Successful Systems in Production Graduate Teaching

Ali Shoker
HASLab, INESC TEC & Minho University
Braga, Portugal
ali.shoker@inesctec.pt

Abstract—**This paper presents our experience in coordinating and teaching a novel graduate systems and computing course named "Successful Systems in Production" (SSP). The course targets graduate students of different research interests in Computer Science. The course aims at giving a breadth knowledge on cutting-edge well-known systems in production, and exploring the potential synergies across different areas of research. Having its roots in Distributed Computing, SSP addresses those systems that overlap with other research areas like Computational Systems, Parallel Computing, Databases, Cloud Computing, Artificial Intelligence, Security, etc. SSP exhibits an agile topic selection model that fits several students' backgrounds in each academic year. The topics focus on the practical aspects of each selected system that is considered "successful", i.e., based on its worldwide impact and technical significance. This is important for graduate students to acquire best practices in industry and academia, necessary to build practical computing systems. In the same vein, the assessment method includes a project that is based on one of the presented systems and also intersects with the student's own research plan. Based on our teaching experience and the excellent feedback of the students, we strongly recommend this graduate course to be taught at other universities.**

Index Terms—**Teaching; distributed systems; graduate; PhD; computing;**

I. Introduction

Graduate courses in the areas of computing and systems aim at giving an in-*depth* knowledge in a specific area of expertise inline with the student's ongoing research, or *breadth* knowledge to understand and establish the synergies with other areas. While in-depth courses are often suitable for graduate students whose research topics are inline with the selected course, they are usually selected from undergraduate or graduate programmes that focus on fundamentals. Consequently, they give little focus on the practical aspects, highly desired in modern computing systems. On the other hand, courses that are not inline with the student's research plan or thesis are of less interest, especially being not explicitly prepared to address different backgrounds. This suggests the introduction of novel courses that target the graduate student, in particular, covering the two breadth and depth aspects with enough flexibility to address the largest branch possible of PhD subjects.

Targeting the case of graduate and PhD programmes that include students of diverse research backgrounds and interests, an interesting model is to propose PhD-tailored courses in different CS areas, as those at the Carnegie Mellon University (CMU) [3]. Although one can argue that such PhD-tailored courses are more convenient than specialized undergraduate or Masters level courses that are offered per student as "Breadth" options in other PhD programmes, e.g., at Stanford university [13] and MIT [11], the dilemma is how to choose the set of courses that largely fits the greatest majority of students' backgrounds. On the other hand, courses that belong to the category of "Special Topics" or "Advanced Topics" as those taught at ETH [14], EPFL [6], and Northeastern university [12], target several topics that may better intersect with many students' interests; however, this diversity remains limited to a specific area of research. In addition, these courses do not give significant emphasis to the best practices of building computing systems for production—arguably crucial to modern scientists.

In this paper, we present our experience in coordinating and teaching a novel graduate course entitled "Successful Systems in Production" (SSP) at the MAPI PhD programme[1] [9]. Given that diverse techniques in different CS areas take part in building a modern system, SSP aims at giving a breadth knowledge of a set of well-known systems in production as those presented in Table II. The purpose is to educate future researchers and scientists in different areas how comprehensive systems in production work, and importantly help them understand the interplay and impact of the different parts of a monolithic system in production. To narrow down the gap between theoretical research and practical systems for production use, SSP gives emphasis to the practical part of a chosen system to convey best practices that are necessary and complementary to the theoretical knowledge base in the computing systems area.

SSP follows an agile selection method for topics (i.e., "successful systems") that adapts to the research areas of the enrolled students in the course. Therefore, from a curated list of in-production systems (as in Table II) that are arguably considered "successful", based on their global impact and technical significance, six systems are chosen to be taught each year, thus maximizing the intersection with most students'

The research leading to these results has received funding from the European Union's Horizon 2020 - The EU Framework Programme for Research and Innovation 2014-2020, under LightKone grant agreement No. 732505.

[1]MAP-I is a PhD programme in Computer Science (CS) that follows the CMU teaching model [3]. It is organized among three northern Portuguese universities (Minho, Aveiro, and Porto).

