Project Summary for Intelligent Nurse Cart V. 1.0

Jane LIU, Chi-Sheng SHIH, and iNuC Team
cshih@csie.ntu.edu.tw
Embedded Systems and Wireless Networking Lab
Graduate Institute of Networking and Multimedia
Department of Computer Science and Information Engineering
National Taiwan University

Copyright ©2010, National Taiwan University.

Report No. NTU/NEWS-10-0007
September 2010
Project Report

How to make

1) Please make all forms in English ONLY.
2) Please make the project report which is based on the standard list of contents given below.
3) There is no page limitation. But please remember that you should have all contents of the standard list in your project report Obligatorily.
4) If you prepare the Video Clip for demonstration, please send it as an attachment.
   (Video Clips are NOT mandatory.)

Standard list of contents

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Project Name, members and their affiliations</td>
<td>5. Development Environments (Tools and languages)</td>
</tr>
<tr>
<td>2. Abstract</td>
<td>6. Constraints and Measures to overcome</td>
</tr>
<tr>
<td>3. Problem Statement and/or the Value of the Project</td>
<td>7. Application areas (not stated in previous sections)</td>
</tr>
<tr>
<td>4. Architecture and Software Details</td>
<td>8. Project Schedules and Roles of Members</td>
</tr>
<tr>
<td>4.1 Overall System Architecture</td>
<td></td>
</tr>
<tr>
<td>4.2 Function of architectural blocks</td>
<td></td>
</tr>
<tr>
<td>4.3 Software Architecture</td>
<td></td>
</tr>
<tr>
<td>4.4 Details</td>
<td></td>
</tr>
</tbody>
</table>
1. Project name, members, and their affiliations:

Project Name: Intelligent Nurse Cart (iNuC) 1.0
Members and their affiliations:

- Jane W.S. Liu  Academia Sinica
- Daniel C.S. Shih  National Taiwan University
- Pei-Hsuan Tsai  Academia Sinica
- Yung-Chun Wang  Academia Sinica
- Yen-Ting Chuang  Academia Sinica
- Hung-Chie Lin  National Tsing Hua University
- Yu-Chi Huang  National Tsing Hua University
- Sana C.Y. Lin  National Tsing Hua University
- Kaze Y.C. Chang  National Tsing Hua University
- Kuau-Chun Chuang National Taiwan University
- Che-Wei Kuo  National Taiwan University
- Po-Ching Lin  National Taiwan University
- Wei-Cheng Wang  National Taiwan University
- Paul P.I. Hsu  National Taiwan University
- Ting-Shuo Chou  National Tsing Hua University
- Ren-Shan Luoh  Academia Sinica
- Chun-Ta Lu  National Taiwan University

2. Abstract

Intelligent Nursing Cart (iNuC) is an open source embedded software project on the design and prototyping of an intelligent, mobile medication cart for the purpose of preventing medication errors and enhancing patient safety. iNuC serves its user as a point-of-care medication administration and patient record keeping tool. It also provides the user with work and time management capabilities (e.g., calendar and to-do-list) and a web portal to hospital services and information system. A user with head nurse authorization can also use the cart to assign nurses to shifts and patients to nurses, define and modify policies and rules enforced by all carts used in her/his ward, and so on. In addition, iNuC has several labor-saving and automation capabilities, including generating shift report from data and notes collected during the user’s shift, tracking medication and medical supply usages. iNuC are also designed with configurability and customization in mind. Many of the features in the system are configurable to adapt the various workflows on different wards in the institute. Last but not the least, iNuC is highly integrated with the electronic controlled nurse cart to provide interlock mechanism and with hospital information system (HIS) to retrieve up-to-date patient records and to update patient vital sign information in real-time.

3. Problem Statement and/or the Value of the Project

Despite tremendous progress in quality of health care worldwide, recent publications (e.g., [1-7]) on the subject still report alarming statistics on occurrences and consequences of preventable medication errors. As examples, estimated numbers of preventable adverse drug events each year in US hospitals, in long term care facilities and among Medicare patients are approximately 400,000, 800,000 and 500,000, respectively. They are the cause of an estimated 25 % of admissions to nursing homes and have led to billions of dollars in hospital cost and deaths in thousands each year [5, 7]. This fact has motivated many projects on identifying and preventing medication errors (e.g., [8]) and the advent and use of an increasingly broader spectrum of information systems and smart devices as tools for prevention of medication errors and improvement in compliance (e.g., [9-28]).

Medication errors can occur at any stage of the medication use process, from ordering, transcription and dispensing to administration. Today, tools designed to support the early stages of the process are relatively more
mature. Computerized physician order entry (CPOE), electronic personal health record (PHR) and medication administration record (eMAR) systems (e.g., [9-14]) are widely deployed by world class hospitals, including National Taiwan University Hospital (NTUH). Pharmaceutical distributors and hospital equipment industry now offer all sorts of medication carts and smart cabinets (e.g., [21-24]) for dispensing medications in a controlled manner and tracking the use and inventory of medications managed by them. In particular, decentralized automated medication dispensing systems, such as MedSelect offered by AmerSourceBergen [21] and MedStation by Pyxis [22], can help reduce errors and improve efficiency in medication management, preparation and delivery. For this reason, NTUH Pharmacy Department is currently planning to automate their medication management and delivery process with the support of such equipment [29].

This project focuses on intelligent devices and tools designed specifically for prevention of administration error, i.e., errors occurring in the last stage of the medication use process. Administration errors, leading to non-compliance to error-free medication directions and preparation, contribute a significant percentage (25 – 40%) of all preventable errors. According to Professor Fe-lin Lin, Director of Graduate Institute of Clinical Pharmacy, College of Medicine, National Taiwan University, the rate of error in name, frequency, dose size, sequence and time of medications given to patients can be as high as one in six doses, even in prestigious, world-class hospitals [7]. As discussed in [6], such an error rate is not unexpected as medication administration process can overtax even the most well trained and disciplined medical care professionals.

While an automated and distributed medication management and dispensing process can also help to reduce administration errors, it does not go sufficiently far to the last mile. One can also make similar statements about currently available mobile point-of-care medication carts. Take the one described in [23] as example. It is among the best mobile carts available today. It does provide bar-code activated interlock for the purpose of ensuring that right medications are given to right patients and support automatic updates of patient medical records, billing records and pharmacy inventory records. Still, it does not support automation and integration to the degree required for error-free medication administration [15]. None of the available carts provides the end users (in this case nurses) with the modern planning, time management, reminder and alert tools that they can be easily customized to conform to the workflow processes of the institution and individual departments and personalized according to their own preferences. Like most medical devices today, the available carts typically have custom built, proprietary interfaces. Even the integration of devices with interface capabilities conforming to HL7 standards requires extensive custom development.

These observations motivated us to design and prototype intelligent medication carts without the above mentioned deficiencies. The project was built on our current work on tools and device for error-free medication process specifically and on component-based design, architecture and development of configurable automation and assistive devices in general. Publications on our work and links to software components can be found at project SISARL (Sensor Information Systems for Active Retirees and Assisted Living) homepage (URL: http://sisarl.org).

4. Architecture and Software Details

4.1 Overall System Architecture

The targeted users of iNuC are nursing staffs in NTUH specifically and in large hospitals in general. It serves them as a point-of-care medication administration and patient record keeping tool. iNuC also provides the users with work and time management capabilities (e.g., calendar and to-do-list) and a web portal to hospital services and information systems. A user with head nurse authorization can use the cart to assign nurses to shifts and patients to nurses, define and modify policies and rules enforced by all carts used in her/his ward, and so on. In addition, iNuC has several labor-saving and automation capabilities, including generating shift report from data and notes collected during the user’s shift, tracking medication and medical supply usages and automating requests for medication replenishment.

iNuC has the following capabilities and distinguishing characteristics:

- **Interlock**: iNuC provides interlocks and supports BCMA (barcode controlled medication administration), which is considered effective for ensuring that “the right medications are administered to the right patients”.
- **Electronic Record Keeping**: NuC supports update of electronic patient records and medical supply inventory records at the point of care as a part of the medication administration process.

- **Scheduler and Planner**: Modern time and schedule management, calendar and information tools provided by iNuC enable the user to make the medication preparation and administration schedules of her/his patients the central core of the user’s workday plan. This is an advantage over using separate personal portable devices and office facilities for this purpose for managing user’s workday schedule.

- **Intelligent Monitor, Alert and Notification**: The intelligent monitor, alert and notification (iMAN) tool has the potential to be one of the most effective tools for prevention of medication errors: The event monitoring and logging functions of iMAN aim to detect event and action sequences that have a high likelihood to cause error and alerting the user to take action for prevention of error.

- **Digital Assistance Tools**: iNuC also provides the user with many digital assistance capabilities. As an example, the user can enter notes about the patient in the process of caring each patient and have the notes merged with patient’s record and printed as shift report at the end of the shift. To a user with head nurse authorization, iNuC provides several management and supervision capabilities, including nurse-to-shift and patient-to-nurse assignments mentioned earlier. Such a user has access to information on all patients in user’s ward and can determine at a glance any work that should have been completed but remains incomplete. The user also can easily request iMAN to monitor specific events of interest and to send notifications in specified ways.

- **Configurability and Customizability**: Flexibility is a primary design objective of iNuC. Different wards in a hospital may follow different medication administration protocols and policies. iNuC provides easy-to-use configurations tools with which users with proper authorization can change configuration parameters that define the rules enforced by carts used in the ward. Examples of such parameters include the lengths of time by which scheduled medication times can be advanced or delayed and records entered via the cart on-line are allowed to be edited on-line, etc. Similarly, they can define for iMAN combinations of events to monitor, conditions to trigger alerts and ways notifications are delivered.

The emphases of the project are on component-based approach to make the device and its user interface configurable and customizable and on flexible architectural framework for their integration with diverse supporting devices and evolving infrastructure. Indeed, configurability and customizability is a primary design objective of the proposed mobile point-of-care cart. Being a user-centric device, the cart should be easily configurable to support manual medication delivery workflow processes illustrated by Figure 1(a). They are still in use in numerous hospitals in Taiwan and most countries in Asia. The cart should also be easily configurable to support the new medication preparation and delivery process which NTUH is planning to adopt in the near future [29]. It exemplifies automated processes illustrated by Figure 1(b) that are becoming more and more widely adopted each day. In addition, the cart can be customizable by users for not only general workflow process of the institution, but also different processes of individual departments and user groups within an institution.
(a) Manual process using supply carts with point-of-care (PoC) carts

(b) Automated process with distributed medication management

Figure 1 Medication Delivery Workflow
**iNuC Use Scenario**

iNuC system is designed to collaborate with other systems in the hospital so as to reduce medication errors and smooth out nurses' workflow. Figure 2 depicts how the nurses use iNuC to complete their daily work. On the left, the figure shows that the physician writes the prescription using CPOE. The prescriptions are sent to pharmacy to dispense the medication. On every day, the prescribed medications are delivered by medication supply carts to the wards. When a nurse is on service, he/she logs in iNuC to start his/her shift. As discussed above, the nurse will be asked to check the availability of the medication. When it is the time for patient to take medication, iNuC will alert the nurse in advance to be ready. When the nurse is in the patient room, the nurse follows the instructions on screen to check patient identity and double check that the medication is correct according to prescription, shape of medication retrieved from medication databases. When necessary, the nurse also records all the vital sign information on iNuC and keeps track all of the medication supplies used for treatment on the visit. When the process completes, the nurse locks the container for the patient and continues his/her service on other patients.

At end of the shift, on-service nurse need to hand over the assigned patients to the nurse in the next shift. The process is completed by printed handover sheet and software interlock mechanism to assure that every patient is served by one nurse at all time. When a nurse logs off the system, the data are then all uploaded to database servers. We will discuss the database servers in Section 4.2.

### 4.2 Function of architectural blocks

As discussed above, iNuC assists nursing staffs to take care of the patients. To accomplish the goal, iNuC collaborates and interacts with other devices and systems in the hospital. In the following, we present how iNuC collaborates with other devices, shown in Figure 1(b).

- **Medication Supply Cart:** in the distributed medication management model, the pharmacy supplies medication for all the wards on daily bases. Medication supply carts deliver the medication for the following day on weekdays or three days on weekends to the nurse station. The supply cart is then left in the nurse station until another supply cart arrives on the scheduled deliver date.

When a nurse is on service, he/she logs in to iNuC and will be asked to prepare the medication first. While preparing medication, the nurse checks if the medications are ready for his/her shift. There are two alternatives to prepare the medication. The first one is to take over the containers owned by the nurses in the preceding shift. During shift handover, the nurse in preceding shift will not only hand over the duty to take care of the patients but also hand over the medication containers. On each container, there is a RFID tag with unique ID to associate a container with the patient. During the shift handover, the R DID tag on each container assures that the container is associated with the right patient when it is inserted into a different nurse cart. When there is not enough medication in the container, the nurse starts medication supply process on iNuC to move the medication on supply cart to iNuC cart. This step is critical to the medication safety. If the medications are stored in a wrong container, it is likely that the nurse will give the wrong medication to the patient. (The nurse is supposed to check if the medication is consistent with the prescription.) To avoid such a mistake, RFID on each container is used to identify the patients.
Hospital Information Systems: There are two major concerns for interacting with HIS: one is connectivity and the other one is portability. To assure the mobility of the nurse carts, iNuC uses wireless network to connect to HIS to access Electronic medication record (eMAR) shown on the left side of Figure 1(b). Although wireless network is a mature technology, there is no guarantee that iNuC is always connected to HIS. Hence, iNuC adopt different strategies for information download and information upload. Information download is conducted when the nurse logs in. All the required information in the login session are downloaded before the system starts to serve. However, information will be uploaded at background in periodic manner. It is evident that the data on HIS may be inconsistent with that on iNuC in general. However, several assumptions assure that we can ignore the data consistency issue. For instance, one patient is only taken care by one and only nurse. Hence, at most one iNuC will own the medication record for each patient at any time.

