An Ontology Supported Meta-Interface for the Development and Installation of Customized Web Based Telemedicine Systems

Jackei H.K. Wong[1], Wilfred W. K. Lin[1], Allan K.Y. Wong[1], Tharam S. Dillon[2]

[1] Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, H.K.S.A.R.
[2] Digital Ecosystems & Business Intelligence Institute, Curtin University of Technology, Australia
{cshkwong,cswklin,csalwong}@comp.polyu.edu.hk, tharam.dillon@cbs.curtin.edu.au

Abstract. The novel and generic meta-interface (MI) paradigm proposed in this paper automates the generation of customized telemedicine software systems (CTSS), directly from the customized application interface (CAI) specifications given. The MI paradigm was tested and verified in the TCM (Traditional Chinese Medicine) telemedicine environment of Nong's Company Limited of the PuraPharm Group, a local Hong Kong TCM telemedicine developer that funded this research. The CAI specification is made by "gluing" together icons selected from the enterprise icon library (IL). The CTSS generation, in effect, extracts the corresponding portion of the subsumption hierarchy from the master ontology, an enterprise standard. Since CTSS prototypes were verified in the Nong's TCM telemedicine environment, they were built with the Nong's master TCM ontology core (onto-core) as the basis and reference. In this light, the ontological extraction for a CAI specification is turned into the local TCM onto-core for the CTSS prototype. The enterprise TCM onto-core, the local TCM onto-core, and the icons in IL all contain formal knowledge derived from the enterprise TCM vocabulary. In Nong's case the enterprise vocabulary is the standard of CTSS terminology, gathered from TCM classics, treatises, and case histories by domain experts with consensus certification. Using CAI as the single input to automate the whole CTSS generation process would eliminate MSPM (multi-site project management) problems. Since the Nong's MC (mobile clinics) based telemedicine system is web-based and pervasive, the CTSS is also referred to as the Nong's web-based telemedicine systems (WTS).

Keywords:. Meta-interface paradigm, software development, CAI, CTSS, WTS, telemedicine system, ontology, enterprise vocabulary, automated

1 Introduction

We propose in this paper the innovative meta-interface (MI) paradigm for developing remotely deployable ubiquitous web-based telemedicine systems (WTS). The goal is to customize each client's WTS from a single master ontological core (onto-core), which would then be maintained by the enterprise (i.e. enterprise/master onto-core). Nong's Company Ltd., of the PuraPharm Group in the Hong Kong SAR, is one such

enterprise and a local leader in supplying TCM (Traditional Chinese Medicine) WTS to hospitals and clinics across the globe. The company had created its own enterprise TCM onto-core with consensus certification [1] to support MC (mobile clinics) based telemedicine systems. From this enterprise TCM onto-core, Nong's customizes WTS variants specified by the customers. After deployment each customized WTS is automatically linked over the mobile Internet to the PuraPharm/Nong's (PP/N) mobile-business (MB) core.The proposed MI paradigm now offers a solution to the PP/N management's long search for a way to effectively automate the WTS customization process. Their lengthy search is understandable because successful development of qualitative software systems and applications is no easy task. This development process needs to satisfactorily address different contemporary issues [2] that deal with problem domains, varied operational environments, and cultural differences (e.g. natural languages). These issues could be complicated when developing web-based applications that are distributed over diverse geographic locations and involve complex ICT (information communications technology) concerns and MSPM (multi-site project management) activities. Even with the same project requirements MSPM linguistic variations could cause ambiguity, resulting in incorrect implementation or non-interoperable software modules [3]. The lack of an enterprise vocabulary to coordinate and disambiguate software development activities means a high cost-effectiveness ratio, indicated by the following surveys:

a) *High cost*: In Australia and USA it is common for 50% or more of enterprise expenditures to be spent on software development and maintenance.
b) *Prone to failure*: Less than 50% of software development projects in the Western world were completed successfully.
c) *Same trend*: The trend of roughly 70% software project failures will continue into the future, remaining the same as it was three decades ago.