978-1-7281-5975-1/19/$31.00 ©2019 IEEE
DOI 10.1109/EduHPC49559.2019.00011

backgrounds. The assessment method is also tailored to diverse graduate research areas as each student shall propose and deliver a project that is related to one of the presented topics and intersects with his own research plan. In our experience, this keeps the student more engaged in the course and reduces the "distraction" off her research plan—that many students consider a waste of time, especially if her research interests do not exactly lie in the computing or systems areas.

Our experience over the past two years of teaching SSP shows that the course achieves its goals being engaging to students of different backgrounds and shows that SSP contributed to the progress of the students' research plan or thesis. The students' anonymous feedback presented in Section IV also supports this observation where more than 90% of the students (where less than one third are familiar with computing systems) recommended this course over other courses in the PhD programme. Based on this experience, we highly recommend SSP to be taught in similar graduate programmes elsewhere.

In the rest of this paper, we overview in Section II MAPI's programme rules and conditions, together with the reasons behind the idea of the course. In Section III, we present the course structure, methodology, sample topics, and the assessment method. Then, we present some success indicators in Section IV before concluding in Section V.

II. PhD Programme and Motivations

A. Programme Description and Rules

MAPI's PhD programme [9] follows the same model at CMU Electrical and Computer Engineering [3] by requiring a set of technical depth and breadth courses and seminars—in addition to the thesis. Technical courses are mostly Bachelor or Masters courses in a specialized area or selected from a list of PhD-courses in *breadth* areas. While CMU (and most American universities) allow Bachelor degree holders to immediately enroll in a PhD programme provided that the candidate takes the necessary in-depth as well as breadth courses, MAPI students are Masters degree holders, and thus they are expected to have the required knowledge to pursue the PhD. Given this, the proposed PhD courses in the programme focus on breadth areas in three *options*: Theory and Foundations, Computing Paradigms, and Technology.

Courses in all CS areas are proposed in each category and should pass the scientific committee's approval and then get selected by at least five students to be taught in an academic year. To make this selection, a symposium is organized at the beginning of the academic year to present the programme rules, potential courses, and follow up the progress of ongoing PhDs. In principle, a PhD student must take five courses (at least one from each category) during his PhD, and she has to pass half of them (usually during the first year) to be able to enroll in the *thesis option* in the following year if a *pre-thesis* (i.e., thesis proposal) is submitted.

TABLE I
SOME OF THE RESEARCH TOPICS AT MAPI PHD PROGRAMME.

Area	Acronym
Systems	Sys
Software Engineering	SE
Databases	DB
Communications & Networking	Com
Cloud Computing	CC
Artificial Intelligence	AI
High Performance Computing	HPC
Security	Sec
Blockchains	BC
Energy Consumption	EC
Internet of Things	IoT

B. Motivations

The idea of Successful Systems in Production is motivated by several challenges at MAPI, among which many are common in other PhD programmes in other universities. These challenges contributed to shaping the course structure, methodology, and evaluation. The main challenges are summarized as follows:

a) Diversity: The PhD programme covers a wide range of research areas to attract PhD candidates with various research focuses in Computer Science. In an academic year, up to ten different research areas could be of interest. Among these areas are those presented in Table I together with their acronym—that will be used in the rest of the paper. This diversity of topics suggests proposing teaching topics that fit a majority of them considering the course topics and deepness.

b) Cost: The number of PhD candidates that enroll each year is limited. At MAPI, it ranges between 15 and 30. Given the diversity of research topics aforementioned, it is unaffordable to open a course unless there is a sufficient number of students selecting it. (Indeed, sometimes the number of proposed courses is equivalent to the number of newly enrolled students in the programme.) For instance, provided that the average rate of candidates per research area at MAPI is two, at least five candidates should show interest in a course proposal to run.

c) Time: The highest majority of candidates have their PhD research subject defined prior to enrolling in the programme. In several cases, the subject is an extension to a Masters thesis. Given this, our experience shows that most PhD candidates consider some of these courses "distracting" to their thesis and time consuming, especially when the course is deep and highly demanding in terms of projects and assignments.

d) Relevance: Although taking only courses inline with a research topic may look optimal, the knowledge of a PhD may remain narrow, which is undesired in a research career due to the growing overlap between different research areas. On the other hand, relevance also suggest designing computing systems that are practical for real use. Consequently, it is wise to study the success stories at leading industry and transmit them to future scientists as learned lessons complementary to their fundamental research.