The block diagram of iNuC in Figure 3 depicts the software structure of iNuC version 1.0. It contains the seven major components: User interface (UI), Work and Time Management (WTM), Lockers and Interlock Mechanism (LIM), Date Refreshing Mechanism(DRM), Record Keeper (RK), Authentication and Authorization (AaA), Intelligent Monitor, Alert and Notification (iMAN), and Cart and User Event Logging (CUEL). We discuss the function for each of the components in the next subsection. Among these components, UI is the one to interact with the users. When design UI, system responsiveness is our first priority. Hence, a dedicated UI thread is assigned to this component. In addition, the interaction between UI and other components in iNuC are all asynchronous so as to avoid blocked user interface. LIM is the component to control the mechanical locks in the nurse cart. It only accepts lock/unlock commands from UI and also detects if a container is locked and senses the ID of the container. DRM component is the one to retrieve data from HIS and other network servers. Because all the data accesses are served by DRM, it becomes trivial to connect iNuC to a new data server. At this moment, web service is the default interface for DRM to interact with data servers.

Figure 3 shows how iNuC’s are integrated with hospital record and information systems. In general, the iNuC server shown in the figure isolates the prototype tools from hospital information and record systems. Specifically, it serves two purposes: First, it holds data and information that are not provided by NTUH server. Examples are calendars of patients and users and iNuC configuration settings for different departments and users. Management information includes nurse assignments to shifts and patient assignments to nurses. This type of information is essential for iNuC access and authorization control operations but is not yet available from NTUH server. Second, iNuC server provides individual iNuC carts access to real-life patient records, medication information, personal information, etc. via the set of iNuC data exchange API functions. iNuC local patient records in the figure refer to records that have updated by iNuC carts and are yet to be sent back to NTUH server.
iNuC is designed to operate as a self-contained unit. All of its application tools run on the same embedded computer that controls the drawers in the cart. The cart is connected to the iNuC and hospital servers via a wireless network. It interacts with the servers only for the purpose of exchange data: During a user login process, the cart relies on the user id/password information on the server for authentication purpose. After the user is authenticated and login as the primary user succeeded, user and patient data are downloaded from the hospital servers to the cart.

In the process of administering medications and updating the patient records, the updated records are temporarily stored locally. Updates from the cart to the server are carried out in the background by the RK/DRM modules within a specified delay interval when the server and network are operational. During network and server outage, the cart can reliably maintain all records and synch with the server(s) when the server(s) becomes available again.

### 4.3 Software Architecture

The block diagram of iNuC in Figure 3 depicts the software architecture of iNuC version 1.0. It contains the following major components:

- **User Interface (UI)** is clearly one of the most important components of iNuC. Most of the tasks done by the cart on user behalf are dispatched from there.

- **Work and Time Manager (WTM)** is responsible for generating and maintaining schedules of the patients cared by the user, as well as the personal schedule of the user. Its responsibilities also include sending reminders prior to scheduled events, checking schedule changes for correctness (i.e., conforming to rules) and sending alerts and logging the occurrences when scheduled tasks are overdue.

- **Lockers and Interlock Mechanism (LIM)** provides easy-to-use functions for other modules (e.g., the UI) to manage medication drawers. Functions provided by LIM include those for querying status of medication drawers and unlocking medication drawers specified by drawer ids or locations. In this way, LIM module hides the control of electronic and mechanical devices used to manage medication drawers.

- **Data Refreshing Mechanism (DRM)** module is responsible for data transmissions between the cart and the server. It guarantees reliable data transmission and specified quality of service when network is available. When the network is down, it is responsible for the completion of all data transmissions requested during network outage as well as yet-to-be-completed transmissions requested before network outage.

- **Record Keeper (RK)** provides the record semantics on top of the DRM module. It provides functions that
the other modules call to upload and download patient records, schedules, prescriptions, and other types of files to and from the server.

- **Authentication and Authorization (AaA)** module authenticates the user during login. It checks the user for authorization to control user’s access to patient data and other data generated and maintained by iNuC.

- **Intelligent Monitor, Alert and Notification (iMAN)** is a run-time monitoring tool. It provides the capabilities for real-time monitoring, capture and analysis of events and conditions that indicate the potential for occurrences, or actual occurrences, of errors, and issuing alerts and notifications to trigger error handling or prevention actions. It also provides notification functions other modules can call to send reminders and alerts.

- **Cart and User Event Logging (CUEL)** system provides custom event logging services to all iNuC modules, enabling each module to have a separate custom event log.

As shown in Figure 4, iNuC is integrated with HIS server with the assistance of iNuC Server. Figure 5 shows how iNuC server assists to assure the portability and interoperability of iNuC. HIS server provides its data via web services, which are widely accepted on online service systems. It will help us to assure that iNuC is not tighten with any database systems. iNuC server can be viewed as the data proxy for iNuC. When the required data is located on HIS server, it sends the request by web service interface to retrieve/update data. When the required data is not available on HIS server, iNuC server accesses its local database. Local database can also cached the data received from HIS database systems. At this moment, Microsoft SQL Server is used in iNuC server to maintain its local data and MySQL can be used with least efforts.

![Figure 5 Data flow between iNuC and HIS DBs](image-url)
Figure 6 illustrates how the components in iNuC and iNuC server exchange their data. As discussed earlier, DRM is the only data access interface between iNuC and iNuC server. It assures the consistency to maintain data. Most of the components in iNuC except UI are implemented in C to assure its portability and UI module is implemented in C# to take advantage of the graphical user interface capability provided by Microsoft Windows environment. As a result, the data should be formatted in C and C# language. On iNuC server, there is also a DRM module, which servers the interface between iNuC server and iNuC.
4.4 Details

Design and implementation of iNuC is a collaborative effort. There are more than part-time graduate and undergraduate students in the team. To assure the quality of the software system, the team starts from requirement capture and design document, rather than coding. The code is not started until the user interface, data exchange API, coding style, and software architecture are all pinned down. In the following, we present how we capture the user requirements and attach data exchange API and coding style document in Appendix A and B.

The project has adopted two practices in the process of capturing and documenting iNuC requirements. We want to ensure the effectiveness and usability of the prototype for NTUH and similar institutions.

Close End User Involvement The first is to involve the end-users (i.e., the actual users) from the start of the process. This widely advocated practice was motivated by the fact that insufficient attention to eliminate misunderstanding in requirements between the users and developers can lead to inconsistencies between views of the user and the system and such inconsistencies are sources of errors known to occur when new automated systems are deployed. As pointed out by studies on this problem, including [2], this and other causes of errors can be avoided to a great extent by involving end users throughout the requirement definition, design and development process.

Following this practice, we started the requirement capture process in May by presenting the iNuC concept and functions in a meeting organized by Head Nurse Yew. The meeting was attended by over 40 NTUH nursing staffs from many various departments. They provided us with guidelines to follow and deeper insight into users’ wishes and views. Shortly after that meeting, the nurse at Ward 4A and Ward 11B were identified as the first amongst trial users and evaluators of iNuC prototype. The CSIE team met with them regularly until concrete use scenarios were developed and requirements were prioritized.

Rapid Prototype as Requirement Capture Tool The second is to use a mockup, sometimes called rapid prototype, as a requirement solicitation, evaluation and documentation tool. Requirements of intelligent tools such as iNuC are typically specified in textual and diagramatic forms, and sometimes, argumented by more precise and formal representations for the benefit of the designers and developers. Such requirement specifications are ineffective as a communication channel between the end-users and developers. In the case of iNuC, a mockup is a virtual iNuC. Except for the absence of the physical medication drawers with interlock mechanism and fake patient data and nurse ids, the mockup gives the look and feel of a real cart. We use it as a tool for acquisition of accurate information on what the user wants the cart to do and what not to do, how the users want the cart to interact with them, how they want the display to look, what their preferences in input/output devices and modes are, etc. More importantly, we hope that through the process of this evaluation involving the actual users using a mockup, we can identify and eliminate most if not all the inconsistencies between views of the user and the system.

The mockup proved to be effective as we had hoped and expected it to be. Through it, we indeed obtained valuable feedback and suggestions. They are summarized in Section 3.1. It also enabled us to identify design defects and potential bugs that are likely to remain unnoticed until the prototype is built and put to use.

Figure 7 gives an example to illustrate this point. The screen dump in part (a) is displayed by the mockup when the user administrates medications to a patient. A serious problem with this screen is the label “Does Amount” (in Chinese) of the fourth column of the table. The term is a bug left unintentionally in the mockup. A design defect is the placement of the column, being just to the left of column that list prescribed doze sizes. As a consequence, evaluators of the mockup invariably thought that the tool allows the user to change the dose sizes.

---

1 A well-known case is the computerized physician order entry system developed and used by University of Pennsylvania Hospital. Over 20 new kinds of error occurred when the automated CPOE system was put to use at University of Pennsylvania Hospital.
of the medications and this column requests her/him to enter the actual dozes given to patient and the reason(s)
for the changes. Changing dose size is of course not allowed for most medications.

The Powerpoint drawing in Part (b) of Figure 7 shows what the developers had in mind. The column should
be labeled “Dose Size in Treatment” and is now placed to the far left of the table: If for some reason (e.g., a
tablet fell on the floor and hence could not be given to the patient) the actual quantity of some medication
consumed at the time differs from the prescribed (and patient consumed) quantity, the user is asked to enter the
actual quantity and the reason for the discrepancy. This information allows the tool to track the supplies more
accurately. It can also be used for inventory control purpose. The revised version still allows the user to record
the fact and provides explanation in the event the patient is not able to take the prescribed dose. Such an event is
logged by the cart, and if so configured, the cart can send notification to alert designated individual(s).
During the design phase, coding style and data exchange APIs are both critical for a success software project. Appendix A and B show these two documents. Although parts of the document are written in Chinese, we are sure that it still shows our efforts to define the APIs before the team starts to write the code.

Appendix C is the light version of iNuC user manual and demonstrates the completeness of iNuC. (The full version can be found at http://www.sisarl.org/index2.php?option=com_docman&task=doc_view&gid=243&Itemid=33) Because the targeted users are the nursing staffs in Taiwan, the document is written in Chinese. The manual is composed by capturing the screenshots to demonstrate how to use the system. Hence, the document can demonstrate the completeness of the system.

5. Development Environments

All the components listed in this report, except the user interface, are written in C. The user interface is written in C#, using Windows Forms. The detailed development is listed below.

- Microsoft SQL Server 2005
- OpenFoundry Platform (http://www.openfoundry.org)
- Internet Information Services (IIS)

6. Constraints and Measures to overcome

The development of iNuC is under beta test. During the design and development phase, we assure that end-users’ feedback is our first priority so as to complete a user-centric systems. When iNuC is under design, user interface mockup allows to collect user feedback in early design stage and helps to avoid requirement adjustment in the later stage. It also helps the team to develop a well-accepted embedded software. When the
system is in alpha test, we received lots of positive feedbacks from our end-users. In particular, the usability and reliability are two valuable feedbacks. However, similar to most of software development, system design specification and requirement change over time. iNuC team also faces the same issue. Hence, from the very beginning, we decided to document every thing before writing the first line of code. The approach reduces the change of design specification and requirement during the development.

7. Application areas

iNuC is designed for nursing staffing in hospital and long term care health institute. In particular, we aim on the nursing staffs in Asia. According to our study, the workflow and workload for nursing staffs in Asia are dramatically different from those in North America and Europe. Specifically, in Asia, one nurse may take care 6 to 14 patients in normal ward and 2 patients in ECU. However, in North America and Europe, one nurse usually takes care of 2 to 6 patients. How to smooth out the workflow for nursing staffs in Asia is much more critical. In the mean time, we also see the trend of nursing practice in Asia for moving forward western style: reducing the number of patients for each nurse. Hence, the design of iNuC also take this into account and allows iNuC to be configurable for different workflows and workloads.

8. Project Schedules and Roles of Members

The project starts in 2008 and is scheduled to release in November 2010. iNuC is now under beta test to assure the integrity with iNuC cart and HIS system. The role of team members are listed below.

- Jane W.S. Liu System Architect
- Daniel C.S. Shih System Architect
- Pei-Hsuan Tsai System Architect
- Yung-Chun Wang Project management and Software development of WTM
- Yen-Ting Chuang Software development of UI and iNuC Core
- Hung-Che Lin Software development of LIM
- Yu-Chi Huang Software development of UI and iNuC Core
- Sana C.Y. Lin Software development of IMAN
- Kaze Y.C. Chang Software development of LIM
- Kuah-Chun Chuang Software development of DRM and RK
- Che-Wei Kuo Software development of AAA
- Po-Ching Lin Software development of RG and user manual
- Wei-Cheng Wang Software development of web configuration tool
- Paul P.I. Hsu Software development of RG
- Ting-Shuo Chou Software development of LIM and hardware on iNuC cart
- Ren-Shan Luoh Software development of RK and iNuC Server
- Chun-Ta Lu Software development of CUEL

Reference:


Baron, R. J., E. L. Fabens, M. Schiffman and E. Wolf, “Electronic health records: just around the corner? Or over the cliff?,” Annuals of Internal Medicine, 2005.


