learning effectiveness continue to fuel debate over the role of AI in programming education.

Will generative AI fundamentally alter the way we approach scientific inquiry? Are we adding yet another layer of abstraction, moving further away from empirical observation, the foundation of many disciplines? As we increasingly rely on AI-generated insights, there is a risk of prioritizing data over direct observation, forgetting that structured, methodical examination remains one of the most powerful tools in fields like conservation.

At the same time, the challenges AI presents are not new. Issues like poor documentation, lack of reproducibility, and weak long-term maintenance have long plagued scientific and computational practices. The question is not whether AI will introduce these problems — it already has — but whether we will finally learn to address them systematically.

The answer is uncertain. AI will reshape scientific workflows, introducing new efficiencies and complexities alike. But if history is any indication, progress rarely comes without friction. What matters is whether we recognize these challenges early and adapt accordingly.

Conclusion: weaving threads of past, present, and future practice

Throughout this exploration of pseudo-developers in cultural heritage, we have traced a path from Otlet's visionary Mundaneum to today's AI-assisted programming environments. This journey reveals striking parallels between historical information management challenges and current computational dilemmas facing our field. Just as the Mundaneum's creators sought to organize knowledge through innovative systems despite lacking formal training in information science, today's cultural heritage professionals adapt digital tools to preserve our collective memory without necessarily possessing software engineering expertise.

The questionnaire responses from KIK-IRPA staff reveal the dual nature of coding in our field, as both a pragmatic necessity and an evolving professional practice. The tension between Research Data Managers' structured approaches and Domain-Specific Coders' pragmatic solutions mirrors broader debates about code quality, sustainability, and reproducibility that have persisted since the earliest days of computing. This tension is not merely technical but reflects fundamental questions about how we balance short-term needs with long-term preservation goals in a field explicitly dedicated to multi-generational timescales.

Collaboration as the Bridge Between Expertise and Innovation

Our findings suggest that the path forward lies not in converting all heritage professionals into software engineers, nor in relegating coding exclusively to technical specialists, but rather in fostering meaningful collaboration across disciplinary boundaries. The challenge is to create environments where domain experts and technical specialists work in tandem, each contributing their essential perspective.

As WF noted, "Few people in the field can code. They use different languages, frameworks or stacks." This diversity of approaches represents both a challenge and an opportunity. By establishing cross-institutional communities of practice specifically for heritage coders, similar to Software Carpentry but tailored to our field's unique preservation mandates, we can accelerate knowledge transfer while preserving the innovative problem-solving that emerges from domain-specific expertise.

Practical approaches might include:

- Establishing regular code review sessions between RDMs and DSCs within and across institutions
- Creating mentorship programs pairing technically proficient staff with domain experts
- Developing shared documentation standards that accommodate both technical rigor and domain-specific knowledge
- Implementing institutional recognition for code contributions to heritage projects
- Contributing to the creation of packages and standards in order to facilitate better collaboration with developers and efficiently fulfill our role as insight providers.

Institutional Evolution in a Computational Landscape

For institutions like KIK-IRPA and similar heritage organizations worldwide, navigating the computational turn requires structural evolution. As SG suggested, "Leading these efforts drives innovation, fosters collaboration, and helps us shape the future of technology and best practices." Taking leadership in computational heritage practice means reconsidering traditional departmental structures, budget allocations, skill development pathways, and professional recognition systems.

The institutional implications extend beyond technical infrastructure to encompass:

- Creating hybrid roles that bridge traditional disciplinary boundaries
- Developing evaluation metrics that value both preservation outcomes and computational contributions
- Establishing sustainable funding models for code maintenance
- Balancing investment between commercial tools and in-house development
- Building institutional memory around computational decisions

The Mundaneum's ultimate fate—partial loss followed by renewed appreciation—serves as both cautionary tale and inspiration. Cultural heritage institutions must consider not only the preservation of collection data but also the preservation of the computational methods that generate and analyze that data. This "meta-preservation" challenge requires institutional commitment to both technical infrastructure and knowledge transfer across generations of staff.

New Research Avenues in a Time of Transition

As cultural heritage computing evolves alongside AI developments, several promising research directions emerge at this intersection:

First, we need empirical studies comparing preservation outcomes from different coding approaches. Does "good enough" code as described by Wilson truly serve preservation goals, or do more rigorous engineering practices yield measurably better results in our context? Such comparative research could establish evidence-based best practices specifically for heritage computing.