III. Successful Systems in Production (SSP)

Successful Systems in Production is a "systems" graduate *breath* course that overlaps with different areas of Computer Science research areas, among them those presented in Table I. Targeting the challenges discussed in the previous section, the course covers diverse well-known systems, flexible methodology, and relevant assessment method. In this section, we present the proposed syllabus in general and the methodology followed in each class. We then exemplify through delving into two selected topics showing the fine-grained content details and exemplify the teaching method of each. We then show the assessment method used in the course.

A. Syllabus

As their name indicate, the taught topics in SSP are chosen based on the following criteria: (1) they tackle "systems" not only fundamental research; (2) they are already used in production, which is important to teach the students best practices in production; and (3) they are considered "successful" being well-known impactful systems with technical significance. In general, these are systems that most Computer Science or Technology people may have heard of, or have a big impact on the modern CS technology and services. The second column in Table II conveys a sample of such systems.

Knowing that the course duration at MAPI is 28 hours, it is obvious that a handful number of topics shall be taught each year. Importantly, the topic selection process is defined in two phases. The first phase proposes a curated list of topics based on the criteria aforementioned and spanning diverse research areas as those presented in Table I. The purpose of this phase is to pass the Scientific Committee approval and attract the needed threshold of students, i.e., often 5 students, to run the course. If this phase was successful, the second phase is needed to short-list the curated list of courses, this time to explicitly fit the majority of the research interests of the enrolled students in the course. This happens right after the first phase through "screening" the profiles of the students after communicating with them (via emails).

a) Curated list of successful systems: In the SSP course proposal, a curated list of twelve topics, presented in Table II, is proposed. This number is a tradeoff to keep the material diverse without having an over-killing lengthy proposal. Topics are chosen to fit the various PhD research areas, focusing on the active areas at the three universities of MAPI [9], roughly summarized in Table I. As shown in the third column of Table II, the selected topics can intersect with multiple research areas at once which increases the chance to have different students converge on a smaller set of topics. The SSP column in the table shows the name of the successful system to focus on, whereas the first column shows the broader topic of the system.

One may also observe that the chosen topics target different parts of the system software stack covering databases, communication, and computations. This is important to give the PhD candidate an overview of the entire stack and challenges in the CS area. On the other hand, the SSP column shows that the chosen systems are arguably well-known successful systems. Although some of them may not be considered "successful" as Bitcoin/Blockchains, the topic is recently the most disrupting technology to different areas of CS business and services, which, we argue, is well worth considering. Note that the selected systems do not mean they are the best in their class, but they are convenient to the course due to other factors, e.g., generality, code availability, ease of use and demonstrate, etc.

TABLE II
A sample curated list of *Successful Systems in Production*. See Table I for the Acronyms used in the "Area" column.

Topic	SSP	Area
Databases	Amazon DynamoDB	All
In-Memory Cache	Redis	Sys,DB,BC,CC,SE
Coordination systems	Apache Zookeeper	Sys,BC,SE,CC,Com,Sec
Publish/Subscribe	Twitter	Sys,SE,BC,Com,Sec
Message brokers	Kafka	Sys,AI,CC,Com,SE,Sec
Blockchains	Bitcoin	All
Cloud Computing	OpenStack	Sys,CC,DB,Com,SE,Sec
Containers	Docker	All
Big Data	Hadoop	All
Stream Processing	Spark	Sys,AI,IoT,Sec,SE,Com
Deep Learning	TensorFlow	AI,Sys,Sec,IoT,SE

b) Short list of taught topics: The second phase of short listing the topics starts right after enough students have chosen the course. The coordinator gets in touch with the students via email to ask for their background and prospective PhD thesis if it is already defined. In principle, six topics (of four hours each) should be chosen within two weeks to start arranging for corresponding invited lecturers and prepare the material. At MAPI, the course runs in two months after selection, which is enough to secure few interested lecturers within the same research areas of the topics selected. However, it is strictly desired to reduce the risks through securing half of the topics. This is done through preselecting three topics that are likely to overlap with several areas of research, and whose lecturer is available. In the past two years, these three topics have been within the area of expertise of the course proponent and coordinator—the author. Indeed, this was easy at MAPI since most students over the two years have recommended most of the topics for the following year through an anonymous questionnaire as depicted in Fig. 1.

