**Teaching Plan 2012-2014 - Core Syllabus and Options for SL - Overview**

This starts with an overview (p 1-3). A detailed plan follows on p. 4 , but is not finished.

This plan is compatible with the book Computer Science Illuminated , which starts at the top (overview), works it's way down to the bottom (details), then builds back up to the top (system design and expansion).

**(A) Studying Existing Systems**

- **History** A brief history of Computer Science
- **Finished Systems** – what have students seen and used – PCs, Cell Phones, Web-sites, Databases … **[1.1]**
- **Software and Applications** **[1.2]**
  OS vs Applications, standard features and uses
  Users and Usability
- **Hardware** **[2.1.1-8]**
  Standard Devices – input, output, storage, processing
  screens, keyboards, disks, RAM, CPU
- **Digital Computing** **[2.1.9-13]**
  binary, logic circuits

**(B) Understanding Automation – Computational Thinking**

How do existing systems accomplish the "work" they do? Investigate.

- **Decisions –** thinking logically, rules (real world vs systems) **[4.1.4]**
- **Procedures** – breaking problems into pieces (modules) **[4.1.1]**
- **Planning –** thinking ahead, data and testing, IPO **[4.1.9]**
- **Abstraction** – representing data and information as data, data-structures, rules and modules **[4.1.17]**
- **Concurrency** – vs iteration, breaking problems into pieces to be processed in parallel, distributed processing, networks **[4.1.14]**

**(C) Implementing Automated Systems** (from a finished design)

- **Using Production Tools** – depending on the chosen option, this will be databases, web-sites, OOP or modeling, so this might be web-site builders, data-base tools, spreadsheets and other modeling tools, or an OOP tool like Scratch or some other high-level programming tool
- **Introduction to programming (first experience)** **[4.3.6-9]**
  - input / output
  - variables and calculations
  - decisions (branching)
  - loops for repetition
- **Nature of Programming Languages** **[4.3.1-5]**
  Compare the language used above to another similar language, e.g. compare Java to JavaScript, Basic to VBA, Scratch to JS

**(D) Networks and Communication**

- **Study Existing Networks –** Internet, cell-phones, e-mail, wireless devices **[3.1.1-5]**
- **Data transmission** – protocols, packets **[3.1.6-11]**
- **Wireless networking –** hardware, protocols, security **[3.1.12-16]**

**(E) Sharing and Re-using Data**

- **Programming (part 2) -** data-collections, arrays, sub-programs **[4.3.10-13]**
- **Sharing data in a LAN** – text-files, peer-to-peer **[extra]**
- **Planning programs -** flowcharts, prototypes **[extra]**

It's unclear how much time each section should take – that will depend on the students' backgrounds. Perhaps 4-6 weeks each.
I'm inclined to make one quick pass through A,B,C,D, maybe in 3 months, and then go through A,B,C,E again in more depth, another 2 months.

**Teaching the Option – finishing Year 1 - Overview**

The Option would occupy the rest of the time in the first year of the course. So SL students should have completed most of the syllabus in one year.

For the OOP option, it makes sense to teach the option AFTER (E) above. Then while teaching the option, making occasional references and connections back to the rest of the syllabus.

For the Database, Web and Modeling options, it probably makes more sense to make references to the option during the original syllabus coverage, and then work on the specific option details after (E) above. Here is a brief plan for the extra option work for SL – the parts after (E) above.

