Considerations and Technical Pitfalls for Teaching Computational Thinking with BBC micro:bit

Considerations and Technical Pitfalls for Teaching Computational Thinking with BBC micro:bit

Markus Tyrén
RISE Interactive
Gothenburg, Sweden
niklas.carlborg@ri.se

Niklas Carlborg
RISE Interactive
Gothenburg, Sweden
markus.tyren@ri.se

Carl Heath
RISE Interactive
Gothenburg, Sweden
carl.heath@ri.se

Eva Eriksson
Aarhus University &
Chalmers University of Technology
evae@cc.au.dk

ABSTRACT
As many countries are about to make changes in the primary school curriculum by introducing computational thinking, new methods and support for teachers is needed in order help them develop and adapt teaching materials. In this paper, technical pitfalls and other considerations for designing teaching materials with the microcontroller BBC micro:bit are presented. The results are based on a series of 21 workshops in different parts of Sweden aiming to investigate what is important to consider when designing teaching materials with the BBC micro:bit for training Swedish primary schools students computational thinking skills. The contribution of the paper are a number of identified considerations that can be helpful for teachers when designing exercises and planning for teaching computational thinking with the BBC micro:bit.

Author Keywords
BBC micro:bit; teaching; computational thinking; programming; school.

CCS CONCEPTS
• Human-centered computing → Interaction design; • Applied computing → Education.

INTRODUCTION
Many countries are faced with rapid changes in the primary and secondary school curriculum in order to incorporate computational thinking (CT) as part of the 21st century skills. Sweden is no exception, where these changes are expected to be implemented by the autumn of 2018 (see Table 1) – leaving little room for further education of all teachers and for development of teaching materials [4]. Primary school teachers face a huge change when computational thinking is introduced in the curriculum [1]. This is particularly difficult since most primary school teachers are generalists, they teach a whole class in most subjects having broad knowledge rather than being specialists within a certain subject. There is a massive shortage in teachers with experience and knowledge within the computing area, the delivery of programming in class is a big challenge and might be hampered severely if teachers is not educated fast enough [2]. As there is no or very limited resources and time to educate teachers in CT, new methods have to be developed to help teachers in an efficient manner.

Table 1: An English translation of the new curriculum changes in Sweden

Programming	Introduced in multiple subjects in primary school, especially in technical and mathematical subjects
Digital tools	Digital texts, media and tools
Critical thinking	Strengthening the critical thinking skills
Systems thinking	Ability to use and understand digital systems and services
Creativity	Ability to solve problems and realize ideas into action in a creative fashion using technology
Impact	Develop an understanding for the impact digitalization has on the individual and society

In the UK the transition to more programming in school started off in January 2012 when The Royal Society published a highly-rated article with recommendations to reintroduce computer science (CS) in schools [6]. Up until then they had been teaching information and communication technology (ICT) but with an increasingly declining reputation over the last couple of years. ICT in contrast to CS focused more on the usage and software rather than the creative and underlying principles of computing. It was not long before the department of education declared the ICT curriculum to be re-written and in its stead officially reintroduce CS teaching in schools again. With this change, several issues were brought to surface. For instance how will the primary schools handle the fast pace of these changes, and how will they make sure that there are enough teachers with the right knowledge? As part of the Make It Digital initiative in 2015, BBC has together with Microsoft, Samsung and other partners, developed the BBC micro:bit

FabLearn Europe'18, June 18, 2018, Trondheim, Norway

ACM ISBN 978-1-4503-5371-7/18/06…$15.00
https://doi.org/10.1145/3213818.3213829

for use in computer education [3]. Every year seven child (age 11-12) in the UK schools receive one of these small computers, or microcontrollers, that can be programmed and customized.

Inspired by the UK, a project was initiated in order to explore how BBC micro:bit can be adapted to aid in the Swedish curriculum transition. This paper reports from a study in a collaboration between researchers at a university and a national research institution aiming to investigate how Swedish teachers can be supported in the transition to the new programming curriculum. The main research question is: *What is important to consider when designing teaching materials with the BBC micro:bit for training Swedish primary school students computational thinking skills?* The contribution of this paper is a set of considerations and technical pitfalls when introducing a new technological platform, such as the BBC micro:bit, for training computational thinking in Swedish primary schools. These considerations are based on non exhaustive empirical design research and be limited to the BBC micro:bit platform and the Swedish school context.