It is also desired to allocate the first lectures to these preselected topics to give more time for invited lecturers to prepare their material. Over the three years of SSP teaching at MAPI, the three preselected topics have been: (Topic 1) Coordination in Distributed Systems (Apache Zookeeper), (Topic 2) NoSQL Databases (Amazon DynamoDB), and (Topic 3) Containers (Docker). Topic 1 is chosen as the first topic to overview the fundamentals of Distributed Systems with emphasis on coordination and consensus—being primary challenges. This choice comes from the observation that most of the "successful systems" are distributed in nature, and this topic is necessary to understand several following topics. Topic 2 focuses on Databases being a necessary part of most of systems and likely has an impact on performance, security, availability,

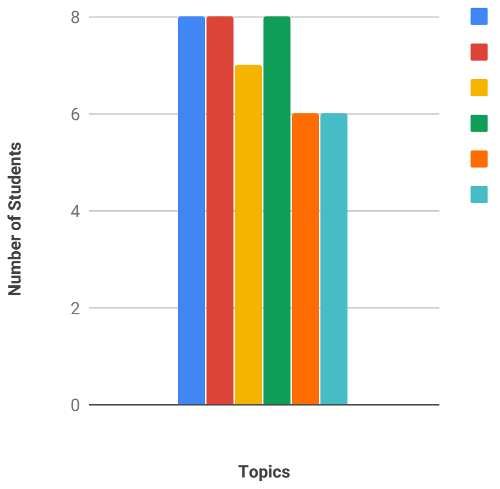

Fig. 1. The figure demonstrates the relevance of the topics' selection method. It shows that the majority of students recommended all topics to be taught in the following year. The figure does not plot the labels of the courses being different over years.

computation, etc. This topic focuses on NoSQL databases being recently widely used either at a large scale as in datacenters or as tiny databases as in web browsers or Internet of Things devices. Topic 3, i.e., Containers, is recently a very hot topic that intersects with most CS topics standing as an easy and portable tool for testing, deployment, and distribution. The practical part of the course, i.e., a demo on Docker containers, has been attracting all the enrolled students over the past (two) years of the course.

The remaining three topics are then selected based on the background and focus of students' research. This is not an exhaustive process, but rather a flexible one that also considers the diversity of overall selected topics, the preference of lecturers, and also addressing several layers of the software stack.

After the selection of the six topics and corresponding lecturers is settled, the coordinator meets the invited lectures to present the goals of the course and the teaching methodology as discussed next.

B. Topic Structure and Teaching Methodology

SSP suggests a common structure and teaching methodology for the taught topics to achieve its goals. This is crucial to maintain some consistency given the diversity of topics and the expertise and teaching style of the invited lecturers. In general, each topic is composed of the three typical stages: Motivation, Theory, and Practice. The latter is of particular importance for SSP. In addition to the widening the knowledge of the student, the focus on the practical side has two benefits. The first is to give the student an overview of how real successful systems are built and used in practice. This shows the big picture of the system explaining how its modules interplay together, and the factors to be considered if his/her research targets a particular part of it. For instance, the overhead of message exchange patterns in a communication protocol, e.g., consensus, can be masked if the operations require access to persistent storage; whereas, it may not worth optimizing a local computational algorithm to gain some milliseconds if the algorithm involves communication delays, e.g., over the Internet. Therefore, understanding the big picture of systems can change how we think about a problems and its solution.

The second benefit is to educate the PhD students with best practices and lessons learned from the industry. In our opinion, several research works in academia focus on theoretical aspects not explicitly practical in real production systems. Therefore, we believe that educating the PhD candidate with some practical expertise is one way to bridge this gap.

