Trends and Challenges for Software Engineering in the Mobile Domain

AUTHORS

- **Luciano Baresi** (Politecnico di Milano)
- **William G. Griswold** (University of California San Diego)
- **Grace A. Lewis** (Carnegie Mellon Software Engineering Institute)
- **Marco Autili** (University of L'Aquila)
- **Ivano Malavolta** (Vrije Universiteit Amsterdam)
- **Christine Julien** (University of Texas at Austin)

ABSTRACT

Mobile computing is relevant, everywhere, and evolves so fast that it deserves special attention. This article builds on conversations that started during a panel session on "*the role of engineering and development in mobile software"* held at the IEEE/ACM International Conference on Mobile Software Engineering and Systems (MobileSoft'18). The panel highlighted that mobile computing is not just Android and mobile apps. It touched on the impact of mobile computing on software engineering practices, the problem of forming the mobile software engineering professional, and the transition of research to industry. It also addressed the problems of logical vs. physical mobility and of supporting the "always on" mentality. At the end of the panel we all felt that there was so much more to talk about, that we continued the conversation and summarized our extended discussions in this article.

KEYWORDS

Mobile computing, Software engineering, Android and beyond, Research and Industry.

1. Introduction

During the past decade, the rapid rise of mobile computing has upended software engineering research and practice [1]. Applications now readily integrate on-device capabilities, from GPS data and wireless communications, to cameras and myriad other sensors [2]. Applications are also immersive, integrating their end users with a rapidly changing digital landscape and leveraging dynamically available resources to enable more expressive sensing or to reduce energy consumption.

The impact of these changes is evident in both software engineering research and in industry settings. New sensing contexts, combined with the power of mobility and ubiquity, have radically changed how mobile software is developed in industry settings. The "classical" phases of software engineering no longer apply directly, and novel approaches, frameworks, and the changing landscape demand a substantial shift of perception with respect to these classical approaches. New methodologies, solutions, and frameworks, and also a different way of working are mandatory and require a direct link between research, industry, and end users.

The recent uptick in research directly related to mobile computing has been dramatic and highlights the importance of the topic. In ICSE and FSE combined, the two flagship software engineering conferences, the number of articles in the main research tracks relating to “mobile” or “Android” has increased from 14% of all accepted research papers in 2016 to 31% of the accepted papers in 2018, a 120% increase in the span of two years. On the end users side, the digital media usage time is strongly driven by mobile devices, with smartphones and tablets accounting for 66% of all time spent, against desktop usage which accounts for 34% only [3]. Starting from these figures, this article provides an organized summary of the outcomes of the discussions that 65 researchers and practitioners had during a panel session on “*the role of engineering and development in mobile software”* held at the MobileSoft’18 conference in Gothenburg, Sweden. The panel was designed as a guided conversation over the dimensions shown in Figure 1.

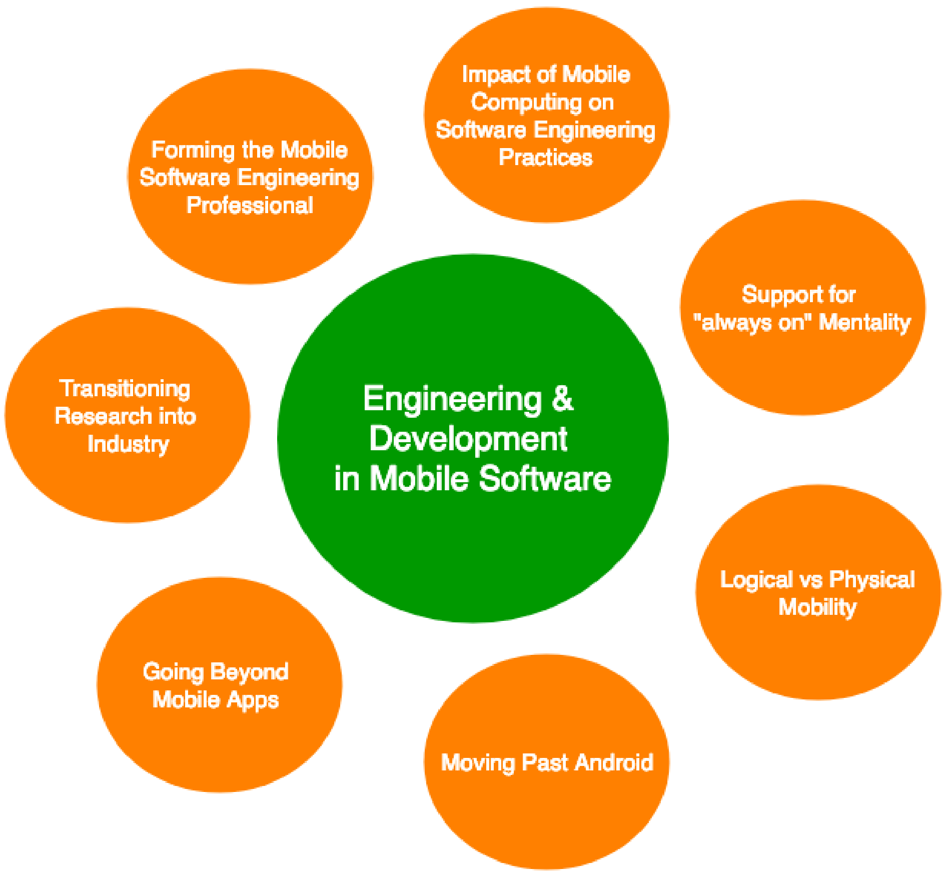
Impact of Mobile Computing on Software Engineering Practices
Forming the Mobile Software Engineering Professional
Support for "always on" Mentality
Transitioning Research into Industry
Engineering & Development in Mobile Software
Logical vs Physical Mobility
Going Beyond Mobile Apps
Moving Past Android

