School of Computer Science and Engineering

Laboratory Manual

NATIONAL EMERGENCY SITUATION MANAGEMENT SYSTEM (NESIMS) PROJECT

Three-Year crisis simulation exercise program to validate preparedness.

CZ3003 Software System Analysis and Design
Course coordinator: Kevin Anthony Jones
Version: 1.0
Effective date: 29 July 2017

Software Engineering Lab3, N4-B1c-14
School of Computer Science and Engineering
Nanyang Technological University

NATIONAL EMERGENCY SITUATION MANAGEMENT SYSTEM (NESIMS) PROJECT

PURPOSE

Purpose of document. This is the primary guideline or instruction set for conducting the lab project. From time-to-time, supplemental guidelines may be provided by the lab supervisor, lab execs, and course coordinator, either verbally or in writing, promulgated via email or announcement on the course site on NTULearn.

Purpose of lab project. This assesses the student's learning of the entire content of the course by practicing skills and knowledge pertaining to software system requirements, design, and implementation for a small, realistic case study, i.e., developing a system to respond to national emergency or crisis situation. This particular case study was used in previous serials of the course, and is quite understandable for the students.

LEARNING DESIGN

The lab project is designed as problem-based learning (PBL) where the learners are assigned into teams that work towards a common objective, and whose members earn the same assessment. The common objective is to solve a specific real-world problem that must be open-ended to engender several possible solutions. The conduct of a PBL session is centred on the classroom with the teacher present. PBL is conducted according to a prescribed process [see Figure 1].

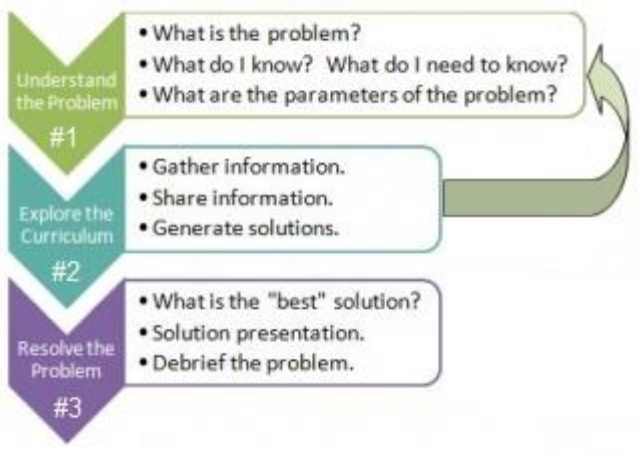

Figure 1. PBL three-stage model.

The process is simplified to three stages: understanding the problem, exploring possible solutions, and resolving the problem.

- First stage. In this stage, the team is assigned a real-world problem to solve, and determines what is needed to be learned in order to solve the problem. The team establishes the accordant tasks and deliverables, and assigns them to the members.
- Second stage. In this stage, the team members go their separate ways to effect self-directed learning associated with their tasks, and collect their specified deliverables. Once this has been completed, the team re-assembles, and one-by-one, each member passes on to the other members what he/she has learned.
- Third stage. In this stage, the team re-evaluates the problem, rejects now-perceived incongruous suppositions, and finalizes two or more suitable alternative solutions. The team discusses the merits of each solution, and subsequently reaches a consensus on one. Finally, the team presents their solution and explains the thinking of the group in reaching that solution to the teacher. If there are multiple teams in the session, each receives the other teams' presentations, and integrates the other teams' learning with their own.

The lab version of PBL has one problem that spans 11 weeks (the period of the lab portion of the semester), with six teacher-present sessions, each focusing on a segment of the one problem. The segments are selected for their representational importance and cleverness in driving the overall project effort forward to its eventual conclusion. Each teacher-present session is conducted as its own "mini-PBL". In other words, the project is one PBL of 11 weeks duration interspersed with six re-occurring mini-PBLs of two hours duration each [see Figure 2]; note that the sixth session is shorter at one and one-half hours. The model shows the 11 weeks of the main PBL on the horizontal axis, and the mini-PBL for a teacher-present session on the vertical axis. The mini-PBL is repeated six times corresponding to the six teacher-present sessions.