Second, the tension between reusability and "rewritability" highlighted by NN deserves deeper investigation. In a field where some analyses might be performed only once on unique artifacts, while

others become routine practice across collections, understanding when to invest in robust, reusable systems versus focused, specialized tools represents a critical research question.

Third, the integration of AI into cultural heritage coding practices merits systematic study. Beyond general concerns about AI-generated code quality, specific research on how these tools function within heritage contexts could identify both unique risks (such as perpetuating disciplinary biases) and opportunities (such as making computational approaches more accessible across diverse institutional settings).

Finally, the ethical dimensions of code development in heritage preservation demand attention. As our field increasingly relies on algorithms for collection management, conservation decision-making, and access provision, questions of transparency, accountability, and sustainability become central to our preservation mandate.

The Continuing Journey

We began with Otlet's prescient observation: "Mankind is at a turning point in its history. The mass of data acquired is astounding. We need new instruments to simplify it, to condense it, or intelligence will never be able to overcome the difficulties imposed upon it or achieve the progress that it foresees and to which it aspires." A century later, our challenge remains remarkably similar, though our technical capabilities have transformed dramatically. The pseudo-developer—that domain expert who bridges disciplinary and technical worlds—continues to play a vital role in heritage preservation, just as the early pioneers of the Mundaneum did in their time.



We must optimize our efforts in a fast-paced and changing world, preserving for eternity, efficiently. This optimization requires not only technical skill but also institutional wisdom and collaborative spirit. As we navigate the AI age, we would do well to remember that our computational methods, like our collections themselves, represent cultural heritage in their own right, records of how we understood, valued, and preserved our collective memory at this moment in history. The code we write today becomes part of tomorrow's heritage. Let us create it with care, document it with clarity, and share it with generosity.

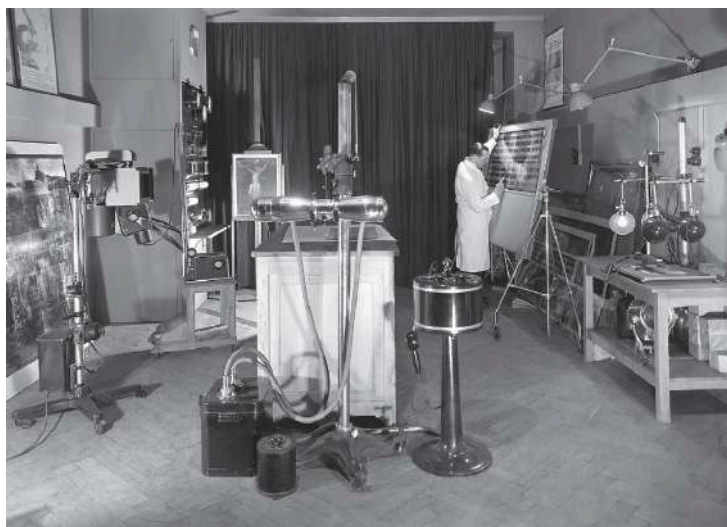


Figure 13 and 14: (above) The set-up of the laboratories of the Museums of Art and History in 1939 and the physical laboratories in that same building in 1947 (below). The documentation of scientific research methods to preserve works of art has always been a part of the institution's mission where systematic knowledge transition has proven to be key. CC BY 4.0 KIK-IRPA, Brussels (Belgium)