BBC Micro:bit

The BBC micro:bit was developed as a part of BBC's Make it Digital Initiative in 2015. It aims to inspire young people to get creative in the digital world, developing core skills in STEM subjects and produce a new generation of inventors and makers. The BBC micro:bit is a small computer, or microcontroller, that can be programmed and customized in order to bring ideas to life [3]. Displaying your name, or making it blink can be coded in seconds even if the user is totally new to programming. BBC micro:bit can also be connected to other devices or sensors and can complement other hardware like Arduino and Raspberry Pi, it works as a great springboard to more complex learning [3]. Key features include:

- 5x5 LED matrix display,
- Two programmable buttons,
- Accelerometer that can detect movement,
- A built-in compass to sense direction,
- The ability to sense temperature and light levels,
- Bluetooth Smart Technology to interact with other micro:bits and mobile devices,
- Five Input and Output (I/O) rings to connect the micro:bit to devices or sensors using e.g crocodile clips.

For ensuring that the BBC micro:bit becomes successful, it is considered important that all partners involved work closely with both teachers, educators and schools to provide resources and information supporting the curriculum.

There are multiple ways of programming the BBC micro:bit, the scope of this project has however been limited to only focus on the Microsoft MakeCode micro:bit editor, see Figure 1. The Microsoft MakeCode micro:bit editor is a free to use online JavaScript/Blocks editor for programming the BBC micro:bit. This means that it runs in the web browser and hence is cross platform compatible, both on different web browsers but also across different operating systems, such as OSX, Windows, iOS and Android. This also implies that an internet connection is required for using the editor. Technically the Microsoft MakeCode editor for BBC micro:bit can be used offline as the application gets cached locally but only if an online compilation has been made first. Either way an internet connection is required at some point.

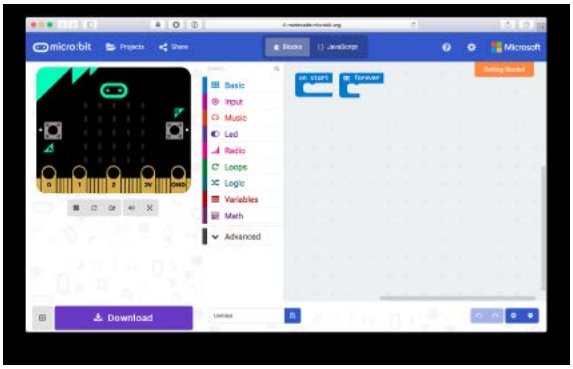

Figure 1: One of the editors in which programs can be created for the micro:bit

The term block editor refers to the puzzle like interaction where the user builds their programs by snapping different function blocks together to create a programs behavior. The editor allows the user to code both with blocks as well as JavaScript code. It provides the possibility to switch back and forth between these on the fly to translate from one to the other.

The user interface of the editor as seen from left to right consists of a simulator, a section of available blocks and an area where the user is to drag blocks and build their programs. At the top, controls are available for saving and loading projects, switching back and forth between block or JavaScript mode, and some more advanced settings. The button for downloading the created program is located at the bottom.

For creating a simple program, the user begins with finding the desired blocks in the middle column folders and drags them onto the block building area on the right. Blocks snapped into the "on start" block will only run once, and blocks snapped into the "forever" block will repeat indefinitely. The functionality of the program can then be evaluated with the simulator on the left. To download the program the user clicks the download button on the bottom left, and transfers the obtained file to the BBC micro:bit flash drive via a USB cable.

METHOD

In order to investigate fundamental theoretical and empirical knowledge concerning BBC micro:bit and using technology to aid education in Sweden, we applied a Human-Centred Design approach. Initially, we gathered information about how teachers and students perceive the introduction of

programming in school through secondary research. Additionally we familiarized us with the BBC micro:bit platform and the editor, and developed some initial exercises. We used insights from evaluated data to create new content iteratively throughout the project. The implementation of the design work for the current objectives mainly comprised of field tests, namely 21 workshops in various school environments (grade 4-6) in Sweden on several occasions. While field testing the teaching prototype, feedback from the testers and the process was collected for later evaluation. Evaluation and analysis of the implementation and gathered data were carried out at the end of each prototyping cycle, using an inductive approach. Based on this, insights were extracted and fed into the planning of the next cycle. In the final phase, all the data was analyzed with affinity clustering in an attempt to reach a theoretical result. We found insights and valuable information for teachers and made them into a bundle of considerations and technical pitfalls when working with BBC micro:bit.