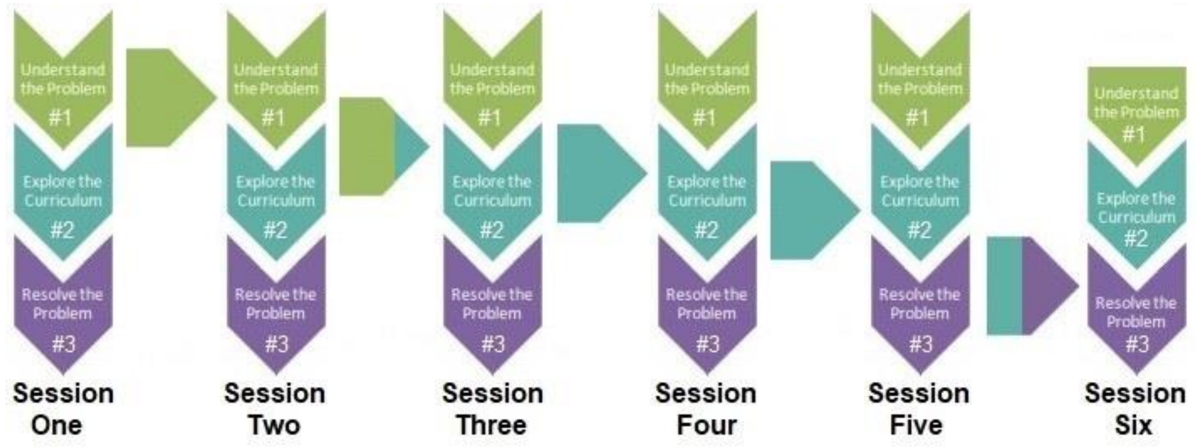

Figure 2. Model of project PBL with session mini-PBLs.

It is expected that stages one and two of each mini-PBL are completed students before the teacher-present lab session, and only the third stage is undertaken in the lab. By inference, each lab session should be considered as a resolution or reinforcement of the work in the previous weeks, and a launching pad for doing the work in the upcoming weeks.

STORYLINE

The Singapore Government is holding a tender for development of National Emergency Situation Management System (NESIMS). Several similarly organized software developers — commercial organizations (COs) whose purpose is to develop software — are proposing a software system solution supported by a live demonstration of a beta test version of the proposed software solution. The developer with the best proposal will be awarded the NESIMS contract.

FUNCTIONAL DESCRIPTION

NESIMS is the organization that deals— its responsibilities are preparedness, response, and recovery — with the humanitarian impact of a potential national-extinction event in Singapore. NESIMS comprises four subsystems: National call centre (911), Crisis management office (CMO), Emergency force (EF), and Prime Minister's Office (PMO). NESIMS is also the name of the software system that supports the online inter-operation of the four subsystems.

National call centre (911)

This subsystem receives and responds to any emergency experienced by people in Singapore. It is comprised of people, software, and hardware. Notifications of emergencies are by phone via a commercial telecommunications network. All calls are answered by specially-trained operators, who determine the natures of the emergencies, and notify the EF, who dispatches the appropriate help. Operators are connected via a secure commercial LAN [see Figure 3 for example].

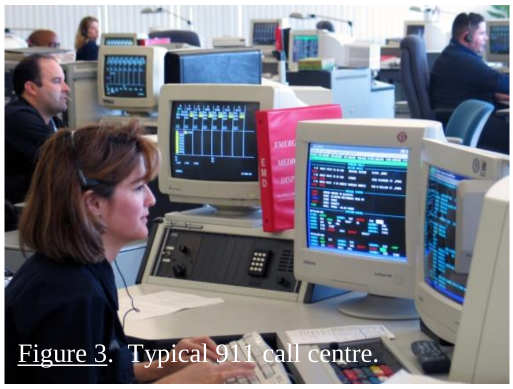
Figure 3. Typical 911 call centre.

If one or several emergencies are revealed to be potential crises, the CMO is notified and passed all data pertaining to the case(s). The CMO stations a liaison officer at 911 to assist them in determining the authenticity of all such suppositions of crisis.

Crisis management office (CMO)

This subsystem is a specialized version of 911. It analyses and responds to high-severity emergencies that threaten the entire nation. It is comprised of people, software, hardware, and digital networking [see Figure 4 for example]. However, the operators are exceptionally trained (more than 911 operators), and there are additional personnel in the CMO with military, scientific, high-level managerial, and political

Figure 4. Moscow Emercon

experience. Also, the CMO is commanded by a General rank serving officer in the SAF.

Notifications of suspected crises are received by the CMO online from 911. These are handled by the CMO operators, who correlate similar notifications to build a case, and then disseminate the information to the appropriate analysts and decision makers. These experts determine the extent of the threat to Singapore, the response options, and make a decision on option to be followed. Accordingly, the CMO orders the deployment of appropriate EFs, and tracks their progress in tackling the crisis. Once the crisis abates, the CMO deploys other EFs to clean up the major damage caused by the crisis, and assists the civilian agencies in returning the situation to normalcy.

Prime Minister's Office (PMO)

This subsystem is a ministerial level executive agency within the Government of Singapore that handles the ministries and other political matters that are of great importance to the nation, including national crises. It is comprised of people, software, and hardware; all its members with a commercial secure network. Notifications of a national crisis, and on-going updates concerning the progress of combating it, are directly from the CMO. In turn, the PMO authorizes the course of action that the CMO will take in eliminating the crisis [see Figure 5].