Although the software engineering discipline has been evolving fast [1], the generic waterfall model in Figure 1 can still abstract the system development cycle in four phases:

a) Phase 1 - Requirement specification and analysis. Its goal is to analyze and accurately extract the following elements from the narrated requirements: the necessary and sufficient number of functions for the target system; formal parameters for each of these functions; and the execution serializability (logical control flow) among the identified functions to ensure coherent and meaningful results. A function normally performs only one application-specific task of transforming the actual parameters into the expected result. For example, if $f(x_1, x_2)$ is a function, (x_1, x_2) are two formal parameters that would assume actual values/parameters before execution (i.e. the transformation process). The functions and their intertwined logical relationships form the *functional specification*; constraints specified for these relationships form the *constraints specification* to govern the ambit of system behavior/dynamics. The *functional* and *constraints* (F&C) *specifications* together form the domain of semantics for the system to know exactly *what to do*.
b) Phase 2 – Design specification. Details of *how the final system should work* are addressed by: i) organizing the system semantics into small manageable modules (modularization) by the principle of *information hiding*; ii) specifying how the

modules should synchronize and associate; iii) proposing the subsumption hierarchy for the modules that can be separated into two basic groups by their nature: control-oriented (CO) and data-oriented (DO); higher-level CO does little computation but controls the timely invocation of other modules; the objective of the lower-level DO is to produce useful information from actual parameters for use by the higher-level modules; iv) proposing the system architecture to support the final system operation; and v) evaluating data structures and algorithms/protocols to support information retrieval and inter-modular synchronizations for coherent operations.

c) Phase 3 – Implementation. This phase aims to correctly translate the design specification into an intermediate form for: i) human understanding and manipulation, and ii) conversion into the machine-executable representation. The intermediate form is a program or software of a specific language (e.g. C++ or Visual Basic). To humans, the program syntactically represents the system semantics; the machine executes its compiled form (executable code).
d) Phase 4 – Testing and debugging. Test cases are created to validate and verify that the implemented system prototype indeed fulfils all the functions indicated in the requirement specification. Debugging a distributed application is more an art than science for we can rarely apply traditional approaches. From the literature the only recognized technique to debug distributed software effectively is program visualization (e.g. [4,5]).

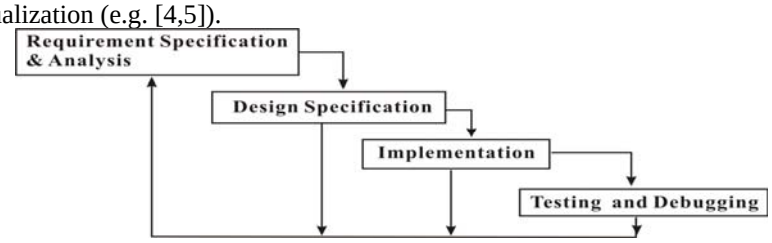

Fig. 1. Generic waterfall development life cycle

The feedback loops (Figure 1) show that if errors are found in the engineering process, changes have to be repeatedly made in the upper source(s). Too many loop-backs make the process expansive. Thus, the emphasis is on producing correct F&C specifications, and this can be achieved by using practical formal methods (e.g. Petri net). The errors in translating the F&C specifications into the design specification can be reduced by using semi-formal, semi-automatic tools such as the DBDesigner (DBD) by Microsoft. The DBD converts the semantic net in the form of a subsumption hierarchy (e.g. DOM (document object code) tree drawn in the DBD format) into logically matched XML-annotated code. The SQL system (also by Microsoft) then can convert the annotated code directly into a usable database. If the CO and DO modules are programmed in VB.net (Visual Basic for the Internet), they interact readily with the SQL database. DBD, SQL, VB.net together fulfill the *congruent automation principle* (CAP) to be explained later. If the activities in Figure 1 are supported by a management scheme that controls system migration, software changes, system versioning, and maintenance, a *configuration control* (CC) framework is formed [1].

2 Related Work

Successful software engineering in the 21st century needs to overcome a set of formidable challenges, including rapid and uncertain technological changes/emergence, cultural diversity leading to ambiguous understanding of the target system, and heterogeneity in hardware and software that prevents interoperability. The paper by Boehm [6] sums these formidable challenges nicely, and one of his guidelines is to avoid THWADI ("*that's how we've always done it*"). This applies well to developing remotely deployable ubiquitous web-based telemedicine systems, which is an emerging phenomenon of the 21st century. In reality, the THWADI guideline is unavoidable, for computing requirements evolve rapidly in different eras, governed by the Moore's Law [7]: i) Amdahl's era (early 1960s) – synchronizing sequential processes correctly was the focus; ii) Gustafson-Barsis era (mid-1980s) – parallel computing (i.e. High Performance Computing (HPC)) to yield speedup; iii) megacomputing era (mid-1990s) – distributed systems formed with an Internet basis; and iv) pervasive era (early 2000) – concern for the mobility of hardware and software entities supported by location-aware capability. Despite the rapid evolution driven by various contemporary forces, we still find that: i) the waterfall model is the foundation; ii) optimal placement of program tasks is a focal issue; and ii) coherent synchronization of these tasks is needed for correct results. From the literature we identified ten major forces that affect the success of developing remotely deployable web-based telemedicine systems. These forces are represented as entries in the set $F = \{A, B, C, D, E, F, G, H, I, J\}$ as depicted in Figure 2:

A. Synchronization and serializability methods: These govern how entities in the system interact coherently. Examples include CR (critical region), RPC (remote procedure call), Corba, and MPI. The method used depends on the problem domain and the intended environment of operation.
B. Channel reliability methods: These shorten the service roundtrip time in client/server interaction. Usually dynamic or adaptive methods are more effective than static methods [8].
C. User participation: This is a necessity for effective fast prototyping so that immediate user feedback improves the prototype. It is ideal if the user participates in all stages of the waterfall model.
D. Software engineering by parts: This is integration of software parts (modules/artifacts) built by other groups into the system being built. It can be physical code inclusions (into the system software) or logical remote invocation via predefined linkages. The parts can be in various programming languages but do not affect the final system performance [4].
E. Tools/methods for creating/managing data structures and databases: These represent the paradigm that data structures on the blueprint are realized automatically into physical databases; for example, converting a DBD drawing directly into a physical SQL database (i.e. Microsoft environment).
F. Testing and debugging tools: These support different testing and debugging situations. For example, program/system behavior visualization is suitable for

monitoring distributed agent-based software in which agents are mobile in a real-time sense [4].

G. System security issues: The aim is allow a system run smoothly without unnecessary interruptions.
H. MSPM (multi-site project management): Usually teams based in different geographic locations are involved in the development of a successful enterprise software system. To eliminate ambiguity a vocabulary to bridge cultural and language differences among working groups needs to be created. The creation of such a vocabulary is regarded by many researchers as an ontological approach (i.e. the vocabulary is the "enterprise ontology") [9].
I. ICT (information communications technology): This discipline combines appropriate technologies to build an efficient web application.
J. Trend and era issues/laws: Inevitably, as the computing industry advances through various trend-setting eras and laws into today's mobility era with mobile hardware for location-aware networks and mobile software agents that migrate at will, some of the older methods and tools will be invalidated.

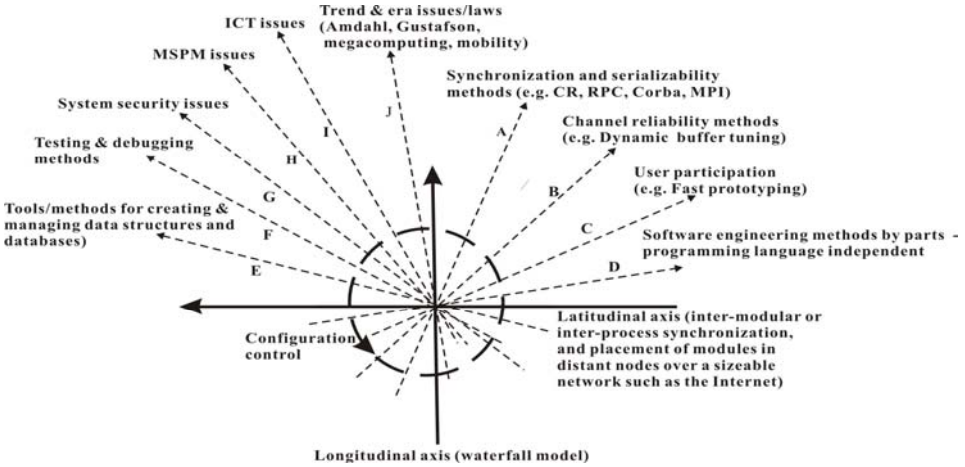

Fig. 2. Ten external forces that affect software system success

The longitudinal and latitudinal axes in Figure 2 form the backbone of the configuration control (CC) to balance these ten forces into equilibrium. Although the waterfall model is the basis for the CC, the two key issues of modular task placement into network nodes and ensuring correct task synchronization to achieve coherent results still need to be addressed. Unfortunately, no previous experience on devising an effective CC scheme for pervasive telemedicine system development has been found in the literature. The Nong's in-house experience, which used the traditional waterfall model as the basis to balance (by trial-and-error) some of the forces shown in Figure 2 is the only useful clue so far. Besides, very limited experience can be found in the literature about formulating telemedicine system architectures. The only useful example that we encountered was the UMLS (Unified Medical Language System) [10].