MEDICATE (Medical Diagnosis, Communication and Analysis throughout Europe) project, http://www.chime.ucl.ac.uk/work-areas/ehrs/Medicate/index.htm.


[32] P. H. Tsai, H. C. Yeh, P. C. Hsiu, C. S. Shih, T. W. Kuo, J. W. S. Liu, “A scarce resource model for...
# Appendix A

## FlexMMS and iNuC Coding Style V1.2

*November 2009*

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>2</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>2</td>
</tr>
<tr>
<td>2 File Organization</td>
<td>2</td>
</tr>
<tr>
<td>2.1 Source files</td>
<td>2</td>
</tr>
<tr>
<td>2.2 Directory/Folder</td>
<td>2</td>
</tr>
<tr>
<td>3 General Appearance</td>
<td>2</td>
</tr>
<tr>
<td>3.1 Line Length and Wrapping</td>
<td>2</td>
</tr>
<tr>
<td>3.2 Indentation</td>
<td>3</td>
</tr>
<tr>
<td>3.3 Blank Lines and Inter-Term Spacing</td>
<td>3</td>
</tr>
<tr>
<td>3.4 Brace Placements</td>
<td>3</td>
</tr>
<tr>
<td>4 Comments and Documentation</td>
<td>5</td>
</tr>
<tr>
<td>4.1 Single Line Comments</td>
<td>5</td>
</tr>
<tr>
<td>4.2 Block Comments</td>
<td>5</td>
</tr>
<tr>
<td>4.3 Description of Files</td>
<td>5</td>
</tr>
<tr>
<td>4.4 Descriptions of Functions</td>
<td>6</td>
</tr>
<tr>
<td>5 Naming Conventions</td>
<td>7</td>
</tr>
<tr>
<td>5.1 Capitalization Style</td>
<td>7</td>
</tr>
<tr>
<td>5.2 General Rules</td>
<td>7</td>
</tr>
<tr>
<td>5.3 Naming Guidelines</td>
<td>7</td>
</tr>
<tr>
<td>6 Declarations and Initialization</td>
<td>12</td>
</tr>
<tr>
<td>6.1 Number of Declarations per Line</td>
<td>12</td>
</tr>
<tr>
<td>6.2 Initialization</td>
<td>12</td>
</tr>
<tr>
<td>7 General Suggestions</td>
<td>12</td>
</tr>
<tr>
<td>7.1 Code Quality</td>
<td>12</td>
</tr>
<tr>
<td>7.2 Defensive Programming</td>
<td>13</td>
</tr>
<tr>
<td>7.3 Source Code Review</td>
<td>13</td>
</tr>
<tr>
<td>7.4 Debugging and Testing</td>
<td>13</td>
</tr>
<tr>
<td>8 Specific Suggestions</td>
<td>15</td>
</tr>
<tr>
<td>8.1 Use Enum and Symbolic Constants</td>
<td>15</td>
</tr>
<tr>
<td>8.2 Use Secure String Copy Functions</td>
<td>15</td>
</tr>
<tr>
<td>8.3 Test Function Return Values</td>
<td>15</td>
</tr>
<tr>
<td>8.4 Use Goto Sometimes</td>
<td>16</td>
</tr>
<tr>
<td>8.5 Terminate Threads Cleanly</td>
<td>16</td>
</tr>
<tr>
<td>References</td>
<td>17</td>
</tr>
</tbody>
</table>
Preface

This version of “FlexMMS and iNuC Coding Style” contains a few clarifications of the earlier version posted in October this year. Both versions are updates of “NuC Coding Style and Rules-v2” posted at http://sisarl.org in December 2008. In addition to corrections, the updates consist of a few naming guidelines for workflow-based code: They are in Section 5.

Please send suggestions on additional rules and guidelines and corrections and questions of the contents here to janeliu@iis.sinica.edu.tw.

1 Introduction

The first part of this note describes coding style and conventions adopted for iNuC (Intelligent Nursing Cart) and FlexMMS (Flexible Medication Management Systems). The underlying principle, style, conventions and many examples are from

“C# Coding Style Guide,” http://debian.fmi.uni-sofia.bg/~iliyan/csharp_coding/

Like the previous versions, this version still puts more emphasis on C programming and contains only minimal rules (e.g., names for methods and classes) specific to C#. Please read the above mentioned document if you have any questions regarding them. That document contains many more illustrative examples and discusses underlying rationales. So, you may want to take a look at it even if you have no questions.

The second part, consisting of the last two sections (i.e., Sections 7 and 8), contains suggestions for writing correct, reliable and clean programs. Please read the subsections that are applicable to your code and follow the suggestions as much as possible.

2 File Organization

2.1 Source files

Keep your functions and files short. A good rule of thumb is to keep file length under 2000 lines. You usually can do this by dividing and structuring the code of your component into parts. Each part, referred to as module hereafter, fits in a file.

2.2 Directory/Folder

Put source files for each component in a folder named by the name of the component. For example, the source files for the intelligent monitor, alert and notification (iMAN) component are in a folder named iMAN.

3 General Appearance

3.1 Line Length and Wrapping

Keep lines, code and comments, no more than 80 characters in length. Break up
expressions that do not fit in one line according to the following rules:

- Break after a comma,
- Break after an operator,
- Use high-level breaks, rather than low-level ones,
- Align the new line with the beginning of the expression at the same level on the previous line.

Example 1: Breaking up a function call:

```c
LongFunctionCall(expression1, expression2,
                   expression3, expression4,
                   expression5);
```

Example 2: Breaking up an expression:

```c
variable = a * b / (c - g + f) + 4 * z;
```

3.2 Indentation

Use spaces for indentation, not tabs. **Indent 4 spaces per level.**

3.3 Blank Lines and Inter-Term Spacing

- Use blank lines to set off each segment of codes that are logically related. The example on code in general listed in the next section illustrates this point.
- Put one or two blank lines between local variables in a function and its first statement, between function definitions and so on.
- There should be a single space after a comma or a semicolon. As an example, write

```c
TestFunction(a, b, c);
```

```c
  not  TestFunction(a,b,c); or TestFunction(a, b, c) ;.
```
- Similarly, surround operators (except unary operators) by spaces. As examples, write

```c
a = b;  not a=b;
for (i = 0, i < 10, ++i); not for (i=0,i<10,++i);
```

3.4 Brace Placements

- After function declaration – The opening brace “{” appears at the start the next line after the declaration statement. The closing brace “}” starts a new line indented to match the corresponding opening brace. Below is an example.

```c
BOOLEAN
MyInitSystem (
    VOID
) {
  // Implementation code here.
}
```

- For coding C# class – Follow the example below from C# Coding Style Guide. Note
that there is **no** space between a method name and the open parenthesis.

```csharp
Class MySample : MyClass, IMyInterface
{
    int myInt;

    public MySample(int myInt)
    {
        this.myInt = myInt ;
    }

    void Inc()
    {
        ++myInt;
    }
}
```

Within code in general – The following example, also from C# Coding Style Guide, illustrates where opening and closing braces should be placed, as well as indentation, inter-term spacing and blank lines. Note that in this example, the opening brace “{“ is placed at different places even within a loop. Both placements are acceptable.

```csharp
for (int primeCandidate = 1; primeCandidate < number; ++primeCandidate)
{
    isPrime[primeCandidate] = TRUE;
}

for (int factor = 2; factor < number / 2; ++factor) {
    int factorableNumber = factor + factor;

    while (factorableNumber <= number) {
        isPrime[factorableNumber] = FALSE;
        factorableNumber += factor;
    }
}

for (int primeCandidate = 0; primeCandidate < num; ++primeCandidate)
{
    if (isPrime[primeCandidate] != FALSE) {
        printf("primeCandidate + " is prime.\n")
    } else {
        printf("primeCandidate + " is not prime.\n")
    }
}

switch (condition) {
    case MostFrequentCase:
        ...
        break;
    case SecondMostFrequentCase:
        ...
        break;
    ...
    case LeastFrequentCase:
        ...
        break;
    default:
    break;
```
For try-catch statements, use the styles illustrated by examples below.

```java
try {
    ...
} catch (Exception) {}

try {
    ...
} catch (Exception e) {
    ...
}

try {
    ...
} catch (Exception e) {
    ...
} finally {
    ...
}
```

4 Comments and Documentation

4.1 Single Line Comments

Use // comment style to add comments to your code. A single line comment should be indented to the indent level of the commented code.

You can use // to comment out lines of code. In that case, place // at the beginning of the line for readability.

4.2 Block Comments

Avoid block comments in your code as much as possible. When you must use it, use the style

```java
/*
Line 1
Line 2
Line 3
*/
```

4.3 Description of Files

Use block comments at the start of each file to present statements on copyright and license, content of the file, environment, author names and revision history, etc. Below is an example.

```java
/*
License and copyright Statement: to be added
Module Name:
```
iMANapi.c

Abstract:

This file contains API functions provided by the intelligent monitor, alert and notification component, including functions to query for alert conditions and to request notification be sent upon occurrences of specified conditions.

Authors:

Your name (your email address) 15-Oct-2009

Environment:

User mode only.

*/

4.4 Descriptions of Functions

Similarly, add block comments before code of each function of some importance to describe the contract the function promises to support: (i.e., the operations the function is to do, the parameters it expects to be given, and the value(s) it promises to return). For example, if you were to write a WaitForMultipleObject(), you might comment it as it is in MSDN:

/*
DWORD
WaitForMultipleObjects(
    IN DWORD Count,
    IN Const *Handles,
    IN Bool WaitAll,
    IN DWORD Milliseconds
);

Routine Description:
This function allows the caller to wait until one or all of the specified objects are in the signaled state or the time-out interval elapses.

Arguments:
Count - Supplies the number of object handles in the array pointed to by Handles.
Handles - Supplies a pointer to an array of object handles
WaitAll - Supplies a flag, which when TRUE, the function returns when the states of all objects in the specified array are signaled. If FALSE, the function returns when the state of any one of the object is set to signal.
Milliseconds - Supplies a timeout interval in milliseconds.

Return Value:
The function returns blah, blah, blah
*/
5 Naming Conventions

5.1 Capitalization Style

We adopt both Pascal casing and Camel casing for different types of names:

- Pascal casing: The first character of every word is capitalized: as in NumberObjects.
- Camel casing: The first character of the first word is in lower case; the first character of the each subsequence word is capitalized, as in numberWaitableObjects.

5.2 General Rules

- **Use good names**: A good name is informative and helps to make your code easy to understand. Arguments of the wait-for function in previous section are good names, except perhaps Count. The longer name NumberObjects is a better – When you have time, take a look at [http://c2.com/cgi/wiki$?BadVariableNames](http://c2.com/cgi/wiki$?BadVariableNames) for examples of bad names to avoid.

- **Do not use names with underscore characters**. The only exceptions are names with all upper-case letters: Use the underscore character to separate words in those names.

- **Do not use case-sensitive names** Your component should be usable from both case-sensitive and case-insensitive languages. In other words, do not use names that differ only by case such as
  
  ```csharp
  System.WinForms.POINT P;
  System.WinForms.Point pp;
  ```

  Except for where is stated otherwise, do not use names according to rules of Hungarian notation (e.g., having an “h” front of a handle name, “c” in front of a class name, and so on.) This style was widely used in Window programming in the past and still appears often in MSDN. It is now obsolete. An exception is names of interfaces. Interface names begin with an “I” as in IBasicCartStateMachine.

5.3 Naming Guidelines

- **Symbolic Constants**
  - Capitalize all characters
  - Connect words with underscore.
  - As an example: #define MAXIMUM_NUMBER_DRAWERS 256

- **Enum names and input (types and values)**:
  - Use Pascal casing for Enum types and values names.
  - Use singular names or adjectives.
  - Name the last entry MaximumSoAndSo so you can easily check for validity of enum values.
As examples:

typedef enum AlertEventObjects {
    LockerEvent,
    TimeWorkManagerEvent,
    PatientRecordEvent,
    UserInterfaceEvent,
    MaximumAlertEvent
};

If you want to name a typedef, capitalize all letters in the name and connect words with underscore for readability, as illustrated by the example below.

typedef enum CartDrawerStates {
    DrawerOpen,
    DrawerClosed,
    MaximumState
} CART_DRAWER_STATES;

You can use the name in declarations such as

    CART_DRAWER_STATES CurrentDrawerState;

- **Struct and members names:**
  - Use Pascal casing for `struct` types and members names.
  - Use nouns for names and use specific names.
  - Do not use “Structure” as postfix.
  - Do not add module abbreviation as prefix.
  - As an example:

    typedef struct PersonalInformation {
        PPERSON_NAME PersonName;
        WCHAR PersonIdNumber[PERSONAL_ID_NUMBER_LENGTH];
        WCHAR BloodType[BLOOD_TYPE_LENGTH];
        CALENDAR_TIME BirthDay;
    } PERSONAL_INFORMATION, *PPERSONAL_INFORMATION;

    PPERSONAL_INFORMATION WendySailor;

    Note here is an exception of Hungarian notation: the prefix “p” in front of the pointer name.

- **Function names and input/output parameters:**
  - Use Pascal casing.
  - Name functions with verbs or verb phrases.
  - Take as examples of function and parameter names: `QueryLogInformation()` and `WaitForMultipleObjects` (IN DWORD Count,
Consider prefixing *API functions* of your component with an abbreviation of the component name, e.g., `iMANRequentUserNotification()`. Names of functions local to your module should not have the prefix.