RESULTS

The result of this study consists of a number of practical considerations when designing teaching materials for computational thinking in primary school with the BBC micro:bit. These results are derived from empirical research done in a Swedish primary school context with the aim to train students computational thinking skills using the BBC micro:bit.

BASIC TOOLBOX

The Basic toolbox relates to a set of fundamental programming concepts that were found useful for students to be introduced to, prior to working with programming exercises. The basic toolbox contains the programming concepts: algorithms, loops, randomness, logic, variables and debugging, see Figure 2. It was found beneficial to introduce and practice these concepts in parallel to each other, rather than to work with each one of them separately, in series. This due to the perceived difficulty to create interesting exercises based on only single programming concept. What was seen as interesting exercises, were combinations of multiple programming concepts.

Algorithms - Algorithms refer to the sequencing of instructions to reach a certain desired behavioral outcome. It has been seen that students benefit from having been introduced to this concepts prior to start working with programming the BBC micro:bit. Algorithms are also mentioned in the Swedish government's policy changes regarding the documents that control Swedish primary and secondary school curriculum [4]. There are many possible approaches to introducing algorithms to students. The way it was introduced in this project was through having the students control each other's movements with instruction notes through an analog game field. This game later came to be called analog workshops.

Loops - Loops refer to the fundamental programming concept that allows certain instructions to be repeated multiple times. It has been seen that students benefit from having been introduced to this concepts prior to start working with programming the BBC micro:bit. Combining loops with other tools from the basic toolbox was seen as necessary to create stimulating and useful exercises with the BBC micro:bit. It was found useful to teach loops in the context rather than simply as an abstract concept.

Randomness - Randomness refers to the basic programming function of a random generator, which is frequently used in many types of programs. It has been seen that students benefit from having been introduced to this concepts prior to start working with programming the BBC micro:bit. Having knowledge about how to use the random function when working with BBC micro:bit proved beneficial when starting to learn programming. Since many of the beginner projects use elements of randomness as a function in their code, such as roll a dice or rock-paper-scissors, being familiar with it became important to complete those programs.