Figure 1 - Dimensions of the role of engineering and development in Mobile Software

The dimensions are orthogonal concerns that crosscut the main stages of the software development life cycle (i.e., requirements, design and development, testing, and maintenance). These dimensions are complementary and give fresh life to the work in [12] that, at the time of writing (2016), discussed current and future research trends and was organized around these stages.

The rest of the article is organized as follows. Sections 2 through 8 discuss the seven dimensions in detail. Section 9 identifies some lessons learned and concludes the article.

2. The Impact of Mobile Computing on Software Engineering Practices

Apps are GUI-centered — efficient, responsive, and appealing user interfaces are key for the success of apps. Apps require ultra-fast development cycles because end users rate and review apps massively and at tremendous rates, new versions of Android and iOS are released at least every year, and new competitors continuously pop up in app stores. Apps are very complex software systems that interact with many physical entities (both inside and outside the phone) and integrate diverse (potentially third-party) services in the cloud. Apps must also work on very diverse devices in terms of version of the operating system, sensors, etc. [4].

Context-awareness [5,6] enables the development of innovative features for mobile applications but creates challenges for mobile software engineering. Being able to take advantage of different sensors on mobile devices, as well as use data from end users, can lead to an improved user experience. However, there are tradeoffs with energy efficiency, security, and privacy that need to be considered as well. We advocate that performing tradeoff analysis should be an explicit tenet of the design and development of mobile apps.

Another unique aspect of mobile software development is the need to be adaptive. For example, in power management, non-urgent communications can be delayed to save power (e.g., reporting pulse rate over time), while urgent ones should not (e.g., reporting tachycardia). This points to another unique aspect, which is the constant presence of failure: poor connectivity, low battery levels, sensor inaccuracies, etc.

These challenges make fully automated approaches to adaptivity difficult. When a (network) failure occurs, sometimes it should not be hidden from the end user in order to allow them to take action outside the application (e.g., troubleshoot the failure and change the phone's wireless settings). End users desire responsiveness, energy efficiency, always-on connectivity, and of course correctness [8]. End users also want the latest features. Developers desire rapid development at reasonable costs. Software researchers have long sought for language-based and compiler-based solutions for managing trade offs. These are critical, but they cannot be a silver bullet [7].

Research in aspect-oriented software development can find application here. Two recent projects highlight the promise, APE and ANEL. APE (and its successor TEMPUS) addressed the challenges of adaptive power management, enabling a developer to declare which computations, under given situations, could be delayed to save power [13]. ANEL addressed the complexities of robust mobile networking, enabling a developer to incrementally and declaratively improve simple networking implementations [15]. There are also many available libraries such as Retrofit[1] and Volley[2] that enable developers to separate complex service consumption and networking code from application and fault tolerance logic.