### Variables and non-constant field names:

- Confine the use of simple counting variables names such as `i`, `j`, and `k` to trivial counting loops. If a count variable is used multiple times, consider giving it a meaningful name. The example in Section 3.4 illustrates this point.
- Use descriptive names, so that it is possible to determine their meanings and types.
- Use Camel casing for local variables, and Pascal casing for global variables.
- Take as examples: `isRunning` and `isOpen` for Booleans, `containerLocation[]` for integer location numbers.

### Object names:

- Use Pascal casing for global objects and Camel casing for local ones.
- As always, use meaningful names.
- Take as examples: `WaitableTimer`, `MonitorThread`, and `myOwnButton`.

### Event and event handler names:

- Use Pascal casing.
- Use an adjective, adjective phrase or verb phrase (and sometimes noun).
- Because some event names look like function names, some event names look like flags, and so on, add the suffix `Event` at the end of each name.
- Examples of event names are `StartEngineEvent`, `ButtonClickedEvent`, `TimerExpiredEvent`, `EnterPanelDragEvent`, `LeavePanelDragEvent`, `VisibilityChangedEvent`.
- Name the handler of an event with the suffix `Handler` or `EventHandler` together with the event name less the suffix `Event`. To illustrate, the handlers of the above listed events may be named as,
  
  ```
  StartEngineEventHandler
  ButtonClickedHandler
  TimerExpiredEventHandler
  EnterPanelDragHandler
  LeavePanelDragHandler
  ```
VisibilityChangedHandler

- As another example, the event handler for MouseEvent is MouseEventHandler in C and is declared in C# as

```csharp
public delegate void MouseEventHandler(object sender,
                                       MouseEventArgs e);
```

- **Namespace**
  - Use Pascal casing for namespaces, and separate logical components with periods.
  - Prefixing namespace names with a company name or established brand names can help prevent having multiple published namespaces with the same name.
  - Use plural namespace names if it is semantically appropriate. For example, use System.Collections rather than System.Collection.
  - As examples:
    - System.Web.UI
    - System.Web.UI.Design
    - iNuC
    - iNuC.UserControls
    - iNuC.PageControls

- **Class**
  - Use Pascal casing.
  - Use a noun or noun phrase to name a class.
  - As stated earlier in the general rules, do not use Hungarian style. For example, use the class name MyFileStream, not CMyFileStream.
  - As stated in Section 5.2, do not use the underscore character.
  - As examples:
    - public class MyFileStream
    - public class MyButton

- **Interface**
  - Use Pascal casing.
  - Name interfaces with nouns or noun phrases that describe behavior.
  - Prefix interface names with the letter I to indicate that the type is an interface. An example is
    ```csharp
    public interface IServiceProvider
    ```
  - Names of class and interface in a class/interface pair (i.e., the class is a standard implementation of the interface) should differ only by the letter I prefix on the interface name. An example is
    ```csharp
    public interface IMyComponent
    ```
// Implementation code goes here.
}

public class MyComponent : IMyComponent
{
    // Implementation code goes here.
}

- **Method**
  - Use Pascal casing.
  - Use verbs or verb phrases to name methods, same as names of functions.
  - As examples:
    ```
    GetPatientList()
    GetNurseNameString()
    CheckUserAuthentication()
    ```

- **State**
  - Use Pascal casing.
  - Name states with adjective or adjective phrase that describe behavior.
  - Use a *State* suffix in state names to make the names clearly distinguishable.
  - As examples:
    ```
    WaitingForUserLoginState
    RecordProcessedState
    VitalSignCreatedState
    ```

- **Activity and compound activity**
  - Use Pascal casing.
  - Use verbs or verb phrases to name activities and compound activities.
  - Use *Activity* as suffix in activity names.
  - As examples:
    ```
    DRGetDataActivity
    DRRPutDataActivity
    ```

- **Workflows**
  - Use Pascal casing.
  - Use nouns and noun phrases to name workflows
  - Use *Workflow* as suffix in workflow names.
  - In most cases, names of the local service interface and the associated workflow should differ only in their prefix (i.e., *I* in the interface name but not in the workflow name) and suffix (i.e., *Workflow* in the workflow name but not in the interface name.)
  - The examples below illustrate the usual and exceptional cases:
6 Declarations and Initialization

6.1 Number of Declarations per Line

The prefer styles is one declaration per line appended with comments. For example:

```c
RFID_TAG_ID drawerTagId;  // Id of RFID tag on a drawer
int drawerLocation;       // Location of a drawer
```

6.2 Initialization

Initialize local variables as soon as they are declared whenever possible. For the example in Section 3.4, you may initialize the elements of the Boolean array `isPrime` TRUE, if the array is local, rather than setting them of in a loop as that example does.

7 General Suggestions

Please keep in mind the following suggestions when you design and write your code. They are rather obvious once someone mentions them to you, but most likely, they may have occurred to you until now.

7.1 Code Quality

Please be obsessed with the quality of your code. We may be able to find many excuses for not being as concerned as this document is trying to make you do: “Mine is not likely to be defined as a medical device and, therefore, is not subjected to medical device software Quality Regulations. (For definitions of medical device software and their quality requirements, see Section 2.1 in [1]).” “Ours is but an experimental prototype, and we have get it done in a very short time” and so on, just to name a few. (Actually, it usually takes less time to write good code to begin with than trying to improve bad code.)

Moreover, if all goes as well as we wish, iNuC and FlexMMS will be put to actual use. Some parts of your component may be safety-critical. A bug hidden in your code may cause serious harm or loss of property. It makes better sense for us to treat it and each of its components as medical device software and strive for software quality required of them. So, let us walk the walk of the best programmers and get into the habit of writing correct and good code. This may take a little longer, but we cannot afford not to take the time.

We will try in the future to do code review by teammates on a routine basis. So, please write your codes and document them so that someone else will be able to understand and
critique them without your personal explanation.

7.2 Defensive Programming

Practice defensive programming. According to the definition in Wikipedia, a defensive programmer makes few assumptions and is always ready to handle all possible error states [2]. A better definition is: a defensive programmer always assumes that everything can be wrong or goes wrong and needs to be checked and handled: (Like a defensive driver, you drive with the assumption that all drivers are stupid, crazy or drunk.) Function (method) and library calls may not work as advertised; input data from the caller may be wrong (e.g., cannot be read, out of range, etc.); your function may not be able to write to the output space; the user may do something unexpected, and so on.

Being defensive also means making it hard for you (i.e., yourself) to do things wrong. A simple thing you can do to reduce errors in your code is by making good use of the compiler: Override the default setting and specify “the maximum warning level and covert warnings to errors”. As suggested by MSDN, document on compiler option, we should use /W4, the highest warning levels, for our new project. You will be extremely annoyed when you try to compile seemingly perfect code, the compiler sputters out lines and lines of warning but no executable. As pointed out by the author of [3], every warning from the compiler is a potential bug. Do not turn back to the easier default setting! Fix the potential bug so it will not show up in your software.

7.3 Source Code Review

It is a good best practice to structure your code so that each function does only one thing. Document the “contract” the function promises to honor. As stated in [3], “anytime you write software containing more than one function, you are defining your own API. … It needs to define its contract with the rest of your code and stick to that contract. Likewise, your code needs to be aware of the contract and do the appropriate tests.”

Your component is likely to be multi-threaded. Many experienced programmers believe that you must be able to prove informally that your code works by reading the source. A reason is that you cannot catch many bugs (e.g., those due to critical races) in multi-threaded programs by debugging and testing as they may not show up with a debugger hooked up. So, review your code yourself first.

7.4 Debugging and Testing

Debug Prints Thus far, you may have relied too heavily on print screen as a debugging method. As an example, we have seen that you have your code print out something every time a 20 Hz loop runs. Since print statements take considerable time, their presence distorts at least the timing behavior of your program. You really cannot conclude that the program
runs at the correct rate, or that it even runs at all once you take out the print statements. So, remove the debug print statements and start to install and use a debugger.

Assert Once you have a debugger installed, you may want to take advantage of ASSERT to help you catch bugs. ASSERT macro works only in a debug build. An assert macro statement lets you state that a certain condition must be true for the program execution to continue. When the condition is not true, the program aborts. With your debugger on, the program breaks into the debugger at where the assert fails. The article by R. Manderson [4] gives C++ examples and information displayed by Visual C++ debug library when an assert fails. Alternatively, look over the examples given by MSDN on ANSI assert macro or _ASSERT macro [5]. You will see from those examples that you can use the macros to catch many bugs such as bad pointers, strings and so on, during debugging.

Test Coverage We will strive to achieve not only statement coverage, as we usually do, but also decision coverage, or the more stringent condition/decision coverage criterion, for each of our components. According to [6], the criteria are defined as follows:

- Statement coverage: Every statement in the program has been executed at least once.
- Decision coverage: Every point of entry and exit in the program has been invoked at least once, and every decision in the program has been taken all possible outcomes at least once.
- Condition/decision coverage (C/DC): Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, and every decision in the program has taken all possible outcomes at least once.

It would be not surprising that medical device software need to meet the C/DC criterion–Later, after we have completed the preliminary designs of all components, we will looking into how we can make our code easier to test and what tools we will need to ensure coverage.

API Testing We need to be sure not only our programs work as intended when provided correct inputs, but also they do what should be done when provided with bad input. A quick reading of your code can tell whether your programs handle bad input at all and do so correctly. Does every API function of yours check for the validity of what are passed to it? The things it should check include [3]

- Each pointer parameter is not NULL.
- Every pointer points to something valid: A read pointer points to somewhere your function can read and each output pointer points to valid memory where your function can write.
- The value of each input parameter is within valid range.
- A handle input is really a handle and so on.

Your API function should refuse to perform any operation and return a meaningful error code if it finds any inconsistency in input from the caller.
8 Specific Suggestions

Below is a short and random list of specific suggestions on C programming. You can find more in [7, 8]. Please share your favorite rules with us as well.

8.1 Use Enum and Symbolic Constants

You probably have seen programs containing a sequence of \texttt{#define} functions, defining related discrete values, e.g.,

\begin{verbatim}
#define DrawerOpen 1    // Locker drawer open state
#define DrawerClosed 2 // Locker drawer closed state
\end{verbatim}

Sometimes, tens of them like these ones. It is better to use\texttt{enum} instead as illustrated by the examples in Section 5.3. The article on defensive programming [3] points out that \texttt{enum}’s make your program more readable. More importantly, each of them defines a data type. So

\begin{verbatim}
typedef enum AlertEventObjects {
    ... 
} AlertEventObjects;
\end{verbatim}

defines the \texttt{AlertEventObjects} type and its values. If you later pass a parameter of this type to a function, the compiler can check for you whether the parameter value is a valid one.

Similarly, do not use numerical constants. Use \texttt{#define} function to give constants meaningful names.

8.2 Use Secure String Copy Functions

Use one of the security-enhanced functions (e.g., \texttt{strncpy_s}) for copying character strings to avoid buffer overflow problem. (See \url{http://msdn.microsoft.com/en-us/library/5dae5d43.aspx} for their definitions and sample code illustrating their usage.)

8.3 Test Function Return Values

Check the value returned by every function you call every time. We frequently see code segments like this one:

\begin{verbatim}
struct verySmallStruct *ptr;
...
ptr = (struct verySmallStruct *) malloc (sizeof (verySmallStruct));
memset (ptr, 0, sizeof (struct verySmallStruct));
\end{verbatim}

\texttt{malloc()} may fail even when the size of memory requested is very small. We should not take the chance not to check the pointer. After all, it is no trouble at all to do it right:

\begin{verbatim}
struct verySmallStruct *ptr;
...
ptr = (struct verySmallStruct *) malloc (sizeof (verySmallStruct));
if (ptr == NULL) {
    goto Cleanup;
}
\end{verbatim}
memset (ptr, 0, sizeof (struct verySmallStruct));
...
return;

Cleanup: ...
...
return errorCode;

8.4 Use Goto Sometimes

We were all taught to avoid goto. But, sometimes the code is actually more readable using it. In general, it comes handy when you want to break out levels of switch, for and while. The previous example illustrates appropriate use of goto statement in initialization. In general, initialization code structured like the pseudo-code segment below is easy to understand.

```c
bufferSpace = malloc (sizeof (...));
if (bufferSpace == NULL) {
    goto Cleanup;
}
memset (ptr, 0, sizeof (...));
...
    // Do some work
firstHandle = ...;    // Create something
if (firstHandle == NULL) {
    goto Cleanup;
}
...
    // Do some more work
secondHandle = ...;   // Create something else
if (secondHandle == NULL) {
    goto Cleanup;
}
...
return succeeded;

Cleanup: ...
    // Clean up and prepare to exit
if (bufferSpace) {
    free (bufferSpace);
}
...
    // Free resources and set error code
return errorCode;
```

8.5 Terminate Threads Cleanly

Your component may be multi-threaded. For the sake of concreteness, let us say that during initialization, the main thread of the component creates other threads (called them follower threads) and creates and initializes all the resources the component needs. If it succeeds, the main thread goes on to monitor events and may be queue work items to be processed by follower threads. In general, the follower threads may do anything, including acquiring critical section, attaching itself to some DLL, and so on.