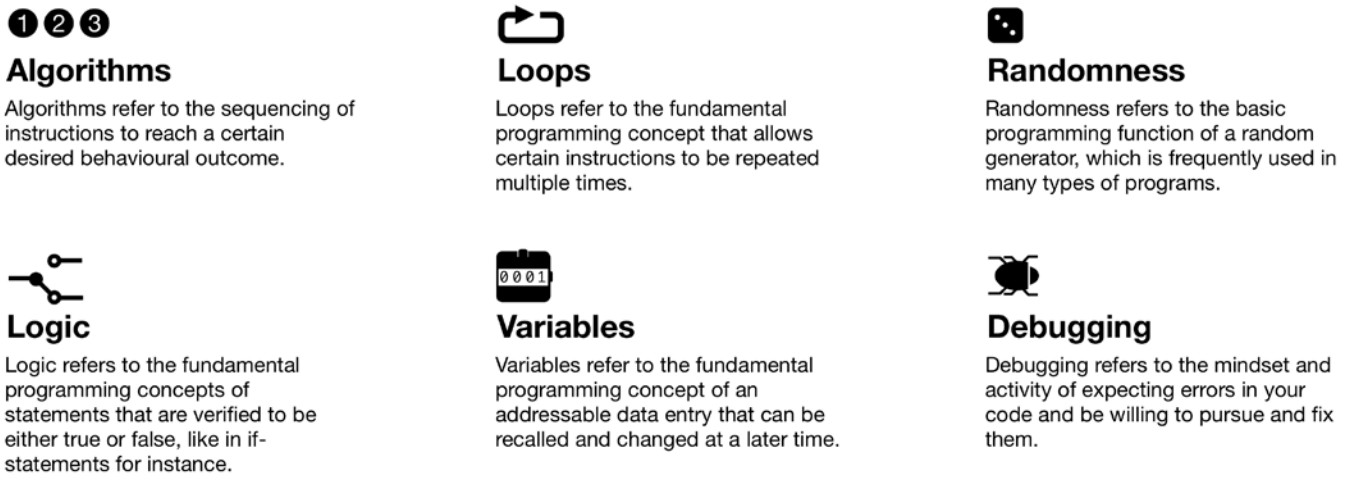

Figure 2: The basic toolbox

Logic - Logic refers to the fundamental programming concepts of logical statements that are verified to be either true or false by a program. This for instance is a central part of if-statements. It has been seen that students benefit from having been introduced to this concept prior to start working with programming the BBC micro:bit. The way logic was introduced to students in this project was through having the students control each other with instructions through an analog game field. Some of these games required the students to create alternative instructions depending on the outcome from a random event.

Variables - One truly fundamental concept of programming is the one of variables, as without variables and other data structures there is no way for a computer to store information. This information can be of different types: values, names, and might prove difficult to grasp at first for students. Nevertheless it has been seen that students benefit from having been introduced to this concept prior to start working with programming the BBC micro:bit. Many of the beginner projects involve the usage of variables, therefore it has an obvious place in a basic toolbox for students. The way variables were introduced to students in this project was through the metaphor of having a high score counter. As students moved through the game later known as analog workshops, they got to add the value of the tile they were standing on to their high score counter.

Debugging - Debugging refers to the mindset and activity of expecting errors in your code and be willing to pursue and fix them. Programming without ever encountering any errors, or bugs, is highly unlikely. Fixing errors can be time consuming but is an essential skill for programmers to learn. As students have been seen able to enter states of indifference or dejection when faced with errors, it is considered useful for students to have been introduced to this approach prior to start working with the BBC micro:bit.

Regarding block programming on the BBC micro:bit it rarely becomes a syntax or coding error due to the nature of blocks. The errors are more likely logical errors. Solving these errors is considered to be an important part of deepening the understanding for programming. There are some steps one can practice when confronted with a bug. Firstly it is a good idea to try and predict what the program should do, and step by step execute the code manually out loud to try and see where the error is. This way the problem often becomes apparent rather quickly. It was verified that debugging was practiced simply by working with the exercises. The role of the facilitator or teacher simply was to encourage debugging when the inevitable errors appeared.

TERMINOLOGY

Terminology is referring to the words used when teaching programming. As there are many new words and concepts that might be intimidating for students at first, it is useful to initially avoid using words such as variables, but rather attempt to convey these through words and concepts that already are familiar to the student. For example the concept of a variable can be described as a high score counter in a game, something that the student might already be familiar with. This does not say that variables are precisely high score counters, but it is a way of conveying the intuitive concept without introducing any new potentially scary words. It can still be encouraged that the students learn the correct terms for concepts, but it is a matter of easing them into it. From a teacher's perspective it can be hard to be aware of when one uses programming terms, therefore it is advised to pay close attention to the language one uses so that no new words are introduced without proper introduction first, preferably linked to previous knowledge. Some qualitative signs have indicated that it might be easier for a student to do something first and then afterwards learn the proper name for it, rather than first being introduced to a new word before learning what it is about.

TECHNICAL PITFALLS

Technical pitfalls refer to practical issues that have been identified to risk obstruct or fail the execution of a workshop. As it has been seen that students can enter states of indifference or dejection if these issues take up too much time, it is suggested to take measures and attempt to prevent them from arising in the first place. Three specific issues that have been observed are relating to: app-store passwords, internet connection and pairing mode bugs.

App-Store Passwords - In the context of running a BBC micro:bit workshop with iPads, the students are required to download the BBC micro:bit app from the app-store. This requires an app-store account, and some schools prefer to control what apps are being installed on their iPads with passwords. It is therefore recommended to obtain these passwords well in advance and work out a good way for these apps to be downloaded in class, or possibly even downloading the BBC micro:bit app to each individual device in advance.

Internet Connection - Unreliable internet connection was seen as another frequent source of frustration. As the BBC micro:bit editor is run through the web browser reliable internet connection is required throughout the workshops. Technically the Microsoft MakeCode editor for BBC micro:bit can be used offline as the application gets cached locally although an online compilation has to be made first. In cases where internet is slow or the connection dropped occasionally, students will get frustrated and precious learning time will be spent troubleshooting internet connections instead. In those cases where workshops rely on online material, such as instruction videos, the internet bandwidth plays an even more noticeable role, as video can be rather bandwidth heavy.