The bottom line is that although architecture, system qualities, and tradeoffs have always been a key part of software engineering, these concepts become more prevalent and

[1] https://square.github.io/retrofit/
[2] https://developer.android.com/training/volley

important to address in mobile software engineering due to much more dynamic operating environments and diversity of end users and platforms [12]. Providing software engineers the tools to analyze and study tradeoffs, in addition to tools to help with separation of concerns, is key for the development of adaptive mobile applications.

3. Support for the "Always-on" Mentality

End users want a seamless experience, whether in the home, at work, or on the move. The challenges of supporting the always-on mentality [9] of end users can only be addressed with the always-on connectivity in the mind of software engineers.

Today's ubiquitous networking pursues the convergence of wireless telecommunication networks and wireless IP networks to provide seamless connectivity from everywhere at any time thanks to the multi-radio capabilities offered by today's mobile devices. As a result, the ubiquitous networking environment cannot be considered as a "passive" entity that only transports data between end points. Rather, mobile apps must consider it as an "active" party to be fully exploited. Supporting the always-on mentality in such a complex and dynamic networking environment calls for the development of ubiquitous-oriented solutions, which give mobile apps the "impression" of perfect connectivity, hence assuming the always accessibility of remote hosts.

End users expect their apps to always work and their devices to always be available, In this respect, Defensive Programming[3] is a needed software engineer mentality in today's world. Systems need to be written expecting that the worst will happen. Fault Tree Analysis, a tool commonly used by hardware engineers, is useful to brainstorm about all the bad things that can happen. Even though not specific to mobile software development, another example of a tool that reflects this mentality is the Netflix project Chaos Monkey, a tool developed in 2011 by Netflix to test their infrastructure. The tool intentionally disables components in the production network to test how remaining systems respond to the outage.

4. Logical vs. Physical Mobility

About 15 years ago, we used to distinguish between Physical Mobility of devices (where mobile devices move around) and Logical Mobility (where pieces of code and state are moved across hosts). We can definitely still distinguish between logical and physical mobility, but the real question is whether it makes sense to continue doing so. For the past decade, ever since Mahadev "Satya" Satyanarayanan introduced the concept of cyber-foraging [8], there has been a lot of research in how to partition and deploy mobile applications so that they can opportunistically take advantage of more powerful *computing resources* in order to optimize battery life and *network usage*. The runtime optimization algorithms that have been developed to determine whether it makes sense to execute locally or remotely are very creative. The techniques to design and partition applications to determine the unit of offload have ranged from offloading at the thread level, method, class, service, to full application. In the end, the conclusion for all that research has been (1) it only makes sense to offload if the

[3] http://www.drdobbs.com/defensive-programming/184401915

cost of remote execution is lower than the cost of local execution, and (2) the more information you have to make this decision the better. However, if you look at the assumptions made by most of this research, they are impossible to guarantee in practice. For example, they assume perfect connectivity, or perfect knowledge of the information needed in the optimization function, or identical local and remote applications that are always accessible, or extensive tagging on behalf of the developer, or that there are trusted edge or cloud servers always available to receive computation offload requests. The real (and more feasible) power of cyber-foraging is in determining how to place data and computation closer to mobile devices, when they need it. This means that the app is no longer solely responsible for making cyber-foraging decisions; instead, the problem becomes a system and network one. Cloud-based systems need to be able to push computation and data to their trusted surrogates, and mobile devices need to be able to (automatically) locate these surrogates and deal with the intermittent connectivity and device mobility.

5. Moving Past Android

With 88% of all sold smartphones being Android phones, Android has a huge portion of the market share today[4]. This is also reflected in the amount of scientific contributions on Android in our flagship conferences. For example, out of the 105 studies published in the technical track of the 40th edition of the International Conference on Software Engineering (ICSE 2018), 15 studies are about mobile apps, and among them 14 are about Android. We can also trace this trend towards Android to *technical aspects*. Firstly, Android is open-source, meaning that researchers can customize the Android OS integrating their solutions and perform experiments on them; the source code of the whole platform is available and versioned since the beginning, allowing researchers to study the internals of the platform and how it evolved over time. Secondly, Android is heavily based on Java, a language that is widely adopted and taught in Universities. Moreover, many researchers are comfortable with Java, thus easing their understanding about the specific phenomena happening in the source code of Android apps. Finally, today a large amount of techniques, tools, and datasets exist for statically analyzing and testing Android apps, which can be leveraged by researchers for scaling up the execution of new studies, thus leaving behind other aspects of the mobile app ecosystem, which are less covered, e.g. in terms of analysis tools.