| *Databases* | *Modeling and Simulation* | *Web Science* | *OOP* |
|---|---|---|---|
| **Example Data/Info Systems [A.1]**<br>- compare data-driven web-sites to static web-sites<br>- study a DB system at school - e.g. attendance, scheduling, library – and identify features that make it different from a web-site or a document<br>- discuss issues of multiple-access and distributed processing vs centralized processing<br><br>**Relational Databases [A.2]**<br>- study existing databases at school and define essential features and vocabulary<br>- discuss the essential issues in database design<br>- build small sample databases using a relational DB tool (MSAccess)<br>- discuss scalability, reliability, usability<br>- discuss higher level design issues, normalization, keys, relations<br>- construct queries & reports | **Basic Modeling [B.1]**<br>- investigate some existing models<br>- outline standard situations where models are used<br>- data and variables<br>- formulas and rules<br>- data-collections<br>- test-cases<br>- assessing effectiveness<br>- revise model, improbe correctness<br><br>**Simulations [B.2]**<br>- compare models to similutions<br>- rules that connect with reality<br>- data-representations<br>- compare various forms of data representation and organization<br>- construct simple models and compare them<br>- construct simple simulations and compare effectiveness<br>- practice changing rules, formulae and algorithms<br>- assess reliability and correctness<br>- discuss using sims for predictions | **Creating the Web [C.1]**<br>- Web infrastructure – protocols, technologies<br>- Web basics – URL, DNS, IP, HTML, packet switching, routing, etc.<br>- What is in a web-page?<br>– HTML<br>– tags vs content<br>– JavaScript<br>– multimedia<br>- static vs dynamic vs data-driven web-pages<br>- browsers<br>- clients, servers, CGI<br><br>**Searching the Web [C.2]**<br>- search engines – several examples<br>- crawlers, spiders, indexing<br>- meta-tags, text content<br>- search engine optimization<br>- web metrics<br>- algorithms and AI in crawlers and search engines<br>- search engine growth | **Objects as a Concept [D.1]**<br>Teach this with little Java detail, with simple programming examples<br>- classes vs objects<br>- choosing objects/classes for a problem<br>- choosing objects for data<br>- choosing objects for behaviors<br>- roles and dependencies of objects<br>- UML for design & documentation<br><br>**Features of OOP [D.2]**<br>Now teach more Java details<br>- encapsulation, especially for data objects<br>- polymorphism, especially for active objects (with methods)<br>- inheritance, for both data objects and active objects<br>- libraries and class hierarchies<br>- advantages and disadvantages<br><br>**Program Development [D.3]**<br>Now teach all the Java details.<br>This is probably connected to the |

| | | | |
|---|---|---|---|
| **Database Management [A.3]**<br>- safety and security<br>- documentation and training<br>- reliability and efficiency<br>- communication, connectivity, compatibility and accessibility<br>- data-mining, data collection, validation and verification | **Visualization [B.3]**<br>- investigate 2D visualization<br>- investigate 3D visualization<br>- outline memory needs of 2D and 3D visualizations<br>- discuss time and memory requirements of 3D animation | **The Evolving Web [C.3 – C.4]**<br>- mobile computing<br>- peer-to-peer<br>- ubiquity<br>- new and future uses<br>- distributed systems<br>- social networking<br>- cloud computing<br>- effects of new developments on individuals and organizations<br>- what enables/disables progress? | Project, so examples should include relatively large programs (several hundred lines of code).<br>- varaiables and parameters<br>- methods – acccessor, mutator, constructor, signature, return value<br>- modifiers – private, public, static..<br>- connect OOP to core programming concepts, e.g. using OOP syntax in relatively simple programs<br>- OOP helps internationalization<br>- OOP helps testing |

**Year 2 - The Project , Finishing HL and Review**

| | | |
|---|---|---|
| **Year 2**<br>At the beginning of year 2, students begin their project. This would involve about 4 weeks of investigation to get started. After that the students develop the product. Students must have frequent contact with their advisors during the entire project period.<br>There is ample time for SL students. They can make a 3rd pass through the entire syllabus, nailing down vocabulary and essential topics, and reviewing past-papers.<br>HL students still need to finish the HL extensions, which means they will have less in-class time to work on projects and less in-class time for review. | **Starting the Project (1 week)**<br>Presumably most students will use the skills learned in the Option to create their project.<br>- choose a topic area<br>- identify a problem<br>- choose an advisor<br>- write an intitial proposal for teacher approval<br><br>**Planning the Project (1-2 weeks)**<br>The plan should include<br>- preliminary list of tools to be used<br>- decomposition in tasks<br>- timeline for completion of tasks | **Design the Product (1-2 weeks)**<br>- create a prototype<br>- discuss the prototype with the advisor<br>- modify and repeat<br><br>**Develop the Product (4-6 weeks)**<br>The tasks and time requirements will be different for various projects. This includes testing and debugging.<br><br>**Documentation (2 week)**<br>- discuss the product with the advisor<br>- create video(s) of the functioning product<br>- evaluate and recommend improvements<br><br>**Package and Submit the Project (1 week)**<br>Teachers should help with this, ensuring that all files are in the proper format and function correctly. |