References

- Abbate, Janet. *Recoding Gender: Women's Changing Participation in Computing*. Mit Press, 2012. <https://doi.org/10.7551/mitpress/9014.001.0001>.
- Bien, Jacob, and Gourab Mukherjee. 'Generative AI for Data Science 101: Coding Without Learning to Code'. *Journal of Statistics and Data Science Education*, 27 January 2025, 1–14. <https://doi.org/10.1080/26939169.2024.2432397>.
- Cadwallader, Lauren, and Iain Hrynaszkiewicz. 'A Survey of Researchers' Code Sharing and Code Reuse Practices, and Assessment of Interactive Notebook Prototypes'. *PeerJ* 10 (22 August 2022): e13933. <https://doi.org/10.7717/peerj.13933>.
- Ceruzzi, Paul E. *A History of Modern Computing*. 2nd. Ed. Massachusetts, U.S.A.: MIT press, 2003.
- Cosaert, Annelies and Beltran, Vincent L., eds. *Tools for the Analysis of Collection Environments: Lessons Learned and Future Development*. Los Angeles: Getty Conservation Institute, 2022. https://www.getty.edu/conservation/publications_resources/pdf_publications/pdf/tools_for_analysis.pdf.
- Dahl, Ole-Johan, Edsger W. Dijkstra, Charles A. R. Hoare, and Charles A. R. Hoare. *Structured Programming*. 11. print. APIC Studies in Data Processing 8. London: Academic Press, 1972.
- Deelstra, Hendrik, and Duncan Thorburn Burns. 'Paul Coremans (1908-1965): A Pioneer Chemist in the Application of Scientific Techniques to the Visual Arts'. In *A Man of Vision: Paul Coremans and the Preservation of Chemical Heritage Worldwide*, edited by Dominique Deneffe and Dominique Vanwijnsberghe, 26–38. Scientia Artis 15. Brussels: Royal Institute for Cultural Heritage (KIK-IRPA), 2018. <https://orfeo.belnet.be/handle/internal/9876>.
- Dijkstra, E. W. 'Go To Statement Considered Harmful'. In *Edsger Wybe Dijkstra*, edited by Krzysztof R. Apt and Tony Hoare, 1st ed., 315–18. New York, NY, USA: ACM, 1968. <https://doi.org/10.1145/3544585.3544604>.
- Hey, Tony, Kristin Tolle, and Steward Tansley, eds. *The Fourth Paradigm – Data-Intensive Scientific Discovery*. Communications in Computer and Information Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. https://doi.org/10.1007/978-3-642-33299-9_1.
- Lau, Sam, and Philip Guo. 'From "Ban It Till We Understand It" to "Resistance Is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools Such as ChatGPT and GitHub Copilot'. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V.1*, 106–21. Chicago IL USA: ACM, 2023. <https://doi.org/10.1145/3568813.3600138>.
- Lian, Yuntao, and Jiafeng Xie. 'The Evolution of Digital Cultural Heritage Research: Identifying Key Trends, Hotspots, and Challenges through Bibliometric Analysis'. *Sustainability* 16, no. 16 (20 August 2024): 7125. <https://doi.org/10.3390/su16167125>.
- Liang, Jenny T., Chenyang Yang, and Brad A. Myers. 'A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges'. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 1–13. Lisbon Portugal: ACM, 2024. <https://doi.org/10.1145/3597503.3608128>.
- Kazemitabaar, Majeed, Justin Chow, Carl Ka To Ma, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 'Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming'. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–23. Hamburg Germany: ACM, 2023. <https://doi.org/10.1145/3544548.3580919>.
- Kim, Miryung, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 'The Emerging Role of Data Scientists on Software Development Teams'. In *Proceedings of the 38th International Conference on Software Engineering*, 96–107. Austin Texas: ACM, 2016. <https://doi.org/10.1145/2884781.2884783>.
- Merali, Zeeya. 'Computational Science: ...Error'. *Nature* 467, no. 7317 (October 2010): 775–77. <https://doi.org/10.1038/467775a>.
- Olet, Paul. *Traité de documentation: le livre sur le livre, théorie et pratique*. Edited by Van Keerberghen et Fils. 1st ed. Brussels: Editions Mundaneum, Palais Mondial, 1934. https://libstore.ugent.be/fulltxt/BIB-038A006_2006_0001_AC.pdf.
- Thimbleby, Harold. 'Improving Science That Uses Code'. *The Computer Journal* 67, no. 4 (21 April 2024): 1381–1404. <https://doi.org/10.1093/comjnl/bxad067>.
- Tukey, John W. 'The Future of Data Analysis'. In *Breakthroughs in Statistics: Methodology and Distribution*, 408–52. Springer, 1962.
- Vanpaemel, Geert. 'Paul Coremans (1908-1965): A Pioneer Chemist in the Application of Scientific Techniques to the Visual Arts'. In *Early Museum Laboratories and the Pursuit of Objectivity*, edited by Dominique Deneffe and Dominique Vanwijnsberghe, 14–25. Scientia Artis 15. Brussels: Royal Institute for Cultural Heritage (KIK-IRPA), 2018. <https://orfeo.belnet.be/handle/internal/9876>.
- Wang, Dakuo, Justin D. Weisz, Michael Muller, Parikshit Ram, Werner Geyer, Casey Dugan, Yla Tausczik, Horst Samulowitz, and Alexander Gray. 'Human-AI Collaboration in Data Science: Exploring Data Scientists' Perceptions of

Automated AI'. *Proceedings of the ACM on Human-Computer Interaction* 3, no. CSCW (7 November 2019): 1–24. <https://doi.org/10.1145/3359313>.