Nevertheless, if we look back in time it is evident that the mobile ecosystem is extremely dynamic, with platforms unpredictably rising and falling in terms of sales of devices (for example, 16% of all smartphones sold in 2010 were running the Blackberry OS), company acquisitions, and end users flowing from/to other platforms.

This technological volatility makes us wonder *what will be the fate of the Android-specific large body of knowledge and tools that researchers are producing*. As a step forward, we suggest to the research community to direct their focus on more fundamental aspects of the mobile ecosystem, such as the (currently sub-optimal) distribution model of mobile apps, the

[4] https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

IDEs, the APIs provided by the mobile platforms, etc. In order to make research results more future-proof and relevant, researchers should focus on fundamental challenges in the mobile software engineering ecosystem. As a first step, researchers may commit to have dedicated sections in their papers about the generality of the conducted studies, e.g., by showing how they are not specifically bound to Android and by highlighting which parts depend on Android and which can be considered as generally applicable.

Complementary, new languages and cross-platform frameworks are emerging for mobile apps (e.g., Kotlin, React Native, Flutter) and they are extremely popular and heavily adopted in industry, but the research community tends to ignore them. As a community, researchers should *get out of the (Java+Android) comfort zone* and strive towards studying new technologies and considering their potential impact on the mobile computing landscape.

6. Going Beyond Mobile Apps

Wearables and constellations of Internet-of-Things (IoT) devices are revolutionizing the way we live and work (e.g., smart home appliances, Industry 4.0, healthcare) and will be driving the research in the near future. New applications of mobile computing will likely have an even bigger impact, such as those enabled by machine learning, computer vision, augmented/virtual reality, natural language processing, and speech recognition. Privacy, security, performance, and energy consumption for mobile apps are being actively investigated by researchers and practitioners, but they have not been fully explored yet in the context of these new applications of mobile computing.

From the infrastructure perspective, we are moving towards a continuum where computing starts on mobile devices and continues onto edge and cloud infrastructures, reaching also many other devices (e.g., ISP gateways, cellular base stations) [10]. We need solutions to exploit such a continuum in a flexible way, without pre-deciding the allocation of components and thus the languages and tools to create them. Edge infrastructures may add a new interesting dimension to the problem of managing computationally heavy tasks on mobile devices, especially in the context of wearable devices with very limited battery life and computational power, connected cars, and smart objects (e.g., home automation and advanced logistics).

In this context, a possible step forward may be the combination of the microservices architectural style and devOps concepts for dynamic surrogate provisioning at the edge, networking protocols that deal better with intermittent communications such as delay-tolerant networking, and new business models to support cyber-foraging at industrial scale. Cyber-foraging will be one of the enablers of augmented reality, given that performing image recognition and analysis are crucial elements of augmented reality frameworks like Google's ARCore[5], but they are extremely expensive for mobile devices both in terms of performance and battery consumption.

7. Transitioning Research into Industry

[5] https://developers.google.com/ar/

If we look at research papers on mobile software, we see a spectrum of solutions, ranging from narrow studies on low-level aspects of mobile app development (e.g., how Java collections may impact the performance of an app) to system-level approaches on the whole mobile ecosystem (e.g., new permission systems for Android). Our perception is that narrow and in-depth studies may be more easily adopted by industry (e.g., a new static checker for Android Studio) compared to wide studies which may strongly improve (but also disrupt) the current status quo.

Adoptability is also facilitated by the development of well-tested and maintained tools embodying the research results proposed by researchers, so that industry can independently verify, check, and start using the proposed solutions. However, if on one side we know that industry expects ready-to-use solutions, on the other side going too much towards this model may lead to the risk of not focusing on more fundamental problems, which may be more rewarding in the long run. As often happens in software engineering, it is up to the involved players to find the right tradeoff. A first step is to build a certain sensibility with respect to the technology transfer process (use [6] as starting point) and to learn from known success stories (see Sidebar). From our experience we learned that when setting up a collaboration, the first and necessary condition towards success is to share and discuss up-front the goals of both the academic and industrial partners and let them converge. This shields researchers from running the risks of not considering the realities of operational environments and also from falling into the trap of trying to sell to industry the answers we have without listening to their questions.