Figure 5. PM Lee online at office.

Emergency force (EF)

This subsystem commands assets for combating crises. Some assets exist solely for addressing certain types of crises, and others have normal responsibilities but are redirected to combat crises as and when required. It is comprised of people, software, hardware, and digital networking. Executive orders for deploying and operating in the crisis zone are received from the CMO. The EF provides real-time status reports on its progress in eliminating the crisis, and cleaning up the aftermath.

Figure 6. Emergency force coast guard ops.

The EF has exceptionally trained personnel and specialized equipment organized in units for combating specific types of crises. Also, the EF has a headquarters (HQ) unit that commands all deployed EF assets across a secure private digital network that it operates and controls. The main kinds of EF assets include military, bomb disposal, coast guard, HAZ-MAT, search and rescue, civilian evacuation, ambulance, emergency traffic control, firefighting, and infectious disease quarantine [see Figure 6 for example].

NESIMS network

The CMO operates and controls a secure private digital network that connects 911, PMO, and EF HQ. For the entire period of the crisis, the CMO maintains ________ communications with the _______ [see Figure 7].

- two-way ; PMO,
- two-way ; EF HQ, and
- one-way ; 911.

Essentially, the CMO is advised of a crisis by 911. The CMO and PMO enter into a dialog concerning the humanitarian aspects of the crisis and deciding on the course of action that will save the most people and built-up area of the nation. The CMO and EF enter into a dialog concerning how to fight the crisis and clean-up after.

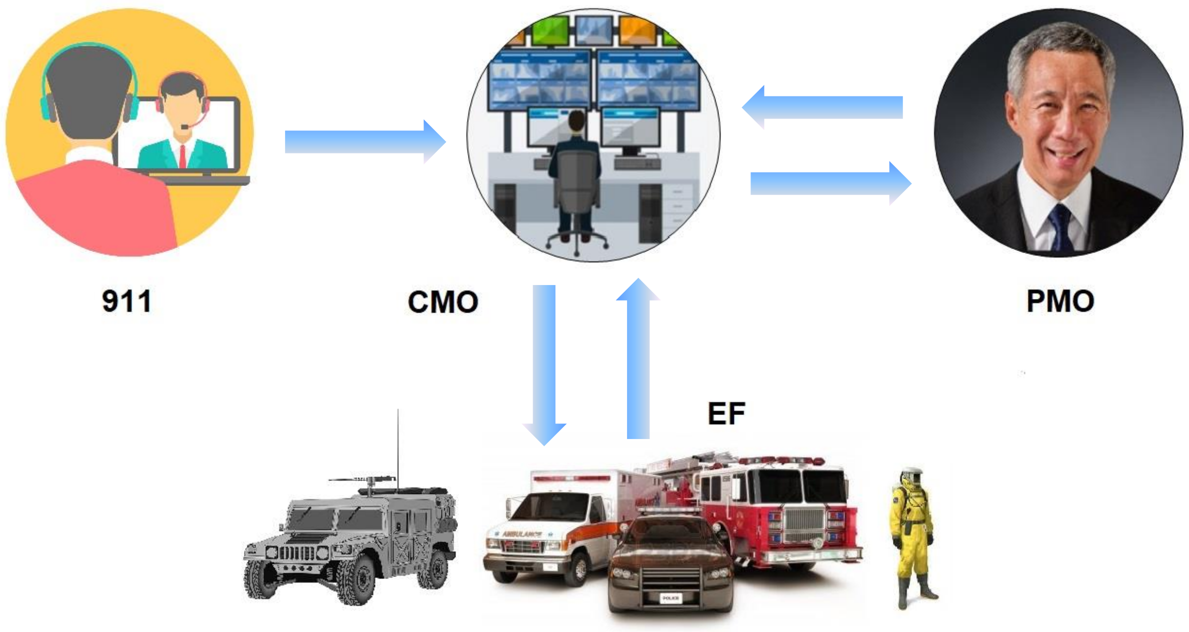

Figure 7. Artist illustration of CMO's digital network.

ORGANIZATION

Every student solely belongs to one software developer CO. Each such developer corresponds to at least one lab group registered for the course. Usually, one lab group is assigned to be one developer. However, in a few cases, several lab groups are combined to form one developer. In even fewer cases, an overly large lab group may be partitioned to form one developer and part of a second, that is subsequently combined with another part from another lab group.

The lab execs organize the lab groups into software developer COs, and assign a proper name to each of them. Table 1 is provided for the student to record the relation of lab groups to developers.

Table 1. Relation of student lab group to software developer CO.