When your component terminates, either because it is shut down or because some step failed during initialization, the main thread can easily terminate by first freeing all the resources it holds and then returning from its procedure. How about the other threads?
In principle, there are three ways to terminate those threads. The first one is to have the main thread call `TerminateThread(aFollowerThreadHandle, exitCode)` to terminate each of the follower threads. A short summary of what MSDN said about this function is “**Do NOT use it.**” (See [http://msdn.microsoft.com/en-us/library/ms686717(VS.85).aspx](http://msdn.microsoft.com/en-us/library/ms686717(VS.85).aspx) for specifics).

The better alternatives are to have each follower thread terminate itself. It can do so by calling `ExitThread()` explicitly. Or better yet, **it simply returns from its procedure.** In any case, you need to provide a way for the main thread to tell the follower threads to run, finish doing whatever must be done before quitting and then return. This is easy if follower threads in your component are workers that wait for work items to process. Just have them wait not only for events that signal work to be done, but also a **shutdown** event. When the thread awakes and finds **shutdown** event set, it just proceeds to return. On XP, you can have the leader thread wait, after setting the **shutdown** event, until all the follower threads have exited and then return.

**References:**

Appendix B

iNuC data exchange API

The iNuC data exchange API uses web service on ASP.NET. Each API is a method in iNuCDataExchange web service class.

Revision History

- Dec 18, 2009:
  - Move DosePeriod from Prescriptions table to DoseFrequencies table. DosePeriod is hospital wide parameter.
  - Add DispenseDayList, IsScheduable, DoseInstruction in Prescriptions table.
  - Add GetHasActivedPrescription() method.
- Dec 23, 2009
  - Add GetAllDepartments() method to get all departments.
  - Add GetWards() method to get all wards.
  - Rename DosePeriod in DoseFrequencies to DoseInterval.
  - Rename DispenseDayList in Prescriptions to DoseDateList.
Table of Contents

iNuC data exchange API ................................................................................................................ 1
Revision History .............................................................................................................................. 1
Table of Contents ............................................................................................................................. 2
iNuCDataExchange Class ................................................................................................................. 3
   iNuCDataExchange Member ......................................................................................................... 3
   iNuCDataExchange Constructor ................................................................................................. 3
   iNuCDataExchange Methods ...................................................................................................... 3
      Login Method .......................................................................................................................... 4
      GetNurseInformation Method ................................................................................................. 4
      GetNurseInformationList Method ............................................................................................ 5
      GetPatientsInformation Method ............................................................................................. 6
      GetPatientInformationList Method ........................................................................................... 6
      GetHasActivedPrescription Method ....................................................................................... 7
      GetPrescriptions Method ......................................................................................................... 8
      GetMedicationInformation Method ......................................................................................... 8
      GetScheduleEvents Method .................................................................................................... 10
      GetSupplyAccountRecords Method ........................................................................................ 11
      GetMemoRecords Method ....................................................................................................... 12
      GetInjectionRecords Method .................................................................................................. 12
      GetVitalSignRecords Method ................................................................................................. 13
      GetVitalSignParameters Method ............................................................................................. 14
      GetDepartments Method .......................................................................................................... 14
      GetAllDepartments Method ..................................................................................................... 15
      GetWards Method .................................................................................................................... 15
      GetScheduleEventMetaInformation Method .......................................................................... 16
      GetCannedMessages Method ................................................................................................... 16
      GetSupplyAccountList Method ............................................................................................... 17
      GetDoseFrequencies Method ................................................................................................... 18
      PutScheduleEvents Method .................................................................................................... 18
      PutSupplyAccountRecords Method ........................................................................................ 19
      PutMemoRecords Method ........................................................................................................ 19
      PutInjectionRecords Method .................................................................................................. 19
      PutVitalSignRecords Method .................................................................................................. 20
      PutVitalSignParameters Method ............................................................................................. 20
      PutScheduleEventMetaInformation Method .......................................................................... 20
      PutCannedMessages Method .................................................................................................. 21
      PutSupplyAccountList Method ............................................................................................... 21
Notes .................................................................................................................................................. 21
iNuCDataExchange Class

Syntax:

```java
public class iNuCDataExchange : WebService
```

iNuCDataExchange Member

None

iNuCDataExchange Constructor

Initiate web service object.

Syntax:

```java
public iNuCDataExchange()
```

iNuCDataExchange Methods

The following table describes brief notes for each method. For detail parameters and query logic, please see next subsection.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Login</td>
<td>Check if the account is a valid nurse account with correct password.</td>
</tr>
<tr>
<td></td>
<td>GetNurseInformation</td>
<td>Get nurse information from NurseId.</td>
</tr>
<tr>
<td></td>
<td>GetNurseInformationList</td>
<td>Get all nurses information in one Department.</td>
</tr>
<tr>
<td></td>
<td>GetPatientsInformation</td>
<td>Get patients information from PatientId.</td>
</tr>
<tr>
<td></td>
<td>GetPatientInformationList</td>
<td>Get all patients information in one Department.</td>
</tr>
<tr>
<td></td>
<td>GetPrescription</td>
<td>Get prescriptions from PatientId and time duration.</td>
</tr>
<tr>
<td></td>
<td>GetMedicationInformation</td>
<td>Get medication information from medication code.</td>
</tr>
<tr>
<td></td>
<td>GetScheduleEvents</td>
<td>Get all events scheduled from id and time duration.</td>
</tr>
<tr>
<td></td>
<td>GetSupplyAccountRecords</td>
<td>Get records from PatientId.</td>
</tr>
<tr>
<td></td>
<td>GetMemoRecords</td>
<td>Get records from PatientId.</td>
</tr>
<tr>
<td></td>
<td>GetInjectionRecords</td>
<td>Get records from PatientId.</td>
</tr>
<tr>
<td></td>
<td>GetVitalSignRecords</td>
<td>Get records from PatientId.</td>
</tr>
<tr>
<td></td>
<td>GetVitalSignParameters</td>
<td>Get records from PatientId.</td>
</tr>
<tr>
<td></td>
<td>GetDepartments</td>
<td>Get department information from DepartmentId.</td>
</tr>
<tr>
<td></td>
<td>GetScheduleEventMetaInformation</td>
<td>Get rule and title of schedule event from DepartmentId.</td>
</tr>
<tr>
<td></td>
<td>GetCannedMessages</td>
<td>Get canned messages from DepartmentId.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GetSupplyAccountList</td>
<td>Get list of predefined supply account from DepartmentId.</td>
<td></td>
</tr>
<tr>
<td>GetDoseFrequencies</td>
<td>Get mapping of does frequency and time.</td>
<td></td>
</tr>
<tr>
<td>PutScheduleEvents</td>
<td>Put schedule event.</td>
<td></td>
</tr>
<tr>
<td>PutSupplyAccountRecords</td>
<td>Put supply account records.</td>
<td></td>
</tr>
<tr>
<td>PutMemoRecords</td>
<td>Put memo records.</td>
<td></td>
</tr>
<tr>
<td>PutInjectionRecords</td>
<td>Put injection records.</td>
<td></td>
</tr>
<tr>
<td>PutVitalSignRecords</td>
<td>Put vital sign records.</td>
<td></td>
</tr>
<tr>
<td>PutVitalSignParameters</td>
<td>Put predefined type of vital sign.</td>
<td></td>
</tr>
<tr>
<td>PutScheduleEventMetaInformation</td>
<td>Put rule or title schedule event.</td>
<td></td>
</tr>
<tr>
<td>PutCannedMessages</td>
<td>Put canned message.</td>
<td></td>
</tr>
<tr>
<td>PutSupplyAccountList</td>
<td>Put supply account record.</td>
<td></td>
</tr>
</tbody>
</table>

**Login Method**

Check if the account is a valid nurse account with correct password.

**Syntax:**

```csharp
public Bool Login( string AccountId, string Password )
```

**Parameters:**

- AccountId: account of a nurse.
- Password: password of login account.

**Return Value:**

- True if it is a valid nurse account.
- False if the nurse account is not exists, or with wrong password.

**Exception:**

- InvalidArgumentsException: the input is null or not a string.

**GetNurseInformation Method**

**Syntax:**

```csharp
public DataSet GetNurseInformation( string NurseId )
```

**Parameter:**

- NurseId: a id equal to AccountId in Login.

**Return Value:**

A DataSet named iNuc contained with one DataTable: Nurses. The NursId in Nurses DataTable is equal to NurseId in input parameter.

Nurses DataTable:
### GetNurseInformationList Method

**Syntax:**
```
public DataSet GetNurseInformationList(string DepartmentId)
```

**Parameter:**
- `DepartmentId`: department id that uniquely identifies a department.

**Return Value:**
A DataSet named iNuc contained with one DataTable: Nurses. The `DepartmentId` in Nurses DataTable is equal to `DepartmentId` in input parameter.

**Nurses DataTable:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnglishGivenName</td>
<td>此護士的英文名</td>
</tr>
<tr>
<td>EnglishLastName</td>
<td>此護士的英文姓氏</td>
</tr>
<tr>
<td>ChineseGivenName</td>
<td>此護士的中文名</td>
</tr>
<tr>
<td>ChineseLastName</td>
<td>此護士的中文姓氏</td>
</tr>
<tr>
<td>Birthday</td>
<td>此護士的生日</td>
</tr>
<tr>
<td>NurseId</td>
<td>此護士的id，護士能用此帳號登入 iNuC</td>
</tr>
<tr>
<td>WardId</td>
<td>此護士負責的病房</td>
</tr>
<tr>
<td>DepartmentId</td>
<td>此護士醫療部門id，能夠獨立辨認醫療部門</td>
</tr>
</tbody>
</table>

**Exception:**
- `InvalidArgumentsException`: the input arguments are invalid.
- `DataTablesNotExistsException`: the DataTables to be returned is not exists.
GetPatientsInformation Method

Syntax:

```csharp
public DataSet GetPatientsInformation (string PatientIds)
```

Parameters:

- **PatientIds**: a list of PatientId, which uniquely identifies a patient. Each id in PatientIds is divided by a space, such as “patientId1 patientId2 patientId3”.

Return Value:

A DataSet named iNuc contained with one DataTable: Patients. The PatientId in Patients DataTable is equal to one of PatientId in PatientIds in input parameter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnglishGivenName</td>
<td>病人的英文名</td>
</tr>
<tr>
<td>EnglishLastName</td>
<td>病人的英文姓氏</td>
</tr>
<tr>
<td>ChineseGivenName</td>
<td>病人的中文名</td>
</tr>
<tr>
<td>ChineseLastName</td>
<td>病人的中文姓氏</td>
</tr>
<tr>
<td>BloodType</td>
<td>A, B, O, RH- etc..</td>
</tr>
<tr>
<td>Birthday</td>
<td>病人生日</td>
</tr>
<tr>
<td>PatientId</td>
<td>病人此次住院的代号</td>
</tr>
<tr>
<td>WardId</td>
<td>此病人的病房</td>
</tr>
<tr>
<td>RoomId</td>
<td>此病人的病室</td>
</tr>
<tr>
<td>BedId</td>
<td>此病人的病床</td>
</tr>
<tr>
<td>DepartmentId</td>
<td>此病人所属醫療部門id，能夠獨立辨認部門</td>
</tr>
<tr>
<td>Sex</td>
<td>性別(男性請寫”男”，女性請寫”女”)</td>
</tr>
</tbody>
</table>

Exception:

- **InvalidOperationException**: the input arguments are invalid.
- **DataTablesNotExistsException**: the DataTables to be returned is not exists.

GetPatientInformationList Method

Syntax:

```csharp
public DataSet GetPatientInformationList (string DepartmentId)
```

Parameters:

- **DepartmentId**: department id that uniquely identifies a department.

Return Value:

A DataSet named iNuc contained with one DataTable: Patients. The DepartmentId in Patients DataTable is equal to DepartmentId in input parameter.
Patients DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnglishGivenName</td>
<td>病人的英文名</td>
</tr>
<tr>
<td>EnglishLastName</td>
<td>病人的英文姓氏</td>
</tr>
<tr>
<td>ChineseGivenName</td>
<td>病人的中文名</td>
</tr>
<tr>
<td>ChineseLastName</td>
<td>病人的中文姓氏</td>
</tr>
<tr>
<td>BloodType</td>
<td>A, B, O, RH-, etc..</td>
</tr>
<tr>
<td>Birthday</td>
<td>病人生日</td>
</tr>
<tr>
<td>PatientId</td>
<td>病人此次住院的代号</td>
</tr>
<tr>
<td>WardId</td>
<td>此病人的病房</td>
</tr>
<tr>
<td>RoomId</td>
<td>此病人的病室</td>
</tr>
<tr>
<td>BedId</td>
<td>此病人的病床</td>
</tr>
<tr>
<td>DepartmentId</td>
<td>此病人所屬醫療部門id，能夠獨立辨認部門</td>
</tr>
<tr>
<td>Sex</td>
<td>性別(男性請寫”男”，女性請寫”女”)</td>
</tr>
</tbody>
</table>

Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

GetHasActivedPrescription Method

Syntax:
public DataSet GetHasActivedPrescription ( string PatientIds, DateTime DateHasBeenActived)

Parameters:
PatientIds: a list of PatientId, which uniquely identifies a patient. Each id in PatientIds is devided by a space, such as “patientId1 patientId2 patientId3”.
DateHasBeenActived: the date to query.

Return Value:
A DataSet named iNuc contained with one DataTable: Prescriptions. The PatientId in Prescriptions DataTable is equal to one of PatientId in PatientIds in input parameter.