The APE approach discussed above is an example of how the complexities of a research product may be a “good fit” for near-term impact. After the APE work was published, Google came out with extensions to the `AsyncTask` Android class to conditionally run a task based on the network or energy status of the device. Although less powerful than the APE approach, Google’s extension was perfect for Android, given the centrality of AsyncTasks in Android. In this regard, the APE-related research was less transferable in the near term, but more impactful in the long term. Another problem with some research results is the excessive complexity of the proposed solutions. To address limitations in APE’s path-based power management, TEMPUS takes an object-oriented approach. This requires elaborate static and runtime analysis, which could make it difficult for industry to adopt. This kind of complexity was criticized by Willy Zwaenepoel in his MobiSys’07 keynote “*P2P, DSM, and Other Products of the Complexity Factory*”, in which he said “*complexity is untenable at scale -- and scale is what industry does*”. We, both researchers and professionals, should remember this and focus more on solving problems rather than creating new (and more complex) ones.

SIDEBAR - Success stories about research that has impacted the mobile industry

Notably, two industry sectors, namely (1) medical devices and equipment, and (2) network systems and communications, have a recent history of extensive collaboration with academic researchers. For instance, concerning the network systems and communications’ sector, we can mention the contributions of applied academic research to

packet switching and the Internet TCP/IP protocol, both key elements in the development of the Internet, contribution to the development of routers, ATM switches, DSL (digital subscriber line) technology, computer graphics, search engines, traffic management, stable broadcast networking and, last but not least, the development of standards.
At Carnegie Mellon University there are several cases of researchers who have started their own companies based on their research results. For example, Luis von Ahn, the CEO and co-founder of DuoLingo, changed the way that people learn languages. He is also the founder of the company reCAPTCHA, which was sold to Google in 2009. Fernando de la Torre founded Faciometrics, a company developing technology for facial image analysis which was recently acquired by Facebook. What all these ideas had in common is that they were (1) simple, (2) addressed a very specific, concrete need, problem or idea, (3) had a solid implementation of a tool/app to accompany the idea.
The examples mentioned above are surely intriguing success stories, but in other cases the impacts of research can be much more diffused and difficult to track. For example, in 2008 Timothy Sohn (an alumni of the University of California, San Diego) conducted a study on mobile information needs [11]. Sohn's study ultimately formed the core of his dissertation and had a relatively high academic impact. But perhaps more importantly, three years later Sohn was hired at Google and worked on Google Now, which was an early version of Google's context-aware features in many of its mobile apps. It is doubtful that Sohn's research gave Google the idea for Google Now, but his research prepared him to work on this project and helped bring these capabilities to the masses.

8. Forming the Mobile Software Engineering Professional

When talking about the future of software, it is inevitable to acknowledge that the supreme value is in students (both Computer Science and other disciplines) and on their professional education. This leads to the following key question: what is the role of education when forming the next **mobile software engineering professional**?

"Mobile software engineering" is a very applied topic, which is why a *very applied program* would be preferable. Our hypothetical program would be composed of four main orthogonal dimensions: *Foundations*, *Experience*, *Business*, and *Research*. *Foundations* is the largest dimension and it includes software engineering and systems engineering courses at different levels, followed by mobile-specific software engineering courses. Given that a mobile software engineer these days is in charge of realizing software which will be likely part of users' everyday life, the *Experience* dimension entails courses related to Human-Computer Interaction, user experience design, GUI design, etc. The *Business* dimension aims at providing the instruments for setting up and working within a software company (e.g., a startup) via courses about entrepreneurship and business-related topics. As far as the *Research* dimension, students would find a real problem to work on, study the current literature on solutions for that problem, and then propose a solution that is novel and advances the state of the art. Orthogonally to those four dimensions, instructors will be in charge of building a rich interface between students and companies, so to facilitate (i) their transition towards positions in industry and (ii) their exposure to industry-relevant problems and practices.