	Lab group	Week day	Time	Software developer CO name
1				
2				
3				
4				
5				
6				
7				

The software developer CO does not have its own organizational management. In other words, the developer organization exists only as a container of sub-organizations. These sub-organizations are teams that develop specific kinds of software system. Typically, the developer is subdivided into four such teams, each tasked to develop one subsystem of the NESIMS; where a lab group is too large, a fifth team may be formed in the developer. Each team develops a unique subsystem; there is no case of two or more teams developing the same subsystem. The teams are essentially autonomous, with their own management. The four teams collaborate to produce the one integrated software system proposed for NESIMS.

In addition to organizing the software developer COs, the lab execs also organize the teams, and assign the NESIMS subsystem to each team. Teams' organizations follow a standard scheme [see below]. All proposed adjustments to this scheme must be approved by the Project architect.

- 911 – allotted four to six members;
- CMO – allotted six to nine members;
- EF – allotted four to five members; and
- PMO – also allotted four to five members.

In the event a fifth team is formed in the developer, supplemental instructions as to its development purpose in the NESIMS will be issued by the Project architect.

Roles

There are three kinds of roles assigned to this project, as listed below. All role holders are permitted to clarify the responsibilities of the role with the Project architect.

1. Student – Team lead, Requirements analysis lead, Design lead, Implementation lead, and SDLC developer;
2. Lab supervisor (school faculty) – Singapore government representative and Software developer CTO; and
3. Course coordinator (school faculty) – Project architect.

Team lead. This is an action role. The incumbent in the role is responsible for making the technical decisions about the subsystem that the team is assigned to undertake. This role also collaborates with the other team leads (in the developer) to undertake the system integration. Finally, this role is responsible for equalizing the workload for all members in the team, and to work with the three other leads in the team to ensure that all members get an opportunity to practice as much as feasible of the course's activities, skills, and models applicable to the team's assigned subsystem. [The team lead is appointed by the lab execs. However, the team lead is strongly encouraged to appoint an alternate to assist in maintaining continuity of work.]

Requirements analysis lead. This is an action role. The incumbent in the role is responsible for making the technical decisions concerning analyzing the requirements — as per the functional description section of this document — for the subsystem that the team is assigned to undertake. This role may petition the Singapore government representative role for alteration or substitution of the written requirements.

Design lead. This is an action role. The incumbent in the role is responsible for making the technical decisions concerning designing the subsystem that the team is assigned to undertake. This role may petition the Software developer CTO role for advice on suitability of aspects of the team's design proposal.

Implementation lead. This is an action role. The incumbent in the role is responsible for making the technical decisions concerning implementing the subsystem that the team is assigned to undertake. This role may petition the Software developer CTO role for advice on suitability of aspects of the team's implementation approach.

SDLC developer. This is an action role. The incumbent in the role is responsible to undertake any activities, skills application, and modeling pertinent to the course, applicable to the team's assigned subsystem, and assigned by at least one of the four leads in the team. This role supports the efforts of the four team leads but does not replace their efforts; in other words, the incumbent in this role is not to be exploited.

Singapore government representative. This is an advisory role. The incumbent in this role will clarify the expectations for already-stated requirements, and make decisions concerning any new or additional requirements.

Software developer CTO. This is an advisory role. The incumbent in this role will advise the students of any additional duties to be taken up by the team, or clarify expectations for already-assigned duties, in development of the team's assigned subsystem.

Project architect. This is an action role. The incumbent in this role is responsible for designing the entire project, and for making all decisions concerning the learning and teaching of the students undertaking the project for the semester.

The Team lead position is filled by a student of the lab execs' choosing. The other three team technical lead positions — Requirements analysis lead, Design lead, Implementation lead — are to be filled by student selected by the team. Additional duties concerning the logistics of the project include:

- Lab execs – provide lists and student photo sets for every team to the lab supervisors, load the entire scheme of developers+teams+team leads into the "Student teams" content area of NTULearn, and monitor attendance of all students in all software developer COs; and
- Supervisor – assess each student's performance during every lab session.

Further additional duties may be assigned by the Project architect as and when required.

WORK AND ASSESSMENT

This lab and its project affords every student the opportunity to practice every activity, develop every skill, and produce every model, explored in the discovery and team-based learning sessions of this course. This is realized through the work to develop the NESIMS subsystem assigned to each team. In other words, each subsystem should afford the students in that team with all the authenticity of purpose and content to do system and architectural thinking, modeling of architecture, imprinting quality, architecural styles, reuse, social technology, etc.

The course coordinator expects that every student understands this, and will make the most of the opportunity. However, in the case where a student feels that he/she is being blocked from this opportunity for any reason, they are encouraged to bring this to the attention of their lab supervisor and the course coordinator. A case in point is, if a member of a team is not completing the rightful work that is assigned to him/her by one of the four leads.