C. Samples of Taught Topics

We explain the structure and methodology of SSP through giving two sample topics. The first is on NoSQL Databases that tackles the Amazon DynamoDB [4] as a successful system in production. The second is on Containers and focuses on Docker [1]. This is different from the former as it includes a hands-on tutorial on how to use Docker containers to make the students' research easy to test, deploy, and reproduce. Although a hands-on part is not research-focused as in the DyanmoDB case, the aim is to give the graduate student new skills useful for experimenting their research. In fact, since most SSP topics are implicitly distributed, Docker is important to reduce the complexity of building and deploying distributed systems, especially for non distributed systems students.

a) NoSQL Databases (Amazon DynamoDB): "*Who owns the information, owns the world!—Francis Bacon*". This represents the main motivation for this topic showing that data storage and retrieval are necessary to build information base, and thus leaving impact on all aspects of society and business. The class also demonstrates that a DB is a critical part of most systems and services nowadays. Then, the lecturer shows the overlap of databases with the different CS areas, especially those of the enrolled PhD students involving Databases, Systems, Software Engineering, Artificial Intelligence, Internet of Things, Security, etc. Our experience reveals that it is not too challenging to convince the students of the importance of databases and its relevance to each of them.

The lecturer then starts explaining the theoretical part through three main stages. The first starts by showing the need for a database. This starts at a very initial state where some data is presented in a raw file. The lecturer then challenges the students showing the importance of indexing, relations, and normalization for both data storage and retrieval. The lecturer then moves to focus on the foundation of schema-based (relational) databases and the added value of DBMS considering performance optimizations or features (integrity, transactions, triggers, etc.). The third stage starts by showing the limitation of schema-based databases when it comes to data heteroginity, system elasticity, scalability, and availability. The class covers the CAP theorem [2] and the ACID versus BASE tradeoffs [10]. The rest of the theoretical part focuses on the properties of schema-less NoSQL databases through explaining how it addresses modern data problems like: unstructured data, scalability, loose synchronization, elasticity, robustness. The lecturer shows that these features come at the cost of more complex programming logic due to the lack of built-in integrity, joins, and the SQL language schema-based databases provide. Finally, the class overviews the variants of NoSQL databases (Key-Value, Graph, Column, Document, Hybrid), how each of them excels, and the fact that SQL databases remain crucial for a substantial number of cases among them classical applications, OLTP, and OLAP.

The remaining part of the class digs in to Amazon's DynamoDB as a successful NoSQL database in production. The lecturer starts justifying the choice of this database for the topic. In this vein, DynamoDB is one of the very first NoSQL databases that used a synthesis of key techniques and features at once like DHTs, consistent hashing, versioning, vector clocks, quorums, sloppy quorums, anti-entropy-based recovery, etc. This made it a source of inspiration of several moderns NoSQL databases, e.g., Cassandra, Riak DB, etc. In terms of impact, the DynamoDB allowed Amazon's Shopping Cart Service [4] serve tens of millions of requests per day. The lecturer then moves to explaining the design decisions of DynamoDB summarized by building highly available always-writable database for latency-sensitive applications. Afterwards, the key properties and features of DynamoDB are presented covering: partitioning (consistent hashing), data versioning, replication, quorums, sloppy quorums, synchronization through Merkle trees, and anti-entropy-based membership. The final part shows how DynamoDB can be used explaining its simple API, provided with some examples. Unfortunately, this topic does not include code snippets since DynamoDB is not open sourced; however, the students are referred to open-source DynamoDB alternatives like DynomiteDB, Voldemort, or Riak KV.

b) Containers (Docker): This topic is more practical as it includes a hands-on demonstration. The class is divided into two parts. The first is more theoretical and starts by explaining the importance of Cloud Computing, in general, from the business and technical perspectives focusing on cost, elasticity, and efficiency. In particular, Containers are motivated through highlighting their extensive use today being an easy way for software packaging, portability, deployment, and being a lightweight virtualization alternative to virtual machines. The lecturer then shows how this topic intersects with the students' research interests and work. Interestingly, this part was easy in our case since everyone was eager to learn about the topic and try to understand how to make use of it in the future. The lecturer continues through introducing the needed background to understand how containers work. Again, this starts by introducing the idea of virtualization back to the early times of Cloud Computing, thus touching upon Cloud services types (SAAS, PAAS, and IAAS), public and private clouds, virtualization types and aggregation (a.k.a., clustering). The lecturer then digs in explaining the concept of containers focusing on *namespaces* and resource management via *cgroups*. This part ends by presenting the differences between bare-metal virtualization, virtual machines, and containers.