- Wilson, Greg, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K. Teal. 'Good Enough Practices in Scientific Computing'. Edited by Francis Ouellette. *PLoS Computational Biology* 13, no. 6 (22 June 2017): e1005510. <https://doi.org/10.1371/journal.pcbi.1005510>.

KIK-IRPA project list — (possibly) in collaboration with other institutions

- **Advanced Ion Data Analysis and Moisture Content Tools:** Web-based application for processing salt/moisture data in heritage conservation. Link: [GitHub \(placeholder\)](#)
- **BALaT+ Frontend:** Redesigned collection portal for accessing KIK-IRPA's heritage documentation. Link: *Under development*
- **C14 Public Database:** Search portal for radiocarbon dating data from MICADAS and legacy datasets. Link: c14public.kikirpa.be
- **CASIMODO:** Analyzes medieval wood frameworks to study forest/climate dynamics. Link: [ANR Project](#)
- **CEAM (Climate and Energy Assessment for Museums):** Tool for analyzing energy savings in heritage spaces. Link: [Climate2Preserv \(under development\)](#)
- **ConSciR:** R package for conservation calculations and workflow automation. Link: [GitHub](#)
- **E-RIHS Knowledge Base:** Repository for FAIR heritage science data and tools. Link: [GitHub](#)
- **Folder Tool:** Organizes digital folder structures for institutional file transfers. Link: folder-tool.kikirpa.be (*internal use*)
- **GIANT.jl:** Open-source Julia library for nonnegative matrix factorization algorithms. Link: [GitLab](#)
- **Migration Test Scripts:** Python scripts for validating data migration between CMS platforms. Link: *Internal use*
- **PREDICT:** Suite of tools for heritage conservation analytics (details unspecified). Link: <https://predict.kikirpa.be/> (*Under development*)
- **Smoothed Separable NMF:** Algorithms and datasets for reproducing research experiments. Link: [GitLab](#)
- **SOPRANO:** Spectral library for synthetic organic pigment analysis. Link: soprano.kikirpa.be
- **Visualize Climate Data – RH Threshold:** Web tool for visualizing relative humidity risks in heritage spaces. Link: [GitHub \(placeholder\)](#)

KIK-IRPA contributors, contact information

- **Clara Penagos**, Dendrochronology lab, Laboratory department
Email: clara.penagos@kikirpa.be, Orcid: <https://orcid.org/0000-0001-7962-9415>
- **Wim Fremout**, Research data management Unit
Email: wim.fremout@kikirpa.be, Orcid: <https://orcid.org/0000-0002-1684-377X>
- **Nicolas Nadisic**, FED-tWIN Professor-Researcher (UGent/KIK-IRPA)
Email: nicolas.nadisic@ugent.be, Orcid: <https://orcid.org/0000-0002-5993-7194>
- **Roald Hayen**, Monuments lab, Laboratories department
Email: roald.hayen@kikirpa.be
- **Sebastiaan Godts**, Monuments lab, Laboratories Department
Email: sebastiaan.godts@kikirpa.be, Orcid: <https://orcid.org/0000-0003-2189-2995>
- **Stephanie Buyle**, Research data management Unit
Email: stephanie.buyle@kikirpa.be, Orcid: <https://orcid.org/0000-0002-1481-8749>

Glossary

A

Artificial intelligence (AI): A field of computer science focused on creating systems that can perform tasks that typically require human intelligence. In the context of programming, AI systems like large language models can generate code based on natural language descriptions.

ALGOL: A family of programming languages developed in the late 1950s as a collaborative European effort. It introduced structured programming concepts and influenced many subsequent languages like Pascal and C.

API (Application Programming Interface): A set of definitions and protocols for building and integrating application software. APIs enable different software components to communicate, creating modular systems similar to Otlet's vision of interconnected knowledge.

Association for Computing Machinery (ACM): Founded in 1947, it is the world's largest scientific and educational computing society. The ACM played a crucial role in formalizing software engineering as a discipline through publications and conferences.

C

C: A general-purpose programming language developed by Dennis Ritchie at Bell Labs between 1969-1973. It provides low-level access to memory and minimal runtime features, making it suitable for systems programming.

C++: An extension of the C programming language developed in 1985 that added object-oriented features. It became widely used for systems programming while offering higher-level abstractions than C.