Course activities, skills, and models

<u>Activities</u>. These include any tasks in the informal "jumpstart" architectural process — i.e., derive components, craft initial architecture, transform with modeling language, and add quality and style — and preparing standard SDLC work products, such as specifications, reports, minutes, prototypes, and executable code.

<u>Skills</u>. These include developing using iterative approach, modeling with UML Component and Communication diagrams, architectural thinking, system thinking, soft skills such as communication, negotiation, and presentation, hard skills such as programming, system integration, and crafting software specifications.

<u>Models</u>. These include architecture structure and behaviour (iterations of initial in language-neutral notation, and final in UML Component and Communication diagrams' notations), artist illustration, context diagram, decision table, DFD, dialog map, ST diagram, and UML UseCase diagram.

Breakdown

The storyline provides the high-level outline of work required to be undertaken by the students for this project: to develop a NESIMS solution and conduct a live demonstration of a beta test version of the software solution. In order to achieve this, each team must develop a solution for its subsystem, and each software developer CO must integrate all four subsystem solutions together into one system solution.

Each team must establish its own proprietary standards for development methodology, document control, progress reviews, quality control, and decision support, for each subsystem. In their analysis and design, each team must consider and produce these artifacts and work products:

- High-level system specification, i.e., swproduct + hw + human interactors + ... ;
- Structure and behaviour architecture, in both initial and final forms;
- Operations: policy (including what set of circumstances becomes a crisis), chain of command, reports and other C^2 documentation, terms of reference, and operator tasks; and
- Enabling technology: framework, API, communications devices and emulators, modeling CASE tool, RDBMS, and web 2.0.

After completing the analyse and design of its subsystem, the team must implement a beta test version of software manifested from the design. Additionally, the team must generate sufficient dummy data so that the beta software subsystem can operate in a planned scenario for the live demonstration. [The programming language is established by the software developer CO. In other words, all teams utilize the same programming language for their subsystems' implementations.]

The complete beta software subsystems are joined together or integrated into one software system, the proposed NESIMS solution. This process is referred to as System integration (SI). The purveyors of the SI are the team lead roles, who share and devise adjustments to the technical aspects of their subsystems in order to achieve complete compatibility across the entire networking of the NESIMS. Such compatibility includes a system data dictionary, API-like connection points between the subsystems, and a system architecture. [There will be two "levels" of architecture: subsystem and system.]. It should be evident that adopting the same programming language in all subsystems would greatly decrease the complexity of the APIs for control and data exchange. [It is important that the Team leads share the SI information with the members of their own teams, so that they can also learn, and also assist in making the necessary adjustments to the subsystems.]

Figure 8 shows the deployment of the beta test version of software. The icons of "Call", "Chat", "Google Map", etc., highlight the major communications associated with the subsystems. The lab execs will provide the IP data for the PCs, firewall breaching, and any other assistance you need for establishing the connectivity of the NESIMS and its subsystems.

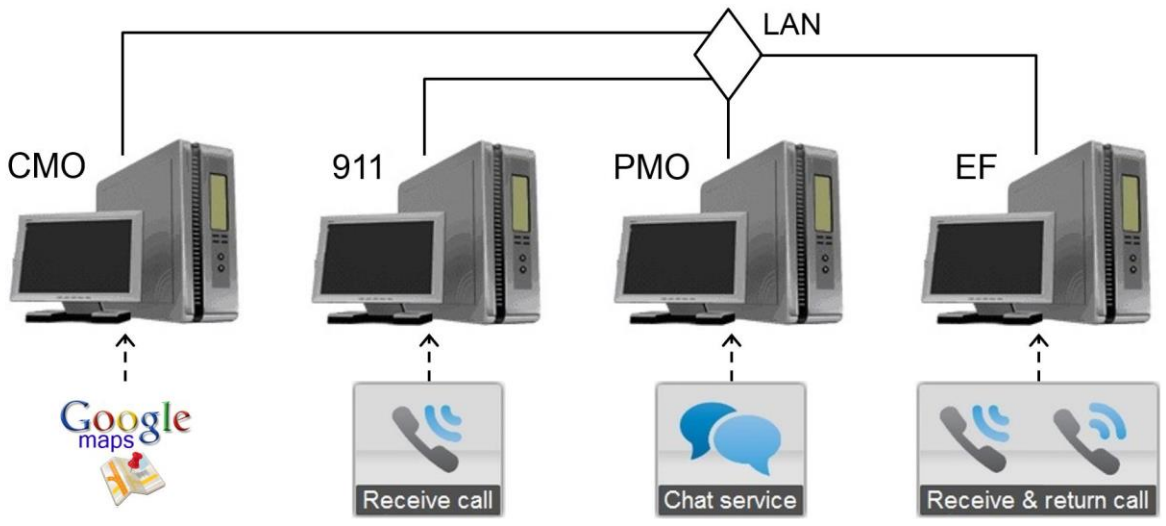

Figure 8. Model of the NESIMS live presentation (UML Deployment diagram).