Prescription DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientId</td>
<td>病人id</td>
</tr>
<tr>
<td>PrescriptionId</td>
<td>藥物id</td>
</tr>
<tr>
<td>CreationTime</td>
<td>醫令開立時間</td>
</tr>
<tr>
<td>PrescriptionStartTime</td>
<td>醫令開始時間</td>
</tr>
<tr>
<td>PrescriptionEndTime</td>
<td>醫令結束時間</td>
</tr>
<tr>
<td>Status</td>
<td>0:使用中  1:被取消</td>
</tr>
</tbody>
</table>
DoseSize | 給藥量  
--- | ---  
DoseSizeText | 存在資料庫中的給藥量文字  
DoseFrequency | QD, QID, etc...  
DoseDateList | 若需要指定給藥日期，如BIW，則需要指定此欄位。資料格式為一個 byte array，每個 bit 表示該天是否需給藥，0為不給藥，1為給藥。資料的最低位表示醫令開始日是否給藥，第2位表示醫令開始後的第二天是否給藥，以此類推。  
IsScheduable | 是否有指定時間給藥，如果有則為 true，否則為 false。如只施用一次，或需參照醫囑施用的，此欄位為 false。  
DoseUnit | 此藥的單位，如 tab、qs  
DoseRoute | SKIN, PO, etc...  
MedicationCode | 用藥八碼  
DoseInstruction | 附註醫囑，包含說明施用條件、連續注射說明，或其他需告知護理士的醫令相關事項。  

Exception:  
InvalidArgumentsException: the input arguments are invalid.  
DataTablesNotExistsException: the DataTables to be returned is not exists.

GetPrescriptions Method

Syntax:  
public DataSet GetPrescription (string PatientIds, DateTime StartDatet ime, DateTime EndDatet ime)  

Parameters:  
- PatientIds: a list of PatientId, which uniquely identifies a patient. Each id in PatientIds is devided by a space, such as “patientId1 patientId2 patientId3”.  
- StartDatet ime: earliest creation time or modified time of prescription.  
- EndDatet ime: latest creation time or modified time of prescription.

Return Value:  
A DataSet named iNuc contained with one DataTable: Prescriptions. The PatientId in Prescriptions DataTable is equal to one of PatientId in PatientIds in input parameter.

Prescription DataTable:  

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientId</td>
<td>病人id</td>
</tr>
<tr>
<td>PrescriptionId</td>
<td>藥囑id</td>
</tr>
<tr>
<td>CreationTime</td>
<td>醫令開立時間</td>
</tr>
<tr>
<td>PrescriptionStartTime</td>
<td>醫令開始時間</td>
</tr>
</tbody>
</table>
### PrescriptionEndTime

**Status**
- 0: 使用中
- 1: 被取消

**DoseSize**
給藥量

**DoseSizeText**
存在資料庫中的給藥量文字

**DoseFrequency**
QD, QID, etc..

**DoseDateList**
若需要指定給藥日期，如BIW，則需要指定此欄位。資料格式為一個 byte array，每個 bit 表示該天是否需給藥，0為不給藥，1為給藥。

資料的最低位表示醫令開始日是否給藥，第2位表示醫令開始後的第二天是否給藥，以此類推。

**IsScheduable**
是否有指定時間給藥，如果有則為 true，否則為 false。如只施用一次，或需參照醫囑施用的，此欄位為 false。

**DoseUnit**
此藥的單位，如 tab、qs

**DoseRoute**
SKIN, PO, etc...

**MedicationCode**
用藥八碼

**DoseInstruction**
附註醫囑，包含說明施用條件、連續注射說明，或其他需告知護理士的醫令相關事項。

### Exception:

- InvalidArgumentsException: the input arguments are invalid.
- DataTablesNotExistsException: the DataTables to be returned is not exists.

### GetMedicationInformation Method

#### Syntax:

```csharp
public DataSet GetMedicationInformation (string MedicationCodes)
```

#### Parameters:

- MedicationCodes: a list of MedicationCode(用藥八碼)，which uniquely identifies a medicine. Each id in MedicationCodes is devided by a comma, such as “MedicationCode1,MedicationCode2,MedicationCode3”.

#### Return Value:

A DataSet named iNuc contained with one DataTable: MedicationInformation. The Medicationcode in Medicationinformation DataTable is equal to one of Medicationcode in MedicationCodes in input parameter.

### MedicationInformation DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedicationCode</td>
<td>用藥八碼</td>
</tr>
<tr>
<td>ScienceName</td>
<td>學名</td>
</tr>
</tbody>
</table>
Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

GetScheduleEvents Method

Syntax:
public DataSet GetScheduleEvents (string NurseId, string PatientIds, DateTime StartDatetime,
DateTime EndDateTime)

Parameters:
NurseId: a id equal to AccountId in Login.
PatientIds: a list of PatientId, which uniquely identifies a patient. Each id in PatientIds is divided by a space, such as “patientId1 patientId2 patientId3”.
StartDatetime: earliest ExpectedExecutionTime of event.
Enddatetime: latest ExpectedExecutionTime of event.

Return Value:
A DataSet named iNuc contained with one DataTable: ScheduleEvents. The NurseOrPatientId in ScheduleEvents DataTable is equal to NurseId in input parameter or one of PatientId in PatientIds in input parameter.

ScheduleEvents DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此事件的id</td>
</tr>
<tr>
<td>ScheduleEventType</td>
<td>0:預約事件  1:有明確時間點的給藥事件  2:有需要才給藥的給藥事件</td>
</tr>
<tr>
<td>DisplayTitle</td>
<td>顯示在畫面的標題</td>
</tr>
<tr>
<td>Content</td>
<td>顯示在畫面的內容</td>
</tr>
<tr>
<td>NurseOrPatientId</td>
<td>事件所有人的id，可能是護士或病人</td>
</tr>
<tr>
<td>Status</td>
<td>0:事件時間已過且未完成 1:事件時間未到且未完成 2:事件已完成 3:事件被取消</td>
</tr>
<tr>
<td>CreationTime</td>
<td>事件建立時間</td>
</tr>
<tr>
<td>DefaultExpectedExecutionTime</td>
<td>建立時的預計完成時間（不可變）, 用來計算事件可以移動的時間區間</td>
</tr>
<tr>
<td>ExpectedExecutionTime</td>
<td>目前的預計完成時間, 隨著事件被移動而改變</td>
</tr>
<tr>
<td>ExpiredTime</td>
<td>最晚被執行的時間</td>
</tr>
<tr>
<td>ExpectedExecutionTimeInterval</td>
<td>完成時間的區間</td>
</tr>
</tbody>
</table>
Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

GetSupplyAccountRecords Method

Syntax:
public DataSet GetSupplyAccountRecords (string PatientIds, DateTime StartDatetime, DateTime EndDateTime)

Parameters:
PatientIds: a list of PatientId, which uniquely identifies a patient. Each id in PatientIds is divided by a space, such as “patientId1 patientId2 patientId3”.
StartDatetime: earliest log time or of prescription.
EndDatatime: latest log time of prescription.

Return Value:
A DataSet named iNuc contained with one DataTable: SupplyAccountRecords. The PatientId in SupplyAccountRecords is equal to one of PatientId in PatientIds in input parameter.

SupplyAccountRecords DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此记录的id</td>
</tr>
<tr>
<td>PatientId</td>
<td>使用此耗材的病人</td>
</tr>
<tr>
<td>LogTime</td>
<td>纪录建立时间</td>
</tr>
<tr>
<td>SupplyId</td>
<td>此耗材的id</td>
</tr>
<tr>
<td>SupplyAmount</td>
<td>此耗材的使用数量</td>
</tr>
<tr>
<td>NurseId</td>
<td>填写此记录的护士</td>
</tr>
</tbody>
</table>

Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

GetMemoRecords Method

Syntax:
public DataSet GetMemoRecords (string PatientIds, DateTime StartDatetime, DateTime EndDateTime)

Parameters:
- PatientIds: a list of PatientId, which uniquely identifies a patient. Each id in PatientIds is divided by a space, such as “patientId1 patientId2 patientId3”.
- StartDatetime: earliest log time or of prescription.
- EndDatetime: latest log time of prescription.

Return Value:
A DataSet named iNuc contained with one DataTable: MemoRecords. The PatientId in MemoRecords is equal to one of PatientId in PatientIds in input parameter.

MemoRecords DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此紀錄的id</td>
</tr>
<tr>
<td>PatientId</td>
<td>此註記針對的病人</td>
</tr>
<tr>
<td>Memo</td>
<td>註記内容</td>
</tr>
<tr>
<td>LogTime</td>
<td>註記建立時間</td>
</tr>
<tr>
<td>NurseId</td>
<td>填寫此註記的護士</td>
</tr>
</tbody>
</table>

Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

GetInjectionRecords Method

Syntax:
public DataSet GetInjectionRecords (string PatientIds, DateTime StartDatetime, DateTime EndDateTime)

Parameters:
- PatientIds: a list of PatientId, which uniquely identifies a patient. Each id in PatientIds is divided by a space, such as “patientId1 patientId2 patientId3”.
- StartDatetime: earliest log time or of prescription.
- EndDatetime: latest log time of prescription.

Return Value:
A DataSet named iNuc contained with one DataTable: InjectionRecords. The PatientId in
InjectionRecords is equal to one of PatientId in PatientIds in input parameter.

InjectionRecords DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此紀錄的id</td>
</tr>
<tr>
<td>PatientId</td>
<td>此紀錄針對的病人</td>
</tr>
<tr>
<td>LogTime</td>
<td>紀錄建立時間</td>
</tr>
<tr>
<td>InjectionLocation</td>
<td>注射位置</td>
</tr>
<tr>
<td>NurseId</td>
<td>填寫此紀錄的護士</td>
</tr>
</tbody>
</table>

**Exception:**
- InvalidArgumentsException: the input arguments are invalid.
- DataTablesNotExistsException: the DataTables to be returned is not exists.

**GetVitalSignRecords Method**

**Syntax:**
```csharp
public DataSet GetVitalSignRecords (string PatientIds, DateTime StartDatetime, DateTime EndDateTime)
```

**Parameters:**
- PatientIds: a list of PatientId, which uniquely identifies a patient. Each id in PatientIds is devided by a space, such as “patientId1 patientId2 patientId3”.
- StartDatet ime: earliest log time or of prescription.
- EndDatet ime: latest log time of prescription.

**Return Value:**
A DataSet named iNuc contained with one DataTable: VitalSignRecords. The PatientId in VitalSignRecords is equal to one of PatientId in PatientIds in input parameter.

VitalSignRecords DataTable:

| Name             | Description                                                          |
|------------------|                                                                    |
| Id               | 此紀錄的id                                                          |
| PatientId        | 接受此次測量的病人                                                  |
| EnglishAbbreviation | 該Vital Sign的英文縮寫，可以獨立辨認不同的 VitalSignParameters |
| LogTime          | 紀錄建立時間                                                        |
| Value            | 測量值                                                             |
| NurseId          | 填寫此紀錄的護士                                                    |

**Exception:**
- InvalidArgumentsException: the input arguments are invalid.
GetVitalSignParameters Method

Syntax:
   public DataSet GetVitalSignParameters ()

Parameters:
   none

Return Value:
   A DataSet named iNuC contained with one DataTable: VitalSignParameters.
   VitalSignParameters DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnglishAbbreviation</td>
<td>該Vital Sign的英文縮寫，可以獨立辨認不同的VitalSignParameters</td>
</tr>
<tr>
<td>ChineseName</td>
<td>該Vital Sign的中文名稱</td>
</tr>
<tr>
<td>ParameterUnit</td>
<td>該Vital Sign的單位</td>
</tr>
<tr>
<td>Maximum</td>
<td>該Vital Sign的最大值</td>
</tr>
<tr>
<td>Minimum</td>
<td>該Vital Sign的最小值</td>
</tr>
<tr>
<td>Type</td>
<td>屬於哪一個測量分類，例如TPR等</td>
</tr>
</tbody>
</table>

Exception:
   InvalidArgumentsException: the input arguments are invalid.
   DataTablesNotExistsException: the DataTables to be returned is not exists.

GetDepartments Method

Syntax:
   public DataSet GetDepartments (string DepartmentId)

Parameters:
   DepartmentId: department id that uniquely identifies a department.

Return Value:
   A DataSet named iNuC contained with one DataTable: Departments. The DepartmentId in Departments DataTable is equal to DepartmentId in input parameter.

Departments DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此部門的id</td>
</tr>
<tr>
<td>Name</td>
<td>此部門的名稱</td>
</tr>
</tbody>
</table>
Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

GetAllDepartments Method

Syntax:
public DataSet GetAllDepartments ()

Parameters:
None

Return Value:
A DataSet named iNuc contained with one DataTable: Departments. The Departments DataTable contains all departments in hospital.

Departments DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此部門的id</td>
</tr>
<tr>
<td>Name</td>
<td>此部門的名稱</td>
</tr>
</tbody>
</table>

Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

GetWards Method

Syntax:
public DataSet GetWards ()

Parameters:
None

Return Value:
A DataSet named iNuc contained with one DataTable: Wards. The Wards DataTable contains all wards in hospital.

Wards DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此病房的id</td>
</tr>
<tr>
<td>Name</td>
<td>此病房的名稱</td>
</tr>
</tbody>
</table>

Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.
GetScheduleEventMetaInformation Method

Syntax:
```csharp
public DataSet GetScheduleEventMetaInformation (string DepartmentId)
```

Parameters:
- DepartmentId: department id that uniquely identifies a department.

Return Value:
- A DataSet named iNuc contained with two DataTables: ScheduleEventTitles and ScheduleEventRules. The DepartmentId in ScheduleEventTitles DataTable is equal to DepartmentId in input parameter. The ScheduleEventTitleId in ScheduleEventRules DataTable is equal to one of id in ScheduleEventTitles.