CIDOC Conceptual Reference Model (CRM): An international standard (ISO 21127:2023) for cultural heritage information, providing a framework for integrating, mediating, and exchanging heterogeneous cultural heritage information.

COBOL (Common Business-Oriented Language): A programming language developed in 1959 for business use. It was designed with English-like syntax to make programming more accessible to non-specialists in computing.

Collection documentation/registration systems: Digital systems used to catalog and manage cultural collections. These evolved from early inventory systems to complex databases incorporating various data formats and international standards.

Cultural heritage (CH): The legacy of physical artifacts and intangible attributes inherited from past generations, maintained in the present and preserved for future generations. In the digital context, it encompasses both physical objects and their digital representations.

D

Data management: The practice of collecting, keeping, and processing data securely and efficiently. In scientific computing, this includes saving raw and intermediate forms, documenting processing steps, and creating data that is amenable to analysis.

Data science: An interdisciplinary field using scientific methods, processes, and systems to extract knowledge or insights from structured and unstructured data. It combines elements of statistics, computer science, and domain expertise.

Developer: A professional who has formal education or extensive training in computer science or programming. They possess deep understanding of software architecture and development principles, enabling them to build complex systems from scratch.

Dewey Decimal Classification (DDC): A widely used library classification system first published in 1876 by Melvil Dewey. It organizes knowledge into ten main classes with decimal subdivisions, influencing later systems like UDC.

Digital Cultural Heritage (DCH): The use of digital technologies to document, preserve, study and present cultural heritage. As shown in the article, DCH research has evolved through three phases: embryonic (1997-2006), rising (2007-2016), and prosperity (since 2017).

Domain Specific Coders (DSC): Cultural heritage professionals who use coding to solve specific problems in their field, focusing on efficiency and immediate utility rather than long-term maintainability. They typically have intermediate skills in one or multiple scripting languages.

E

ENIAC (Electronic Numerical Integrator and Computer): The first programmable, electronic, general-purpose digital computer, completed in 1945. Women mathematicians programmed ENIAC using physical switches and cables to solve complex equations.

Environmental data: Measurements of natural phenomena affecting heritage preservation, collected through sensor networks and monitoring systems. This includes time-series data on temperature, humidity, light exposure, and other variables.

Europeana: A digital platform launched in 2008 that provides access to millions of books, paintings, films, museum objects, and archival records across Europe, representing large-scale efforts to digitize and share cultural heritage.

F

FAIR principles: Guidelines for scientific data management and stewardship that make data Findable, Accessible, Interoperable, and Reusable. These principles have become increasingly important in cultural heritage data management.

FORTRAN (Formula Translation): One of the oldest high-level programming languages, developed by IBM in 1957. It was designed primarily for scientific and engineering calculations with a focus on numerical computation.

G

Gemeinsame Normdatei: A German authority file established in 2012 to standardize the cataloging of persons, corporate bodies, conferences, geographic information, topics, and works in cultural heritage institutions.

GLAM institutions: Galleries, Libraries, Archives, and Museums - the primary organizations responsible for preserving and providing access to cultural heritage.

GoTo Statement: A programming language instruction that causes execution to jump to another part of the program. Dijkstra's 1968 letter arguing against its use helped establish structured programming principles.

H

High-level programming languages: Programming languages that abstract away machine details, allowing programmers to use more natural language constructs. They make programming more accessible by hiding hardware complexities behind easier syntax.

I

ICONCLASS: A classification system originating in the Netherlands in the 1950s that defines codes linked to subjects, enabling precise identification of iconographic content across language barriers.

IIIF (International Image Interoperability Framework): A set of standard APIs for delivering high-quality, attributed digital objects online at scale, widely used in cultural heritage digital collections.

Information managers: Professionals who organize and maintain digital information systems in cultural heritage institutions, ensuring data is accessible, secure, and properly structured.

Insight providers: Domain experts in cultural heritage who analyze collection data and translate their expertise into actionable preservation strategies.

Interbellum: The period between the First and Second World Wars (1918-1939). The Mundaneum existed and flourished during this period, representing innovative approaches to information organization before digital technologies.

J

Japan Search: A national digital platform launched in 2020 to integrate and provide access to Japan's cultural heritage collections, similar to Europeana in Europe.

L

Large language models (LLMs): AI systems trained on vast amounts of text that can generate human-like text and code. They represent the current frontier in AI-augmented programming, helping developers by generating and explaining code.