Assessment

As stated in the course introduction, the lab assessment is the consolidation of four elements: 1) live demonstration, 2) student contribution, 3) assignment, and 4) attendance. In previous serials of the course, the students' lab assessments were generated from one major event (the live demonstration) and normalized to one mark for all the team members; differences in individual efforts and learning were not recognized. Now, the lab assessment scheme has shifted to approximately 40% individual effort and 60% team effort. This is intended to increase the fairness in recognizing the top-performing students, while retaining a degree of "propping up" the very weak students.

Attendance. This is an individual assessment. It is gleaned from the sign-in record of the student for the six teacher-present lab sessions. This sign-in sheet is made available at the start of the lab session, and closed after 30 minutes has passed. It is expected that the student remains in the lab after sign-in. If a student is found to have left the lab prior to the period's end, or being dismissed by the lab supervisor, the student will be marked as not attending.

Contribution. This is an individual assessment. It is degree and usefulness of each student's contribution to the team's work as observed by the lab supervisors. Naturally, the student has to be present to be marked in this assessment. The lab supervisor may approach a poorly-contributing student to advise them of his/her assessment, with the intention of affording the student a chance to improve.

Assignment

This is a team assessment. The assignment assesses the student's learning of content of the first six weeks; essentially it is a mid-term assessment. It can also be a mid-course sanity check for your team. Use it wisely! The assignment involves two main tasks: 1) posting the team's documents, and 2) giving a presentation.

Post. Each team is to provide at most one document for each of seven topics, as they apply to your assigned team's operations in the lab project as you understand them to be. The topics are: 1) system thinking, 2) architectural thinking, 3) software system composition, 4) initial architecture structure with functions, 5) initial architecture behaviour, 6) imprinted software quality, and 7) analysis modelling. The lab execs will advise the posting site. [It is a matter of honour that you respect every team's copyright on ideas and work products. Feel free to give constructive criticism to another team's work, but do not plagiarize it. Maximum penalty for acts of plagiarism is receiving a zero mark for the lab component.]

Presentation. Each team will give a short verbal presentation of the most noteworthy aspects of its work so far. It shall be under five mins. The intention of the presentation is to impress the other teams with how well your team has done in its project, but with superficial information; details are in your post. This will be conducted in the classroom for the weekly review. Marking of both tasks is by instructor and your peers. Marks by peers will be entered into LAMS rubric on the same day that the presentation is conducted. Teacher will advise details of using LAMS in week six.

Solution proposal

This is a team assessment. The solution proposal involves two main tasks: 1) posting the final set of the team's documents, and 2) giving the presentation of the developer's proposal for NESIMS.

Final post. Do not repeat the documents from the mid-term assignment. Instead, submit your final architecture (structure and behaviour) for your architecture and your best sales pitch models/video/etc. to explain/justify/highlight your team's contribution to the system. The lab execs will advise the posting sites for assessing and archiving documents and software. The Project architect will announce the deadline for submission.

Proposal presentation. Each developer gives a multimedia presentation of its NESIMS design and a live demonstration, in a contiguous sequence of its four subsystems separately, followed by the system. Total time for the developer's presentation is not more than 90 mins. In week 12, the developer submits to the Project architect and lab supervisor their teams' and SI individual presentation lengths and order for approval, who then allocates day and time slots for each developer's presentation in week 13.

SCHEDULING

For the semester

The schedule of teacher-present lab sessions is according to the published timetable; supplemented by assignment requirement and final presentation. The published schedule shows five teacher-present lab sessions; the sixth sessions is specially arranged by the Project architect and the lab execs. Some lab groups have sessions on odd numbered weeks, remainder on even weeks. Odd weeks sessions start on week 3, even weeks on week 4. Both groups (even and odd) have the same number of teacher-present sessions; the only discrepancy is that the even groups have one less week after their fifth session to do last-minute touch-ups on their final presentation in week 13.

Table 2 provides a summary schedule of the activities and deliverables for the lab project. Supplemental deliverables may be provided verbally by the lab supervisor. Additional supplemental guidelines may be promulgated by announcement on the course site on NTULearn.

Table 2. Detailed schedule of activities and deliverables.