9. Lessons Learned and Next Steps

This article summarizes the main takeaways that emerged during the panel discussion, plus our continued discussion on the topic. Researchers and practitioners can build on the highlighted themes and on some lessons learned:

- Diverse sensors and data can lead to an improved user experience, but there are tradeoffs with energy efficiency, security, and privacy.
- Mobile software must be adaptive (e.g., power management) and ready to manage the constant presence of failures (e.g., poor connectivity or low battery levels). Systems need to be written expecting that many bad things will happen.
- The research community is urged to focus more on fundamental aspects of the mobile ecosystem, instead of being too Android-specific.
- We must be ready to manage a computing continuum where computing starts on mobile devices and continues onto edge and cloud infrastructures.
- Industry expects ready-to-use solutions, but going too much towards this model may lead to the risk of not focusing on more fundamental problems.
- *Mobile software engineering* is a very applied topic that calls for a very applied educational programs that include Foundations, Experience, Business, and Research.

As for next steps, there is room for taking mobile computing in many different ways:

- *Advancing the mobile experience* - The always-on mentality of today's mobile users calls for the engineering of ubiquitous-oriented network- and middleware-layer solutions which give mobile apps seamless connectivity, while guaranteeing security and privacy.
- *Innovation and growth* - The ultra-fast and low-latency network capacities, and low-power consumption promised by 5G networks, call for a new generation of engineers that can revamp the mobile market with cutting-edge apps that fully exploit the power of AI-enabled devices coupled with the power of a 5G network.
- *Protection* - Mobile systems are increasingly autonomous in making decisions over and above users or on behalf of them. Often, their autonomy exceeds the system boundaries and invades user prerogatives. As a consequence, ethical issues such as unauthorized disclosure and mining of personal data, or access to restricted resources, are matters of utmost concern because they impact the moral rights of each human being and affect social, economic, and political spheres [14]. As next step, engineers must approach these problems from the regulatory side by contributing to the introduction of new laws, and from the technical side by adopting transparency and accountability criteria in software development.

References

1. C. Ebert, K. Shankar, "*Industry Trends 2017*", IEEE Software, vol. 34(2), pp. 112-116, 2017 - doi. 10.1109/MS.2017.55.
2. V. Pejovic, M. Musolesi, "*Anticipatory Mobile Computing: A Survey of the State of the Art and Research Challenges*", ACM Computing Surveys, vol. 47(3), 29 pages, 2015 - doi: 10.1145/2693843.

3. A. Lella, A. Lipsman, The U.S. Mobile App Report, comsCore white paper, 2017.
4. M. E. Joorabchi, A. Mesbah, P. Kruchten, "*Real challenges in mobile app development*", in ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 15-24, 2013 - doi: 10.1109/ESEM.2013.9.
5. K. Mens, R. Capilla, H. Hartmann, T. Kropf, "*Modeling and Managing Context-Aware Systems' Variability*", in IEEE Software, vol. 34, no. 6, pp. 58-63, 2017 - doi: 10.1109/MS.2017.4121225.
6. E. Eshet, H. Bouwman, "*Addressing the Context of Use in Mobile Computing: a Survey on the State of the Practice*", in Interacting with Computers, vol. 27(4), pp. 392-412, 2015 - doi: 10.1093/iwc/iwu002.
7. C. Wohlin et al., "*The Success Factors Powering Industry-Academia Collaboration*", in IEEE Software, vol. 29(2), pp. 67-73, 2012 - doi: 10.1109/MS.2011.92.
8. G. P. Picco, C. Julien, A. L. Murphy, M. Musolesi, G. Roman, "*Software engineering for mobility: reflecting on the past, peering into the future*", in Proceedings of Future of Software Engineering, pp. 13-28, 2014 - doi: 10.1145/2593882.2593884.
9. M. Satyanarayanan et al., "*Pervasive Computing: Vision and Challenges*", in IEEE Personal Communications, vol. 8(4), pp. 10-17, 2001 - doi: 10.1109/98.943998.
10. L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, G. Quattrocchi. 2019. A Unified Model for the Mobile-Edge-Cloud Continuum. ACM Transactions on Internet Technology. 19, 2, Article 29, 21 pages, 2019 - doi: 10.1145/3226644.
11. T. Sohn, K. A. Li, W. G. Griswold, J. D. Hollan, "*A diary study of mobile information needs*", in Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, pp. 433-442, 2008 - doi: 10.1145/1357054.1357125.
12. M. Nagappan and E. Shihab, "*Future Trends in Software Engineering Research for Mobile Apps*", in Proceedings of the IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, pp. 21-32, 2016 - doi: 10.1109/SANER.2016.88.
13. N. Nikzad, O. Chipara, W. G. Griswold, "*APE: an annotation language and middleware for energy-efficient mobile application development*", in Proceedings of the 36th International Conference on Software Engineering, pp. 515-526, 2014 - doi: 10.1145/2568225.2568288.
14. M. Autili, D. D. Ruscio, P. Inverardi, P. Pelliccione and M. Tivoli, "*A Software Exoskeleton to Protect and Support Citizen's Ethics and Privacy in the Digital World*", in IEEE Access, vol. 7, pp. 62011-62021, 2019 - doi: 10.1109/ACCESS.2019.2916203.
15. X. Jin, W. G. Griswold, Y. Zhou, "*ANEL: robust mobile network programming using a declarative language*", in Proceedings of the 5th International Conference on Mobile Software Engineering and Systems, pp. 202-213, 2018 - doi: 10.1145/3197231.3197237.