M

Macro X-ray fluorescence (MA-XRF) scanning: A hybrid technique that merges analytical chemistry with imaging science to create element distribution maps, representing the convergence of multiple data categories in cultural heritage analysis.

Material analysis data: Laboratory-generated data revealing the fundamental composition, structure, and condition of cultural materials, produced through various spectroscopic, chromatographic, and microscopic techniques.

MATLAB: A proprietary programming language and environment released commercially in 1984, specialized for mathematical and technical computing. It's widely used in engineering and scientific research for data analysis and visualization.

Modeling specialists: Conservation scientists, building physicists and preventive conservators who develop predictive models for material aging, environmental impact assessment, and risk analysis in cultural heritage.

N

NATO: The North Atlantic Treaty Organization, which hosted the 1968 Software Engineering Conference in Garmisch, Germany. This conference was where the term "software engineering" was first coined to address growing complexity in software development.

P

Plankalkül: The first high-level programming language, designed by Konrad Zuse between 1942-1945. It was conceptually ahead of its time, including data structures and structured programming features decades before they became standard.

Platform builders: Digital asset managers and database administrators who design and implement systems for managing diverse heritage data types across institutions.

Programming: The process of creating instructions for computers to perform specific tasks. It involves designing algorithms, writing code, testing, debugging, and maintaining software.

Programming languages: Formal languages comprising sets of instructions that produce various kinds of output when executed. They serve as the interface between human logic and machine execution.

Pseudo-developer: A domain expert (often a researcher) with limited training in code development who writes code to automate processes. They understand programming logic but typically work alone, focusing on solving specific problems rather than building complex systems.

Python: A high-level, interpreted programming language that gained prominence in the late 1990s. Its emphasis on readability and simple syntax has made it particularly popular among scientists and other non-professional programmers.

R

R: A programming language and environment for statistical computing and graphics created in 1993. It has become a standard tool for statisticians, data analysts, and researchers in various fields.

Reproducible Analytic Pipeline (RAP): A methodology that automates the entire workflow from data acquisition to final output, ensuring that analyses can be consistently replicated. Thimbleby proposes extending this to include code development in scientific workflows.

Research data managers (RDM): Cultural heritage professionals who master multiple programming languages, tools, and standards, and can apply software engineering practices to build or contribute to larger-scale applications, databases, or infrastructure.

S

SNOBOL (String Oriented Symbolic Language): A programming language developed in 1962 focused on text processing and pattern matching. It represented an early domain-specific approach to programming focused on textual data.

Software: Programs and other operating information used by computers. In scientific contexts, this refers to scripts and programs used in analysis, which should be organized and shared with appropriate documentation.

Software and Data Carpentry: Training programs established by Greg Wilson to teach researchers essential computing skills. They focus on practical "good enough" practices rather than theoretical ideals of software engineering.

Software engineering boards (SEBs): Proposed by Thimbleby as forums to authorize and provide advice on implementing quality software engineering in research. They would function similarly to ethics boards in reviewing and improving computational aspects of research.

Spatial Data: Vectorial representations of physical objects and environments through 2D drawings, 3D models, and spatial datasets that preserve dimensional and relational properties of cultural heritage.

SPECTRUM: A UK collections management standard first published in 1994 that provides procedures for documenting objects and the processes they undergo within a museum.

U

UNESCO: The United Nations Educational, Scientific and Cultural Organization, which registered the Mundaneum as World Heritage in 2013. UNESCO works to preserve cultural heritage, including innovative information systems like the Mundaneum.

Universal Bibliographic Repertory (RBU): A vast index card system created by Paul Otlet and Henri La Fontaine as part of the Mundaneum. It aimed to catalog all published knowledge using the Universal Decimal Classification system.

Universal Decimal Classification (UDC): A classification system published in 1905 based on the Dewey Decimal system but more flexible. It uses a numerical code system to organize human knowledge across language barriers.

V

Visual Basic: A programming language developed by Microsoft in 1991 that made building Windows applications accessible to casual programmers. It pioneered the rapid application development approach with visual design tools.

Z

Z1: The first programmable computer, built by Konrad Zuse in 1936 though never properly functioning. It consisted of an arithmetic unit, memory, and input/output units with a punched paper tape program.

Z3: The first fully functional programmable computer, built by Konrad Zuse in 1941 in Germany. It represented Zuse's vision of computing as an integrated whole of hardware and software.