Pairing Mode Bugs - There were some issues identified regarding the pairing of BBC micro:bits to iPads. Firstly there is a certain procedure that is required to initially pair a BBC micro:bit to an iPad. This procedure can be rather tricky at first, as it requires the student to press three buttons in a certain order and remember a series of six numbers shown in

a rapid sequence. Students grasp the pairing process the fastest by being shown the full procedure and then try it themselves. Having students imitate this procedure in real time is not recommended, as it can lead to disorder in the class. Another issue one might encounter is a bug relating to sending a program (flashing) from a mobile device to a paired BBC micro:bit. This process requires the BBC micro:bit to be in pairing mode again, despite that this is not mentioned in the documentation. As this bug is not documented, it can be hard to solve and can hinder an entire workshop from progressing. Lastly there have been a few rare occasions where BBC micro:bits were impossible to put into pairing mode. This bug is resolved by simply having a computer nearby and flash any type of program from the computer to that BBC micro:bit via USB-cable. This way the BBC micro:bit gets reset and can be paired with an iPad again.

OTHER CONSIDERATIONS

Furthermore there were eight single insights from workshop interventions that were found useful, namely: tinkering, self instructing materials, stupid computers, end on a positive note, text based instructions, video bubble, editor navigation and awareness of dependencies, see Table 2.

Table 2: An overview of other insights and considerations

Tinkering	Allowing to freely familiarize with new content before exercise
Stupid computers	Clarify that computers are stupid and only do what they are told
Text based instructions	Instructions by other means are preferred
Editor navigation	An initial walkthrough of the editor might help students navigate
Self instructing materials	Alternative to providing individual help to large groups
End on a positive note	Ending a session sruggeling with a hard exercise can be demotivating, try to leave the session with a good feeling
Video bubble	Videos can decrease the social aspects in class, isolating students in a bubble
Aware of dependencies	When teaching is dependent on software, be aware of changes in software updates

Tinkering - Tinkering is allowing students to freely familiarize with new content before starting to work with exercises. This explorative approach without any goals or objectives was seen as a way for students to get outlet for any curiosity that might arise as they are introduced to new technology, platforms or concepts. To prevent frustration or confusion to arise, it is recommended to keep this initial tinkering short in time.

Stupid Computers - Stupid computers refers to making it clear to students that computers are simply following instructions and should not be considered as intelligent per se. In some cases we have seen students who expect that computers are smart just because they are computers. Clarifying this initially can potentially prevent some of these misconceptions.

Text Based Instructions - Purely text based instructions might not be the optimal way for conveying exercises. Giving instructions through other means can potentially be more successful. Students mostly ignore printed papers with instructions that have been handed out. Conversations wit students showed that they found written instructions incomprehensible, and prefer facilitators to answer their questions. Other means of giving instructions could also be through video.

Editor Navigation - Editor navigation refers to the need to understand how a certain programming editor works and how to navigate it in order to use it. It is easily overseen and can be considered trivial by someone familiar with it, nevertheless is it important for a first time user.

Self-Instructing Materials - Self-instructing materials such as instruction videos, have been seen as potentially useful in cases where student group sizes exceed the number of students that the facilitators are able to provide help to. Even if the class size is manageable self-instructing material allows students to work at their own pace and given that the students are ready for the exercises it can work as an offload for the teacher, which can focus their help where most needed.

End on a Positive Note - End on a positive note refers to concluding workshop sessions with an exercise that leaves the students in a positive state of mind, rather than leaving them confused or frustrated. Struggling with exercises might be important for the development of grit, the harder exercises should be placed in the middle of a session and allocate the end for easier ones that allow students to feel successful. From a facilitator point of view this means planning the timing of the session well, and never try to cram any exercise in the very last minute, just because you want to convey something that might not really have gotten across. Energy levels of the class is usually rather low in the end, and trying to force last minute teachings in here, seems to possibly make more harm than good.

Video Bubble - When using instruction videos as teaching material for an entire class, one ought to be aware of the potentially negative effects this might have on social aspects of the group. A classroom full of students watching different videos on their computers will get rather noisy, headphones are recommended. This however also has the effect of isolating students into their own video bubble. This might be positive for some, in terms of concentration, but it also removes many of the interpersonal social interaction that can be positive in a group. This way of providing video material is probably better suited for homework.