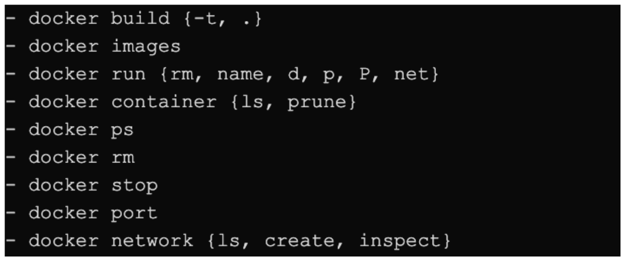
```
- docker build {-t, .}
- docker images
- docker run {rm, name, d, p, P, net}
- docker container {ls, prune}
- docker ps
- docker rm
- docker stop
- docker port
- docker network {ls, create, inspect}
```

Fig. 2. Docker commands and options extensively used in the demo of containers topic.

The second part follows a tutorial hands-on style. A script of commands is distributed over the students who execute it step by step. The tutorial starts by choosing a software to "containerize", e.g., a simple one like a web server or a more relevant one as Apache Spark. If the students are asked to install the platform, e.g., Docker, prior to the class, it is possible to cover the crucial parts like: prepare and build a software, build a container image from scratch, access a container through IP and Ports, build a docker network, and run multi-dockers. Fig. 2 depicts some Docker commands and options that have been extensively used in the demo, during the two past years, including some intentional bugs to teach the students how to debug an issue, and what is the reasoning behind it. As expected, the topic was the most interactive and enjoying topic for all, students and lecturers.

D. Assessment Method

The assessment method of SSP is directly inspired from the motivations presented in Section II: it must be practical and considers the diversity, time, and relevance aspects. Therefore, the student is required to propose, execute, and present a project that is related to one of the SSP presented topics and intersects with her own research topic. In our experience, this method is key to ensure the student is more involved in the course and significantly reduce the sense of distraction as long as the project "adds something" to her own research.

The process starts at the first lecture of the course where the lecturer presents an overview of the entire SSP topics, thus giving the students the spark to look up their project proposals. Then, the students are given a period (of maximum one month) to deliver a proposal through discussing it with one of the lecturers (preferably whose expertise are more relevant to the topic). The proposal shall not be over-killing nor naive, and thus estimated to 50 working hours. Finally, the projects are all presented in the final class that is a small symposium where each student has a time slot to present his work, run a demo, and answer questions posed by the lecturers and the students. The project and presentation/discussion are, respectively, given 60% and 20% of the final mark—the remaining 20% go for involvement and attendance.

Our experience together with the feedback of the students reveal the promising and interesting assessment method in SSP. Surprisingly, or maybe not, the discussions during the symposium (presentation day) have been vivid and engaging; many students asked questions and suggested ideas to the presenter. In addition, most projects have been useful to the students either through learning and building new tools to assist them in their research, or even to develop some seed research ideas or publish parts of them later. Finally, to address the few cases where students cannot develop a practical project, e.g., students with pure mathematics background, the lecturer may resort to a more theoretical project, e.g., a study to compare a presented successful system to another. However, this option should be avoided if possible.

IV. Feedback and Execution Challenges

To give a concrete indicator to the success and impact of SSP, we give some concrete examples on the acquired skills and tangible outcomes of the course. We also touch upon the feedback of the enrolled students and the programme's scientific committee. We finally present some of the course execution challenges we faced.

a) Acquired skills and tangible outcomes: As a practical graduate course, SSP focues on the tangible experiences and skills a graduate student aquires throughout the project, and how the latter supports her thesis or research topic. These expected oucomes are required to be explictly arcticulated in the project proposal and delivered material. Our observation was that the majority of students found this course an opportunity to bootstrap the practical side of their thesis. This was expected for a first-year graduate student. In some cases, the outcome was more significant than others, and thus appeared in peer-reviewed publications or sparked new practical ideas. We give three particular examples for students with different research interests, e.g., Security, Distributed Computing, and Big Data, respectively.