ScheduleEventTitles DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此单一事件的id</td>
</tr>
<tr>
<td>DepartmentId</td>
<td>建立此預設事件的部門</td>
</tr>
<tr>
<td>EventDisplayTitle</td>
<td>此事件要顯示的標題</td>
</tr>
<tr>
<td>Content</td>
<td>此事件要顯示的內容</td>
</tr>
</tbody>
</table>

ScheduleEventRules DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此規則的id</td>
</tr>
<tr>
<td>ScheduleEventTitleId</td>
<td>對應的 ScheduleEventTitleId, 每條 rules 都會對應到一個 scheduleEventTitle （例如胃鏡檢查）</td>
</tr>
<tr>
<td>ContinuousEventDisplayTitle</td>
<td>此連續事件的標題</td>
</tr>
<tr>
<td>ContinuousEventContent</td>
<td>此連續事件的內容文字</td>
</tr>
<tr>
<td>IsBefore</td>
<td>在對應的 ScheduleEventTitle 前執行為true，之後為false</td>
</tr>
<tr>
<td>ExpectedExecutionHour</td>
<td>和對應的 ScheduleEventTitle間隔的小時數</td>
</tr>
<tr>
<td>ExpectedExecutionMinute</td>
<td>和對應的 ScheduleEventTitle間隔的分鐘數</td>
</tr>
</tbody>
</table>

Exception:
- InvalidArgumentsException: the input arguments are invalid.
- DataTablesNotExistsException: the DataTables to be returned is not exists.

GetCannedMessages Method

Syntax:
public DataSet GetCannedMessages (string DepartmentId)

**Parameters:**
DepartmentId: department id that uniquely identifies a department.

**Return Value:**
A DataSet named iNuc contained with one DataTable: CannedMessages. The DepartmentId in CannedMessages DataTable is equal to DepartmentId in input parameter.

CannedMessages DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此訊息的id</td>
</tr>
<tr>
<td>DepartmentId</td>
<td>設定此罐頭訊息的部門</td>
</tr>
<tr>
<td>MessageType</td>
<td>訊息分類，如器具借用、給藥等，為純文字</td>
</tr>
<tr>
<td>MessageContent</td>
<td>訊息內容</td>
</tr>
</tbody>
</table>

**Exception:**
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

**GetSupplyAccountList Method**

**Syntax:**
public DataSet GetSupplyAccountList(string DepartmentId)

**Parameters:**
DepartmentId: department id that uniquely identifies a department.

**Return Value:**
A DataSet named iNuc contained with one DataTable: SupplyAccountList. The DepartmentId in SuppluAccountList DataTable is equal to DepartmentId in input parameter.

SupplyAccountList DataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>此耗材的id</td>
</tr>
<tr>
<td>Category</td>
<td>耗材的主分類</td>
</tr>
<tr>
<td>SubCategory</td>
<td>耗材的次分類，若沒有次分類則為NULL</td>
</tr>
<tr>
<td>Content</td>
<td>耗材名稱</td>
</tr>
<tr>
<td>groupID</td>
<td>耗材群組id，在同一個群組內的耗材，一次只能使用其中一項</td>
</tr>
<tr>
<td>DepartmentId</td>
<td>設定此耗材種類的部門</td>
</tr>
</tbody>
</table>

**Exception:**
InvalidArgumentsException: the input arguments are invalid.
GetDoseFrequencies Method

Syntax:
public DataSet GetDoseFrequencies()

Parameters:
none

Return Value:
A DataSet named iNuC contained with one DataTable: DoseFrequencies.

DoseFrequenciesDataTable:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoseFrequency</td>
<td>給藥頻率，如 BID, QOD</td>
</tr>
<tr>
<td>Time</td>
<td>給藥時刻對應的單日給藥時間，若需多次給 藥，則用”-”分隔。如 BID 要在 9:00 和 17:00 給藥的話，則標記“9-17”</td>
</tr>
<tr>
<td>DoseInterval</td>
<td>給藥週期，若每天給藥則為1，若每兩天給一次藥為2，若每 n 天給固定次數的藥，則為 n。</td>
</tr>
</tbody>
</table>

Exception:
InvalidArgumentsException: the input arguments are invalid.
DataTablesNotExistsException: the DataTables to be returned is not exists.

PutScheduleEvents Method

Syntax:
public bool PutScheduleEvents (DataSet ScheduleEvents)

Parameters:
ScheduleEvents: a DataSet named iNuc contained with one DataTable: ScheduleEvents. The Datarow to be updated or inserted is marked in RowState.

Return Value:
True if update successes.
False if this method fails immediately.

Exception:
InvalidArgumentsException: the input is null or not a DataSet.
UnauthorizedPutException: the update is invalid, such as write to table not exists.
PutSupplyAccountRecords Method

Syntax:
public bool PutSupplyAccountRecords (DataSet SupplyAccountRecords)

Parameters:
SupplyAccountRecords: a DataSet named iNuc contained with one DataTable: SupplyAccountRecords. The Datarow to be updated or inserted is marked in RowState.

Return Value:
True if update successes.
False if this method fails immediately.

Exception:
InvalidArgumentException: the input is null or not a DataSet.
UnauthorizedPutException: the update is invalid, such as write to table not exists.

PutMemoRecords Method

Syntax:
public bool PutMemoRecords (DataSet MemoRecords)

Parameters:
MemoRecords: a DataSet named iNuc contained with one DataTable: MmeoRecords. The Datarow to be updated or inserted is marked in RowState.

Return Value:
True if update successes.
False if this method fails immediately.

Exception:
InvalidArgumentException: the input is null or not a DataSet.
UnauthorizedPutException: the update is invalid, such as write to table not exists.

PutInjectionRecords Method

Syntax:
public bool PutInjectionRecords (DataSet InjectionRecords)

Parameters:
InjectionRecords: a DataSet named iNuc contained with one DataTable: InjectionRecords. The Datarow to be updated or inserted is marked in RowState.

Return Value:
True if update successes.
False if this method fails immediately.

Exception:
InvalidArgumentsException: the input is null or not a DataSet.
UnauthorizedPutException: the update is invalid, such as write to table not exists.

PutVitalSignRecords Method

**Syntax:**
```csharp
public bool PutVitalSignRecords (DataSet VitalSignRecords)
```

**Parameters:**
- VitalSignRecords: a DataSet named iNuc contained with one DataTable: VitalSignRecords. The Datarow to be updated or inserted is marked in RowState.

**Return Value:**
- True if update successes.
- False if this method fails immediately.

**Exception:**
- InvalidArgumentsException: the input is null or not a DataSet.
- UnauthorizedPutException: the update is invalid, such as write to table not exists.

PutVitalSignParameters Method

**Syntax:**
```csharp
public bool PutVitalSignParameters (DataSet VitalSignParameters)
```

**Parameters:**
- VitalSignParameters: a DataSet named iNuc contained with one DataTable: VitalSignParameters. The Datarow to be updated or inserted is marked in RowState.

**Return Value:**
- True if update successes.
- False if this method fails immediately.

**Exception:**
- InvalidArgumentsException: the input is null or not a DataSet.
- UnauthorizedPutException: the update is invalid, such as write to table not exists.

PutScheduleEventMetaInformation Method

**Syntax:**
```csharp
public bool PutScheduleEventMetaInformation (DataSet PutScheduleEventMetaInformation)
```

**Parameters:**
- PutScheduleEventMetaInformation: a DataSet named iNuc contained with one DataTable: PutScheduleEventMetaInformation. The Datarow to be updated or inserted is marked in RowState.

**Return Value:**
True if update successes.
False if this method fails immediately.

Exception:
InvalidArgumentsException: the input is null or not a DataSet.
UnauthorizedPutException: the update is invalid, such as write to table not exists.

**PutCannedMessages Method**

Syntax:
public bool PutCannedMessages (DataSet CannedMessages)

Parameters:
CannedMessages: a DataSet named iNuc contained with one DataTable: CannedMessages. The Datarow to be updated or inserted is marked in RowState

Return Value:
True if update succeeds.
False if this method fails immediately.

Exception:
InvalidArgumentsException: the input is null or not a DataSet.
UnauthorizedPutException: the update is invalid, such as write to table not exists.

**PutSupplyAccountList Method**

Syntax:
public bool PutSupplyAccountList (DataSet supplyAccountList )

Parameters:
supplyAccountList: a DataSet named iNuc contained with one DataTable: SupplyAccountList. The Datarow to be updated or inserted is marked in RowState

Return Value:
True if update succeeds.
False if this method fails immediately.

Exception:
InvalidArgumentsException: the input is null or not a DataSet.
UnauthorizedPutException: the update is invalid, such as write to table not exists.

**Notes**

Detail of all DataSet described above, e.g. column data type, is in iNuCDataAPI.xsd.
Appendix C

iNuC 簡易手冊
iNuC 總功能表

個人時程表

今日行程

瀏覽行程

備藥

查詢

病人資訊

藥物資訊

連結院內系統

設定

起始頁面設定

提示功能設定

快顯列設定

背景樣式設定

字型設定

安全設定
目錄

INU C 總功能表 ............................................................................................................. 2

目錄 ............................................................................................................................... 3

序言 ................................................................................................................................. 5

INU C 功能鍵說明 ......................................................................................................... 6

登入 ................................................................................................................................. 7

個人時程表 .................................................................................................................. 8

　今日時程 ..................................................................................................................... 8
　個人行程 ..................................................................................................................... 9
　給藥 ............................................................................................................................ 10
　　　針劑注射 .............................................................................................................. 12
　　　註記 ....................................................................................................................... 12
　新增行程 .................................................................................................................... 13
　修改行程 .................................................................................................................... 16
　刪除行程 .................................................................................................................... 16
　還原行程 .................................................................................................................... 16
　瀏覽行程 ..................................................................................................................... 17
　備案 ............................................................................................................................ 18

查詢 ................................................................................................................................ 18

病人資料 ....................................................................................................................... 18
　測量記錄 ................................................................................................................... 19
　醫療單單 ................................................................................................................... 21
　用藥記錄 ................................................................................................................... 21
　病人時程表 ............................................................................................................. 22
　帳單記錄 ................................................................................................................... 22
　藥物資訊 ................................................................................................................... 23

連結院內系統 ............................................................................................................... 24

設定 ................................................................................................................................ 24

　起始頁面設定 .......................................................................................................... 24
　提示功能設定 .......................................................................................................... 25
　快顯列設定 ............................................................................................................. 25
　背景樣式設定 .......................................................................................................... 25
　字型設定 ................................................................................................................... 25
安全設定 .................................................................................................................. 25

功能列 .................................................................................................................... 26

上一層 ..................................................................................................................... 26

上一頁 ..................................................................................................................... 26

列印 ........................................................................................................................ 26

註記 ........................................................................................................................ 27

快顯 ........................................................................................................................ 28

鎖定 ........................................................................................................................ 29

登出 ........................................................................................................................ 29

關機 ........................................................................................................................ 29
序言

「iNuC」 (Intelligent Nursing Cart), 以下稱為 iNuC, 爲一適用於研究型教學醫院的智慧型行動護理推車。使用者以護理人員為主，其目的是減少給藥失誤，並提升病房用藥安全。
iNuC 提供現有行動護理推車具有的功能：如利用條碼機制確保藥物施用在正確的病人上，記錄病患之最新生理狀況、用藥反應、用藥流程以及記錄藥物的庫存量並幫助盤點等等。除此之外，iNuC 也提供給使用者數種先進與獨特的功能。如排程與提醒工具，可幫助使用者從事藥物準備工作，並排定病患的給藥時程表等重要的核心工作相關事務。
本手冊將以 iNuC 介面的圖片為順序，以文字的方式來說明整個 iNuC 系統，並逐步介紹 iNuC 各項功能，詳細說明 iNuC 的操作方式，讓使用者能在短時間內對 iNuC 有初步的認識與了解並進而上手操作。
iNuC 功能鍵說明

- 上一層：回到上一層介面
- 上一頁：回到上一頁介面
- 列印：列印資料
- 註記：新增註記
- 快顯：跳出功能捷徑，可以快速跳至其他頁面
- 鎖定：鎮定熒幕保留目前的使用狀態避免被其他人更動
- 登出：離開系統以供其他人員使用
- 關機：登出並關閉系統結束程式
- 常用字檢索：套入一些預設的詞彙
- 清除欄位：將填寫的欄位內容全部清除
- 取消
- 確定
登入

進入iNuC系統後，首先顯示的是登入介面，使用者可在此選擇以[主要使用者]或是[訪客]登入。

![iNuC登入界面]

**登入主要使用者**
1. [登入身分]點選 [主要使用者]。
2. 輸入[使用者名稱]。
3. 輸入[使用者密碼]。
4. [是否採用個人化設定]點選 [是]。
5. 按 [確定]登入系統。

**訪客登入**
使用者只能使用[連結院內系統]服務。

**不採用個人化設定**
系統介面(如：桌面以及字型大小)將使用系統預設的設定。
個人時程表

您可在此選擇[今日行程]管理使用者和病人的行程，也可選擇[瀏覽行程]來瀏覽已排程之行程，此外也能在此執行[備藥]。

今日時程

您可於此管理您的行程，進行新增、修改、刪除、還原等動作。

管理行程

左邊的區塊中顯示今日使用者所要做的工作時間與工作事項。

- **左列行程時間**：表示這一列所有事務要執行的時間，預設以兩小時為區隔。
- **右邊的區塊中為沒有排定時間的行程。**

個人事件圖示，點選後可進入觀看此行程

病人事件圖示，點選後可進入觀看此行程
病人給藥事件圖示，點選後可進入執行給藥動作

附註：事件與給藥主要是由左下角的圖示區分。

行程
使用者所要做的事情，分成如上(個人事件、病人事件、病人給藥事件)三種，事件可以藉由拖拉到別的時間列去改變他的排程時間。例如：您可以將12:00護理師圖示拖拉到10:00病人圖示旁後，並輸入時間，此事件其執行時間將會改成新的時間。