Awareness of Dependencies - When creating teaching materials dependent on software, one needs to be aware that it can be changed in future software updates. For instance if

one creates a set of instructions on how to navigate an editor that in detail refers to certain buttons, the names of these buttons may well be changed in future software updates. To avoid the risk of confusing students, it is therefore suggested to continuously verify that the teaching material with dependencies is up to date with the current state of the software. This was discovered as teaching materials, that intentionally had been color coded to match the BBC micro:bit editor, turned out to no longer match the colors of the editor at a workshop.

DISCUSSION

The research process was intended to be an iterative design process with a few well defined iteration cycles. The way it turned out however was that the sharp boundaries between these well defined iteration cycles got rather blurred. Out of 21 workshops 10 can be considered to be small but complete iteration cycles. The initial idea was to only gather insights about users, before starting to create workshop materials. But it turned out that we needed to create a workshop in order to observe the users in it. Hence it became somewhat of a chicken or the egg dilemma, and we ended up doing a bit of both in parallel. Still the first workshop interventions were more of introductory nature and mostly focused on gather qualitative data about the users. Some tweaking of the content and execution of workshops still occurred but the focus was not on improving the workshop design but rather to observe the behavior, needs and progress of the students in their first encounters with the BBC micro:bit, and in some cases even programming.

Many different technologies and development platforms are available and new ones are constantly being developed. Choosing what technological platform to invest in might be a very relevant question to educators, however this is not within the scope of this paper. This project uses the BBC micro:bit, as a given educational platform as the client stakeholders considered it to be affordable, already well spread in the UK, and having a lot of potential with its many onboard sensors. Furthermore there are multiple different editors available for working with the BBC micro:bit, thus in this project we only consider the Microsoft MakeCode BBC micro:bit editor, and specifically the block editor part of it.

Throughout the project we have related to the basic concepts of programming in our design process. These concepts permeate through all programming teaching activity and promote computational thinking and problem solving. Even though our results are for BBC micro:bit specifically there are many similarities that makes it versatile. Most of the platforms used to teach programming for primary school uses a block type editor, just like BBC micro:bit, why we believe that the results can be claimed to have a wider applicability even though quantitative data to support the long term effects of using such the results are so far non-existent.

CONCLUSION

Through an iterative design process, a total of 21 workshop interventions in class 4-6 were conducted in Sweden to collect qualitative data and gain insights about pupils and teachers interactions with BBC micro:bit teaching materials. The results is a set of considerations and identified technical pitfalls to give concrete answers to the research question: *What is important to consider when designing teaching materials with the BBC micro:bit for training Swedish primary school students computational thinking skills?* It is the hope of the authors that these results can be helpful for teachers developing teaching materials for computational thinking in the new Swedish curriculum.

ACKNOWLEDGMENTS

We thank all the students, teachers, and school leaders involved in this research. The research is funded by Vinnova grant nr 2015-02319.

REFERENCES

1. Rachel Charlotte Smith, Ole Sejer Iversen, and Rune Veerasawmy. Impediments to digital fabrication in education: A study of teachers' role in digital fabrication. *International Journal of Digital Literacy and Digital Competence (IJDLDC)*, 7(1):33–49, 2016.
2. Neil CC Brown, Sue Sentance, Tom Crick, and Simon Humphreys. Restart: The resurgence of computer science in uk schools. *ACM Transactions on Computing Education (TOCE)*, 14(2):9, 2014.
3. About the BBC Blog Head of BBC Learning, Sinead Rocks. BBC micro:bit, groundbreaking initiative to inspire digital creativity and develop a new generation of tech pioneers. http://www.bbc.co.uk/mediacentre/mediapacks/microbit/, 2016. [Online; accessed 2 March-2018].
4. Regeringskansliet. Stärkt digital kompetens i läroplaner och kursplaner - regeringen.se. http://www.regeringen.se/pressmeddelanden/2017/03/starkt-digital-kompetens-i-laroplaner-och-kursplaner/, May 2017. [Online; accessed 2-March-2018].
5. Jeannette M Wing. Computational thinking. In VL/HCC, page 3, 2011.
6. Steve Furber et al. Shut down or restart? the way forward for computing in uk schools. The Royal Society, London, 2012.