The first project is a cloud computing security solution (called "SafeNoSQL") that provides modular security techniques for NoSQL databases. This work eventually lead to a peer-reviewed publication [8]. In the project, the student built *SafeDynamo*: a proof-of-concept prototype to demonstrate how the modularity of this solution improve its applicability to diverse NoSQL Dynamo-like databases. A second example is *Tricks* [5]: a benchmarking tool for deploying and experimenting distributed computing protocols on Kubernetes [7]. The goal of Tricks is to make running real experiments as simple as defining the experimental settings (e.g., network topology, number of nodes, operation size, etc.) in a *YAML* file. This is enough to automatically create the needed distributed network, run the benchmark, and report the results in graphs. A third (not published) example focused on comparing the accuracy and overhead of running Spark or Hadoop ecosystems in a cluster versus distributed container-baised deployments. Although this work was not research-intensive, the lectueres acknowledged the administration skills acquired throughout the project to deploy and run such a complex distributed system. These examples (among others) gave an indication that the course achieved its goals in supporting graduate students with useful skills for their research despite having different backgrounds. This resulted in tangible outcomes as in the case of [8] and [5].

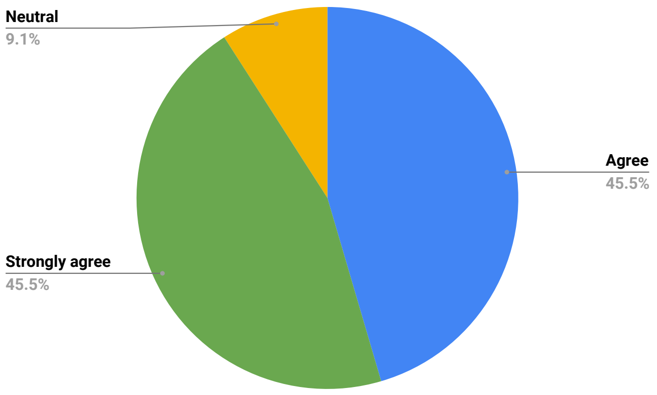

Fig. 3. The agreement on the proposition "SSP is the most useful course I took at MAPI (excluding those of my own area)?".

b) Student feedback: Being the target of the course, we took the feedback of the students during the two taught years via an anonymous questionnaire. The purpose was mainly to understand the weaknesses and strengths of the course and improve upon it for the next years. All the enrolled eleven students in SSP contributed to this questionnaire. The students had diverse backgrounds, spanning Distributed Systems, Databases, Mathematics, Artificial Intelligence, High Performance Computing, Security, etc., which is important to get fair results. The results of the questionnaire showed that the average *Likert* scale rate given to the course was 8.3/10. As expected, this was also justified through the students' comments through recommending the focus and assessment model of the course. In particular, all students have considered the assessment method practical and convenient; however, they suggested it to be more systemized (e.g., suggest some projects a priori and schedule follow ups).

On the other hand, the Likert rate of the relevance of topics was 8.6. As previously mentioned in Fig. 1, the greater majority of students recommended 90% of the selected topics to be taught in the following year. Other answers demonstrated that most students liked more the topics that include hands-on tutorials. In our experience, these numbers are considered excellent given the challenges mentioned above.

To understand the strengths of SSP over other courses within the PhD programme, we asked this question "SSP is the most useful course I took at MAPI (excluding those of my own area)?". Interestingly, the results came as depicted in Fig. 3, where more than 90% percent of the students over the two years affirmed the proposition, although less than one third are "systems" students.

c) Scientific committee feedback: The doctoral programme scientific committee often adopts successful PhD programme models followed in well-known universities, especially CMU [3]. Consequently, the committee recommends course proposals that are taught or inspired from leading worldwide universities. Despite being a novel course that has a little chance on this front, SSP has been approved during all the three recent academic years it had been proposed. Fortunately, this approval has been followed by the selection of the needed number of students to run the course during the two past years.

d) Execution challenges: Since the lecturers' feedback is consistent with that of the students, and the desired objectives of the course are considered successfully met, we rather focus on the challenges we faced during the execution of the course.

The main challenge is referred to the centralization of the course, which imposes more load on the coordinator being the principal lecturer, whereas the other lecturers are invited on a yearly basis. For instance, the coordinator's responsibility appears at a very early stage of the preparation of the proposal, presenting it in the selection symposium, reaching out the interested students to confirm the selected list of topics, looking up corresponding lecturers depending on the selected topics, synchronizing topics, moderating the projects' selection, etc. Nevertheless, we believe that these extra overheads are justified as a price for the dynamic and flexible nature of SSP.