Authors bios

Luciano Baresi is a full professor at the Politecnico di Milano. Luciano was visiting professor at University of Oregon (USA) and visiting researcher at University of Paderborn (Germany). His research interests are in the broad area of software engineering and include formal approaches for modeling and specification languases, distributed systems, service-based applications and mobile, self-adaptive, and pervasive software systems. Email: luciano.baresi@polimi.it

William G. Griswold is a Professor of Computer Science and Engineering at UC San Diego. He received his Ph.D. in Computer Science from the University of Washington in 1991. His research interests include ubiquitous computing, software engineering, and computer science education. Griswold is a pioneer in the area of software refactoring, and was a major contributor to the development of Aspect-Oriented Software Development. Later he built ActiveCampus, an early mobile location-aware system. His recent CitiSense and MetaSense projects investigated mobile technologies for low-cost ubiquitous real-time air-quality sensing. He is a professional member of the ACM and IEEE Computer Society, and a past Chair of the ACM Special Interest Group on Software Engineering (SIGSOFT). Email: wgg@cs.ucsd.edu

Grace A. Lewis is Principal Researcher and lead of the Tactical and AI-enabled Systems (TAS) initiative at the Software Engineering Institute at Carnegie Mellon University. Her current areas of research and interest include software engineering for AI/ML systems, IoT security, edge computing, software architecture (in particular for systems that integrate emerging technologies), and software engineering in society. She is a member of organizing and program committees for multiple conferences in the areas of software engineering and software architecture, as well as an author and reviewer in these fields. She received her PhD in Computer Science from Vrije Universiteit Amsterdam. She is a Senior Member of IEEE and a member of the IEEE Computer Society Board of Governors (2020-2022). Email: glewis@sei.cmu.edu

Marco Autili is associate professor at the University of L'Aquila. His research focuses on automated synthesis for composing distributed systems, context-oriented privacy-aware mobile software programming, resource-oriented analysis of mobile apps, formal specification and checking of temporal properties. He is (has been) involved in several EU and Italian research and development projects, as scientific coordinator, scientific and technical leader, research unit coordinator, and work package leader. He is in the editorial board and in the programme committee of several top-level international journals, international conferences and workshops. He received his PhD in Computer Science from the University of L'Aquila in 2008. He is a professional member of ACM and IEEE. Email: marco.autili@univaq.it

Ivano Malavolta is assistant professor at the Computer Science Department of the Vrije Universiteit Amsterdam (The Netherlands). His research focuses on data-driven software engineering, with a special emphasis on software architecture, mobile software development, and robotics software. He applies empirical methods to assess practices and trends in the field of software engineering. He has authored several scientific articles in international journals and peer-reviewed international conferences proceedings in the software engineering field. He received a PhD in computer science from the University of L'Aquila in 2012. He is a member of ACM, IEEE, VERSEN, Amsterdam Young Academy, and Amsterdam Data Science. Email: i.malavolta@vu.nl

Christine Julien is a full professor in the Center for Advanced Research in Software Engineering at the University of Texas at Austin. Her research is at the intersection of software engineering and dynamic, unpredictable networked environments, with a focus on models, abstractions, tools, and middleware to ease the software engineering burden associated with building applications for pervasive and mobile computing environments. Her research has been funded by numerous foundations and organizations, her work has appeared in many peer-reviewed journal and conference publications, and she serves regularly as a peer reviewer for conferences and journals in these fields. She is a Senior Member of the IEEE. She received her D.Sc. from Washington University in Saint Louis. Email: c.julien@utexas.edu