**Existing Systems (detailed plan)**

| ***General Area*** | ***Experience/ Essential Questions*** | ***Investigate / Explore*** | ***Core Assessment*** | ***Comments*** |
|---|---|---|---|---|
| Systems | Are all computer systems the same? If not, what are some common types? | Run standard software and look at common uses of standard computer systems:<br>Web-site, database, word-processor, spreadsheet, communication (email, SMS...) | **1.1.1** | |
| | What does "compatibility" mean? To what extent are various systems compatible or incompatible? | Compare/convert file-types and test the compatibility:<br>HTML vs wp-doc,<br>graphics types,<br>animation vs video,<br>database vs spreadsheet<br>... | **1.1.3**<br>**1.1.6** | |
| | How are computer systems chosen? What purpose do they serve? | Investigate computer systems that are used around the school, at home and at parents' workplaces:<br>Windows vs Mac platforms<br>Office programs<br>Educational software<br>Communication systems(SMS, email), .... | **1.1.1**<br>**1.1.5** | |
| | What are differences between local applications, "apps" and online systems? What is Software as a Service? | Find an application, and app and an online system that offer similar functionality. Install all three and compare their features and performance, for example:<br>MS-word, Google docs, a notes app | **1.1.4** | |
| | What are standard ways to provide support and documentation? | Choose one computer system and find various types of support and documentation for that system. Assess their effectiveness. Compare a system with poor documentation to one with comprehensive documentation. | **1.1.8**<br>**1.1.9** | |

| | | | | |
|---|---|---|---|---|
| Systems | How are new systems installed? What problems must be managed? | Install a new piece of software for a specific task, e.g. a graphics editor. Compare it to software that was previously used.<br>Discuss students' experiences with new devices at home, e.g. getting a new cell phone or a new PC. What problems occurred? How are they solved? | **1.1.2** | |
| | How can user training be implemented? What are good and bad methods? | Read stories about problems that happened in businesses. Compare the stories to more local problems around the school and home.<br>Compare "for idiots" books to more standard books. | **1.1.10** | |
| | How do we ensure that computer systems work reliably? | Practice making backups, deleting data and then restoring it from backup.<br>Practice changing passwords.<br>Try "cracking" passwords on a test account.<br>Investigate backups of online services.<br>Discuss the schools "backup strategy" and compare it to students' personal backup strategies. | **1.1.11**<br>**1.1.12**<br>**1.1.13** | |
| | How can hardware and software be tested? | Try out some standard tools, like:<br>benchmark software, system process manager, | **1.1.7** | |
| | How and why are updates managed? | Investigate the following updates:<br>OS (Windows) , browser , virus scanner<br>How often do they occur? Why are they necessary? What improvements are achieved? | **1.1.14** | |
| | How are PCs and organizational computer systems different? | Interview an IT support employee at school. Ask them to compare the problems they solve at school to the problems they solve at home. | **1.1.1 - 1.1.14** | |

**Software / Applications (detailed plan)**

| ***General Area*** | ***Experience/ Questions*** | ***Investigate/Explore*** | ***Core Assessment*** | ***Comments*** |
|---|---|---|---|---|
| SW types | What is the difference between OS and applications? | Compare features of PC OS to features in a smart-phone. Find tasks that work better on one device than the other. | **2.1.6** | |
| SW standards | What comprises a "complete" or "standard" set of software? | Read ads for new PCs, smart-phones, etc and compare software packages. | **2.1.7** | Avoid discussing prices - use middle cost examples rather than the extremes |
| Usability and HCI | What do various users need or want from software? | Interview a variety of types of users:<br>young students vs old students<br>students vs teachers vs administration<br>home users vs office users (parents)<br>producers vs consumers | **1.2.12**<br>**1.2.15** | |
| | How can we assess usability? | Compare ease of use of various devices:<br>- telephone vs cell phone vs smart-phones<br>- PC vs netbook vs tablet<br>- Wireless vs wired devices<br>Read "comparison" articles in magazines<br>Suggest usability issues and make a possible check-list to rate usability | **1.2.12**<br>**1.2.13** | |
| | How can usability be improved? | Investigate some software upgrades, plug-ins, patches, etc that might improve usability | **1.2.15** | Need to repeat this when discussing HW |
| | What special needs do various users have? | Investigate programs that increase accessibility, e.g.:<br>- One Laptop Per Child<br>- Indian $50 tablet<br>- rich vs poor<br>Investigate accessibility for the disabled | **1.2.14**<br>**1.2.16** | |