Another related challenge is the coherence across different topics. This is resolved through short meetings across lecturers to identify the best order of classes and reduce the overlap. Furthermore, consistency in the teaching methodology across topics is another challenge that the coordinator shall emphasize and maintain before the invited lecturers prepare the material. In our experience, this is not an easy task especially when the lecturer is used to another teaching methodology and cannot easily stick to SSP's methodology; obviously, it is harder to impose a strict teaching style on an invited lecturer—especially senior ones.

Finally, the material of SSP's topics is more practical which requires special preparation unusual in fundamental or classical academic teaching. Although this is manageable, care should be taken when the time is short to prepare the material from scratch, otherwise the lecturer will end up using patches from previous slides. In the past years, we tried to advance the more standard topics taught by the principal lecturer, thus giving more time for invited classes' preparations.

V. Conclusions

Successful Systems in Production is a novel graduate systems course that mainly targets the practicality, diversity, and relevance requirements and challenges often exist at the PhD level. The selected topics of the course are broad enough to cover many research backgrounds, each topic targets the practical part of a system in particular, and the assessment method is agile to respect the diversity of students' backgrounds and exploit the time of the course to serve their own research. Based on our teaching experience depicted in this paper, we recommend this course to graduate teaching at other universities that exhibit similar conditions. This does not necessarily suggest to adopt the entire course model though. For instance, the methodology used is generic to be applied to any practical course whereas the assessment methods can easily be applied to have the graduates make better use of their time. All in all, we believe that future researchers should know how "successful systems" work "in production", whether they pursue their research career in academia or industry.

Acknowledgments

The author would like to thank the following invited lecturers for their valuable discussions and comments that lead to shaping this course: Rolando Martins, Jose Maria Fernandes, Vitor Santos Costa, and Ines Dutra.

References

[1] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[2] Eric Brewer. Cap twelve years later: how the. *Computer*, (2):23–29, 2012.

[3] Carnegie Mellon University Computer Science Department. The computer science ph.d. program at carnegie mellon university. https://csd.cmu.edu/sites/default/files/CSD-PhD-Handbook-2018-19.pdf, 2018. Accessed: 2019-08-29.

[4] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In *ACM SIGOPS operating systems review*, volume 41, pages 205–220. ACM, 2007.

[5] Vitor Enes. Tricks. https://github.com/vitorenesduarte/tricks, 2017. Accessed: 2019-10-05.

[6] EPFL. Edic course catalogue & registration. https://www.epfl.ch/education/phd/programs/edic-computer-and-communication-sciences/edic-course-offering/, 2019. Accessed: 2019-08-29.

[7] The Linux Foundation. Kuberntes. https://kubernetes.io/, 2019. Accessed: 2019-10-05.

[8] Ricardo Macedo, João Paulo, Rogério Pontes, Bernardo Portela, Tiago Oliveira, Miguel Matos, and Rui Oliveira. A practical framework for privacy-preserving nosql databases. In *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pages 11–20. IEEE, 2017.

[9] MAP-I. Map-i doctoral programme in computer science. https://mapi.map.edu.pt/, 2019. Accessed: 2019-10-05.

[10] Dan Pritchett. Base: An acid alternative. *Queue*, 6(3):48–55, 2008.

[11] MIT Electrical Engineering & Computer Science. Eecs graduate subjects. http://www.eecs.mit.edu/grad-areas/2-ai/subjects.html, 2019. Accessed: 2019-08-29.

[12] Northeastern University. Computer science graduate courses. https://www.khoury.northeastern.edu/academics/courses-fall-2018/, 2019. Accessed: 2019-08-29.

[13] Stanford University. Phd computer science requirements. https://cs.stanford.edu/academics/phd, 2019. Accessed: 2019-08-29.

[14] ETH Zurich. Doctoral and post-doctoral courses. http://www.vvz.ethz.ch/Vorlesungsverzeichnis/sucheLehrangebot.view?abschnittId=81372\&semkez=2019W\&ansicht=1\&lang=en\&seite=1, 2019. Accessed: 2019-08-29.