個人行程

點選個人事件、或 圖示後，可進入觀看行程內容，若此時完成此行程，可點選 [此事件完成] 表示已完成此事件，同時此事件在行程表上消失。
給藥

點選 圖示後會跳出一視窗要求輸入或掃描病人條碼以開啓藥盒。

在輸入正確的條碼後即進入給藥行程畫面。

完成給藥行程

此區域中的藥物為本次給藥行程中使用的藥物。

1. 點選[施用情況]。

2. 在施用情況選項中選擇本次施用情況。

3. 以右下角[數字輸入]或以鍵盤來輸入[病患服藥量]及[此次消耗量]的數量。

4. 如有需要註記的部分可點選右上角[註記]輸入

5. 按下[確定]完成給藥情況的輸入。

6. 勾選藥物名稱。

7. 在給藥行程中勾選本次給藥行程所使用之藥物。

8. 重複1~7步驟直到所有給藥行程中的藥物均已勾選。

9. 按下右下角[完成給藥]進行確定。
增加非固定時間給藥

<table>
<thead>
<tr>
<th>藥名</th>
<th>藥品名</th>
<th>用法</th>
<th>單位</th>
<th>種類</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dipyrpylamine and Aspirin</td>
<td>Aggrenex Modified Release Capsule</td>
<td>6</td>
<td>QM AM PEN</td>
<td>SN</td>
</tr>
<tr>
<td>Dipyrpylamine and Aspirin</td>
<td>Aggrenex Modified Release Capsule</td>
<td>4</td>
<td>QM AM PEN</td>
<td>SN</td>
</tr>
</tbody>
</table>

1. 勾選非固定時間給藥中的藥物。
2. 按下[新增為本次給藥]。

移除勾選的藥物

1. 勾選給藥型行程中藥物左側方塊(可複選)
2. 點選移除勾選的給藥。

您可於此次給藥行程中，移除非固定時間給藥新增進來的藥物，但不能移除已經排定於此次給藥行程內的藥物。

藥物查詢

點選學名、簡品名連結，會連結到藥物查詢資料庫顯示此種藥物的詳細資訊。

輸入[病患服藥量]及[此次消耗量]

點選給藥行程中的[病患服藥量]及[此次消耗量]後，可從螢幕數字小鍵盤直接輸入數字。
針劑注射

點選進去之後會出現人體皮下注射位置記錄圖，圖中顯示的位置表示身體各部位的代碼。

記錄本次注射
1. 點選本次注射位置，於點選處出現藍色圓圈
2. 確認下方顯示的代碼是否正確。
3. 按下確定完成本次針劑注射紀錄。

紅色圓圈
依顏色深淺及圓圈大小標示出過去注射時間。

註記
點進去之後可觀看對此病人的所有備忘提醒及特殊紀錄(請見功能列的註記功能)
新增行程

您可於此增加個人事件或病人事件到時程表裡。

新增個人事務行程
1. 填寫[事務名稱]。
2. 填寫[事務內容]。
3. 點選此個人事務的預計執行的時間。
4. 按下右下角確定完成新增行程

設定為周期性行程
如果此事件是周期性的事件，在按下後會跳出一個視窗讓使用者設定周期性規則。

1. 選擇[循環模式]為[每天]、[每周]或[每月]。
2. 設定循環範圍中的開始日期
3. 設定結束日期，可選擇此事件在反覆執行幾次後結束或是直接設定結束時間。

新增病人事務行程
1. 選擇您負責的病人名稱
2. 填寫[事務名稱]。
3. 填寫[事務內容]。
4. 點選此個人事務的預計執行的時間。
5. 按下右下角確定完成新增行程
直接引入預設通則
如果想使用系統中預設的事件通則，勾選[直接引入預設通則]後會出現視窗如下。

1. 點選預設事件名稱。
2. 輸入事件內容。
3. 如果此預設事件有相關連續事件，將會出現在下方。
4. 按下右下角確定

設定其他相關連續事件
如果此事件有其他相關連續事件，勾選[設定其他相關連續事件]後即可進行設定。

新增連續事件
按下[新增事件]後會出現視窗如下

1. 輸入連續事件名稱。
2. 輸入連續事件內容。
3. 選擇此連續事件預計在此事件前(後)幾小時幾分執行。
刪除連續事件
點選事件名稱左方按鈕即可刪除此項連續事件。

修改連續事件
您可點選以修改事件名稱、內容、預計執行在此行程之前或之後的時間。修改後系統會自動計算，並在最右邊顯示預計此相關連續事件執行時間。

修改行程
點下修改後（變成綠底）再點選要修改的事件，即可修改個人事件或群人事件，但不可修改給藥行程

1. 修改此事務的名稱
2. 修改此事務的內容
3. 修改次事務預計執行時間
4. 按下右下角[確定]

刪除行程
點下刪除後（變成橘底）再點選要刪除的事件，確認刪除後此事件會從時程表消失，但不能刪除給藥行程

還原行程
進入資源回收桶裡可將已經被刪除的事件復原。
1. 點選需要還原的事件，點選後事件顯示橘色底色
2. 按下右下角[還原此事件]還原點選的事件
瀏覽行程

查詢使用者及負責的病人全部的行程

點選護理師圖示可以進入您的每月行程表，每月行程表中有護理師圖案表示當天有事件，後面的數字代表事件數。

點選病人圖示可以進入病人的每月行程表，同一天的行程會被放在同一列裡，圖示則會顯示事件的種類與時間

點選病人每月行程表中的圖示可進一步顯示該事件詳細資料。
備藥
您可以進入此部分來退出未鎖定之藥盒進行備藥。

![藥盒圖示](image)

藍色代表藥盒抽屜打開狀態，紅色表示關閉的藥盒抽屜。

1. 輸入或掃描需要備藥的病人條碼

   ![條碼輸入框](image)

2. 按下對應藥盒確認要將藥物放入此藥盒中，此時再關上藥盒，該對應藥盒圖示即會變成紅色表示此藥盒抽屜已關上

查詢
查詢使用者負責的病人的相關資料以及所有藥物的資訊

病人資訊
在此可查看並列出病人的相關資訊

![病人名單](image)

1. 在病人的名單中，點選某位病人圖示即可查看病人資訊
2. 如上圖顯示病人的基本資訊
3. 新增或查詢測量紀錄
4. 瀏覽病人藥單及醫囑紀錄
5. 瀏覽病人用藥紀錄
6. 瀏覽病人的行程表
7. 瀏覽病人過去的帳單紀錄

測量記錄

您可從查詢進入此部分，在此您可新增或瀏覽測量紀錄

新增 TPR 或一般測量紀錄

1. 圈選 TPR 項目

2. 按下中間視窗右下角[確定]鈕後出現如下視窗

   ![TPR測量紀錄視窗](image)

3. 輸入 TPR 各項目測量紀錄
4. 按下右下角確定鈕完成新增

新增其他測量紀錄

1. 圈選其他項目
2. 勾選想要新增的測量資料(可復選)
3. 按下中間視窗右下角[確定]

4. 輸入勾選項目的測量值
5. 按下右下角確定鈕完成新增

增減欄位

1. 如果想要再新增其他筆測量記錄，按下畫面右上角
2. 利用勾選來新增或減少想要增加的測量資料數量

瀏覽過去測量資料

1. 勾選想要瀏覽的項目

2. 按下右上角更新，出現表格如下

3. 也可切換成趨勢圖瀏覽
TPR 趨勢圖瀏覽
點選 TPR 趨勢圖，直接連覽 TPR 測量記錄

醫囑藥單
您可於此瀏覽醫生根據該病人需要所下的指示以及開立的藥單

<table>
<thead>
<tr>
<th>開立時間</th>
<th>藥名</th>
<th>副品名</th>
<th>用量</th>
<th>頻次</th>
<th>候復</th>
<th>結果</th>
<th>備註</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.04.01 12:00</td>
<td>Carvedilol</td>
<td>Dilatrend Tablet 25 mg/tab</td>
<td>1</td>
<td>QD</td>
<td>PO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009.04.02 12:00</td>
<td>Carvedilol</td>
<td>Dilatrend Tablet 6.25 mg/tab</td>
<td>2</td>
<td>Q4H</td>
<td>SKIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009.04.02 12:00</td>
<td>Lysine Acetylsaliclyte</td>
<td>Stin for Injection 0.9 g lysine acetylsa...</td>
<td>2</td>
<td>Q4H</td>
<td>SKIN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

用藥紀錄
顯示病人過去的用藥紀錄

<table>
<thead>
<tr>
<th>藥名</th>
<th>副品名</th>
<th>用量</th>
<th>頻次</th>
<th>候復</th>
<th>結果</th>
<th>備註</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lysine Acetylsaliclyte</td>
<td>Stin for Injection 0.9 g lysine acetylsa...</td>
<td>2</td>
<td>Q4H</td>
<td>SKIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carvedilol</td>
<td>Dilatrend Tablet 25 mg/tab</td>
<td>2</td>
<td>Q4H</td>
<td>SKIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carvedilol</td>
<td>Dilatrend Tablet 6.25 mg/tab</td>
<td>2</td>
<td>Q4H</td>
<td>SKIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carvedilol</td>
<td>Dilatrend Tablet 6.25 mg/tab</td>
<td>2</td>
<td>Q4H</td>
<td>SKIN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
病人時程表

您可在此查詢病人的行程

查詢病人行程

1. 點選想要查詢的行程

2. 出現如下圖示窗顯示病人行程

3. 按下確定關閉視窗

帳單紀錄

在此新增病人的帳單紀錄

新增帳單紀錄

1. 點選此處的分類按鈕，會在下方視窗列出有相關詳細項目。
點選想要加到帳單的項目後再按下[+]鈕後，即可新增此項目至右方帳單中。

藥物資訊
列出藥物的相關資訊，同時進入藥物資訊畫面首頁會先連結到台大醫院藥劑部網頁

查詢藥物資訊

1. 輸入藥物代碼或藥名後按下搜尋鈕進行搜尋後出現如下畫面。

2. 如果想要查詢的藥物未出現在下方，可再使用[藥名查詢]、[關鍵字查詢]、[懷孕分級查詢]來進行搜尋。

3. 如想要出線的藥名出現在下方，點選藥物藥名查看詳細資料。
連結院內系統

連結到幾個固定網站的網頁系統查詢相關資料，畫面右側有網站連結。

設定

內有各種環境及提示功能等設定

起始頁面設定

設定登入系統時，最先出現的畫面

設定起始頁面

1. 選擇一連結當作您登入後第一個顯示的畫面。
2. 選擇完畢後按下儲存鈕將設定儲存。
提示功能設定
設定使用者及病人的事件提示方式，您可在此設定以訊息、列印、聲音、LED 閃爍來提示行程，同時可設定提醒的時間。

快顯列設定
新增移除快顯列中的連結。
設定以快顯列顯示方式
設定快顯列
用拖曳方式將圖示在左、右兩區域內移動以設定快顯列中的連結，加入左邊的圖示會出現在快顯列中。

背景樣式設定
設定系統背景樣式
1. 設定背景圖案。
2. 設定完成後按下確定鈕儲存設定。

字型設定
設定系統字型大小
設定字型大小，共有大、中、小三種字體選擇。

安全設定
設定系統進入鎖定狀態的時間。
設定閒置幾分鐘之後自動鎖定螢幕，非個人化設定時預設為閒置10分鐘之後會自動鎖定螢幕

功能列

登入系統後，可在畫面下方看到一排功能列

上一層

回到上一層介面

上一頁

回到上一頁介面

列印

列印資料
選擇要使用的印表機
1. 選擇列印的份數
2. 選擇是否列印前要先預覽報表
3. 選擇報表種類
4. 選擇要列印資料的病人，可複選
5. 選擇全部的病人
6. 選擇病人列印資料的時間範圍
7. 確定後開始列印(如有勾選預覽列印則會跳出預覽視窗)

註記
按下註記後會跳出使用者負責的病人，選擇其中一位病人後，開啓
註記視窗讓使用者對該位病人進行新增(瀏覽、修改)註記。
新增註記

1. 輸入註記內容
2. 儲存輸入的註記

瀏覽、修改註記

1. 點選內容部分進行註記修改（注意：只能修改本日或前一日的註記）
2. 儲存修改後的註記

若註記內容過長無法完全顯示在內容視窗內，將游標停在該項目上後可顯示完整內容

快顯

跳出功能捷徑，可以快速跳至其他頁面

目前所在位置：功能表→快顯
1. 點選要前往的頁面，即可快速前往該頁面

**鎖定**

鎖定螢幕保留目前的使用狀態避免被其他人更動

使用者名稱： 300273

使用者密碼：

解除鎖定
1. 輸入使用者名稱
2. 輸入使用者密碼
3. 按下確定後，解除鎖定狀態

**登出**

離開系統以供其他人員使用

**關機**

登出並關閉系統結束程式