**Hardware / Components**

| ***General Area*** | ***Experience/ Questions*** | ***Investigate/Explore*** | ***Core Assessment*** | ***Comments*** |
|---|---|---|---|---|
| HW | What components should a computer system include? | Examine and compare :<br>PC , portable (e.g. tablet) , server<br>Identify the normal components:<br>Memory - RAM, ROM , Cache<br>Storage - disk, Flash, cloud<br>Processor - fast vs slow, low power | **2.1.2**<br>**2.1.5** | |
| | Why are SmartPhone interfaces different and/or similar to PC interfaces? | Identify similar and different features in PC interfaces and phone interaces. | **2.1.8** | |
| | How are memory sized, especially memory cards, measured? Why are the numbers so strange? | Students bring a variety of devices and extra memory cards and USB sticks to class.<br>Compare sizes and data-transfer-speeds. | **2.1.9** | Define: byte, kilobyte, megabyte, gigabyte<br>Define data-transfer-speeds (e.g. bandwidth) in both megabits and begabytes |
| Fundamentals | How is data actually stored inside a computer? | Use a hex-editor to examine the contents of various file types - text, documents, html page, graphics. Try making meaningful changes in the files, after learning the meaning of binary and hexadecimal. | **2.1.10** | Define: bit, bit-map, binary code, ASCII, binary, hexadecimal, decimal |
| | What can be changed or upgraded to make a computer faster? | Compare simple tasks on various machines, like copying files, reformatting a document, converting video formats, uploading and downloading files.<br>Read **advertisements** and discuss which upgrades might be sensible, and how they can improve performance. | **various** | |
| | Where do speed ratings come from? | Examine processor speed ratings in tables.<br>Run a couple of standard benchmark tests. | **various** | Define the term "mobile processor" |

**Notes to Readers**

I'm not sure the time allocations are very exact, but I think they are roughly correct.

Here are some general ideas:

1. Comp Sci is now in group 4. We don't really do "experiments" as such, as we are not "discovering" natural laws. We are not actually an "experimental science". However, observation and investigation should be part of the course. Topics should start with (or include) some sort of experience and investigation. Reading is still useful, but it would be a shame to teach this as a textbook course, just learning vocabulary. Although vocabulary is necessary, I'm pretty sure it will be difficult to pass exams without a substantial background in investigation and problem solving. Hence, I'd prefer to outline a set of investigations which cover the syllabus, rather than outlining topics as described in the course guide.

2. In the old course, programming occupied a central role. Along with that, the size of the IA dossier forced us to concentrate on **creation and production** of programs. Now in the new course we can spend more time observing and investigating, as the production requirements are much more modest. "Understanding" should now be as important as "creating".

3. The reduced role of programming is compensated by the introduction of "computational thinking". Although students enjoy programming and the satisfaction that comes when a program runs successfully, it's important now that the students can also **think about** and **discuss** problems and solutions without necessarily implementing them. Presumably this will be the major expectation in exams. Hence any teaching plan should emphasize computational (algorithmic) thinking in a variety of contexts, including but not limited to programming or other use of tools.

4. For me, the major question throughout the course is: "What is happening inside computer systems and how is it happening?" Computer systems are our experimental playground, just as living creatures are the arena for biologists and the laboratory is the place for doing chemistry. Understanding data representations, algorithmic processes and automation is the key to understanding how computer systems function. Investigations help us to ask questions, then reading and lectures should help the students answer them.

5. There will be a wide variety of backgrounds among the students in a class, and an even wider variety from school to school. Hence the amount of time spent on a specific investigation may be larger or smaller, depending on the needs and backgrounds of the students. Some students have already used the task manager frequently, but others have not. Hence specific times for specific items cannot be suggested with any accuracy. They will probably even change from year to year in the same school with the same teacher.