Week	Date	Activity	Main deliverables
3 – 4	28 Aug – 8 Sep	Establish policy & analyse requirements	• operations policy, chain of command, reports and other C^2 docs, terms of reference, and operator tasks • artist illustration, context diagram, decision table, DFD, dialog map, ST diagram, and UML UseCase diagram
5 – 6	11 – 22 Sep	Design subsystems and SI	• high-level system specification • structure and behaviour architecture in initial form.
7	25-29 Sep	Submit assignment	Assignment
7 – 8	25 Sep – 13 Oct	Start coding subsystems	Code with framework
9 – 10	16 – 23 Oct	Conduct SI testing	• architecture • API
11 – 12	30 Oct – 10 Nov	Prepare live demo session	• sales pitch scripting and role-playing • handout • video
12	6 – 10 Nov	Request presentation slot	Presentation plan
13	13 – 17 Nov	Present solution proposal	• structure and behaviour architecture in final form • any other final vers of docs not submitted in assignment. • code of subsystems and integrated system

For the session

The teacher-present lab sessions are two hours in duration. Accordingly, the session is divided into half, with different activities scheduled for each half [see Figure 9]. In the first hour, the four teams in the developer assemble to 1) share their work products, especially models, and 2) highlight common issues to the lab supervisor in his/her roles of either Singapore government representative or Software developer CTO. This is the time for the teams to reset their bearings on the work ahead, and clear up any unknowns or uncertainties.

A new and innovative software for wireless media streaming will enable this sharing and highlighting. [It is important that all students appreciate that time is short, so don't be late, and come to with that session's document(s) ready!]

The second half is for the teams to continue with their development work. This includes revising their work products based upon the critique from the lab supervisor and their peers in the first hour, holding SI meeting and design review, and discussing new questions with the lab supervisor in or out of role.

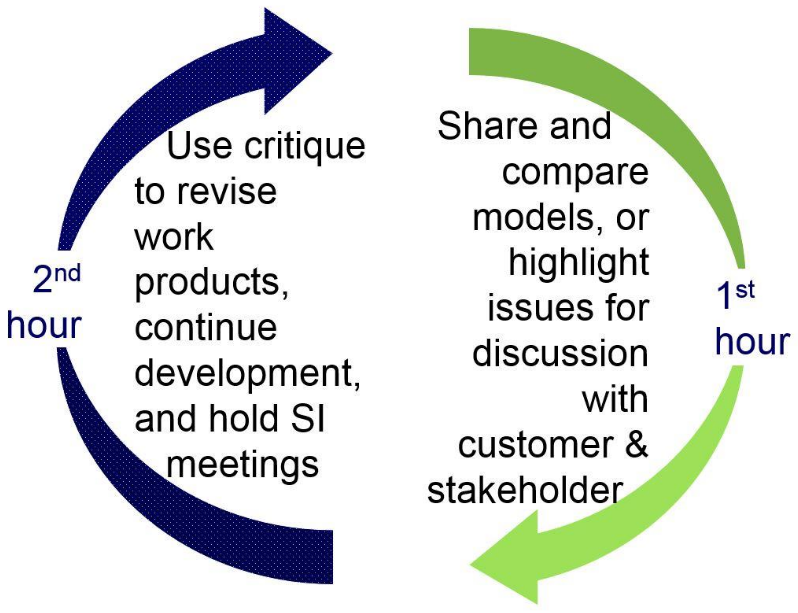
Use critique to revise work products, continue development, and hold SI meetings
Share and compare models, or highlight issues for discussion with customer & stakeholder
2nd hour
1st hour

Figure 9. Model of the teacher-present session scheme.

TECHNICAL

The lab execs are responsible to 1) provide the links for SVN and wiki, 2) ensure the availability and usability of all the PC standard software loaded in the lab's PCs, including office applications and visual Paradigm, and 3) provide on request all forms of general technical assistance to all members of the software developer CO. If a student is experiencing a technical problem with any equipment, hardware and software, in the lab, they may approach the lab execs for assistance.

The following is a short synopsis of instructions concerning the equipment in the lab. Supplemental instructions may be provided verbally by the lab execs and lab supervisor. Additional supplemental guidelines may be promulgated by announcement on the course site on NTULearn.

Personal computer. Students may use their own computer for crafting all work products, conducting all communications, and operating the final demo. In these cases, the student must ensure that all work is uploaded to a repository accessible by all members of the team.

Programming language. All developer orgs choose between C++, Java, and Python for software integration. In other words, four teams within a developer must employ the same software. However, each developer org may make their own separate choice of software.

Visual Paradigm. This is the modeling tool provided for student use in the software labs. Standard edition ver 14.1 of the software is the approved version. A licence key for this software is loaded on all desktop PCs in the lab.

Online sites for documents and code

There are two kinds of sites prescribed by lab execs for viewing — assignment and live demonstration — and archiving the team's and SI documents and code:

1. SVN – for code; and
2. Wiki – for documents.

The links for these sites will be provided by the lab execs to the Team leads during the first teacher-present session. Any change to this will be announced by the lab execs.

Solstice Pod

This is a wireless media streaming solution enabling multiple users to instantly connect, share and control content on a display device, perfect for enriching the learning process in a bring your own device classroom. It is an innovative technology for learning that has just been purchased; our course will be one of the first to adopt it into its instructional design. With it, students can: 1) share any type of image, video, web content, or application from their devices to the display, 2) easily organize, stack, enlarge, and move shared content on and off-screen, and 3) present in-depth multimedia presentations for group projects [see Figure 10].

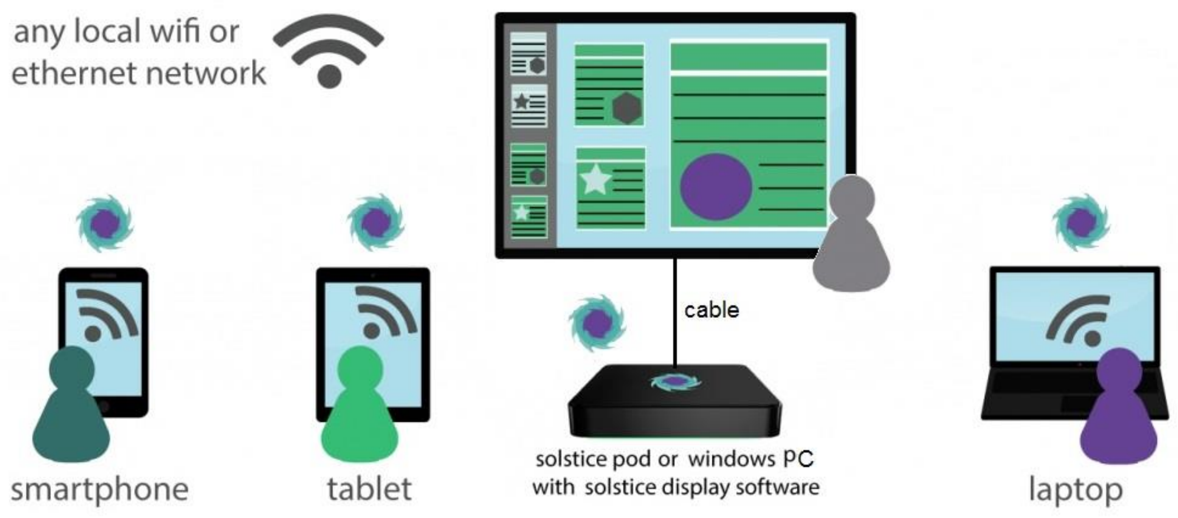
any local wifi or ethernet network
cable
smartphone
tablet
solstice pod or windows PC with solstice display software
laptop

Figure 10. Model of the Solstice Pod setup.

More information on the Solstice Pod is available in these videos:
https://youtu.be/HtyLFRkQgYs?t=47, and https://youtu.be/wn98BWn3C94

The teams for each developer will use Solstice Pod for comparing models or for high-lighting issues for discussion in each teacher-present session. During staged demonstrations of the device, it was evident that four different media streams could be projected on the lab's display screens, and all appeared to be usable and readable. So, the intention is to project the four teams' work products with the Solstice Pod. Each student PC has Solstice control centre to add and annotate. Most importantly, all teams' docs for that session must be added into the Solstice Pod at beginning of the session. [Be on time with the docs ready to load!]

HINTS

These are miscellaneous helpful suggestions, in no particular order, for undertaking this project. [The Project architect will add new hints to this list as appropriate.]

1. Importantly, appreciate two foundational tenets of this project. First, that the teams have full discretion in planning the subsystem features and functionality. Second, the most interesting and significant aspect of the project is the SI.
2. The software developer COs also have full discretion on the crisis scenario established for the live demonstration. The scenario can be either serious or fun; you will marked the same. In other words, crisis scenarios such as zombie attack and invasion by North Korea, will be marked the same as flood, firestorm, and terrorist. Just ensure that the EF subsystem is equipped with the units that can successfully combat the threat.
3. Further to the SI significance, appreciate that the subsystems are data and procedure generators. In other words they drive the content across the NESIMS, and they facilitate the SI by providing the API-like connection points. Therefore, teams must adopt a minimalist approach to the development of the features and functions of the subsystems. Do enough that a sufficient amount of content is generated and processed. Don't labour over features that are nice-to-have. And, use frameworks to be more productive!
4. If you are not handling a lot of data for the demonstration, then a RDBMS is not mandatory.
5. It is very acceptable for teams belonging to the same software developer CO to collaborate. Essentially, all four teams are actually on the "same team", so helping one weak team actually helps all four of the teams equally.
6. By the same token, ensure you maintain a level of confidentiality when it comes to after-hours contact with students from the other software developer COs. Be clear, your developer CO is in competition with the four others, and no-one can claim plagiarism for secrets given away freely!
7. A person can do many development tasks concurrently, but problems are best solved sequentially. Even if several persons partner to solve problems, let them do so in sequence.