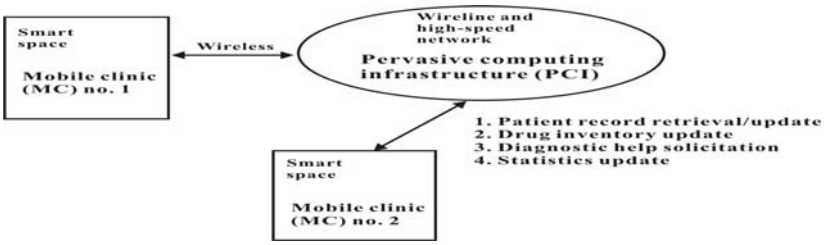

Fig. 3. A pervasive telemedicine system model

3 Nong's Telemedicine Framework Background

Telemedicine, a term that was consolidated around 1999 [12], aim is to electronically deliver healthcare (i.e. e-health) to every corner of the globe. Its realization over the mobile Internet, however, is an art for there is little experience published in the literature for this budding discipline. Since the mobile Internet supports both wireline and wireless communication technologies, interacting agents of a telemedicine system on the web require reliable mobility and communication supports. A telemedicine operation is basically a digital ecosystem, in which agents/entities of different species (e.g. mobile clinics (MC) and surrogate agents) collaborate closely [13]. In response to the potential business benefits of telemedicine the Nong's Company Ltd. developed several WTS, which are now deployed in different locations over the globe. The fundamental Nong's WTS concept is depicted in the Figures 3 and 4. Figure 3 shows the mobile nature of the Nong's telemedicine approach, which has a central PCI (pervasive computing infrastructure) support on a high-speed wireline network. Once an MC has moved into a smart space (a wireless communication cell with location-aware capability) it could interact with other MCs and the PCI at will. Typical MC tasks invoked via the application interface of the system include: i) patient record retrieval/update; ii) drug inventory update (both central (in PCI) and local (on MC)); iii) diagnostic help solicitation from remote physicians (i.e. collaborated diagnosis); and iv) statistics for effective MC management and disease control (e.g. as required by the Hong Kong SAR government). An MC (i.e. local telemedicine unit) is manned typically by: a physician, a dispenser, a paramedic, and the customized telemedicine software system (CTSS) which is conceptually depicted in Figure 4 as CAI. The CTSS is architecturally similar to the UMLS by having three distinctive layers (Figure 5) but functionally it differs by supporting real-time frontline clinical practice.

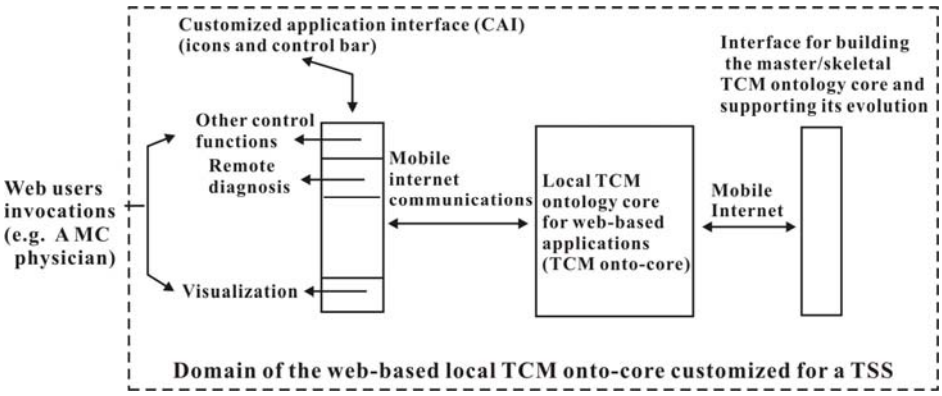

Fig. 4. Conceptualized CTSS (customized telemedicine software system)

The CTSS (or WTS (web-based telemedicine system)) architecture has three layers:

a) Bottom layer (i.e. CTSS bottom-domain in Figure 5): This is the local TCM onto-core customized from the enterprise's time-honored total knowledge as logically indicated by [V] in Figure 5.

b) Middle layer (i.e. CTSS middle-domain in Figure 5): This is the semantic net (network) that fully and logically represents the local CTSS TCM onto-core in the machine process-able form. The parsing mechanism (parser) is the software that draws the logical conclusion for the query input from the top layer (e.g. $Q\{p_1, p_2, p_3\}$; $\{p_1, p_2, p_3\}$ are parameters to drive the parsing mechanism).

c) Top layer (i.e. CTSS top-domain): This is the customized application interface (CAI) specification for the target CTSS (i.e. F&C specifications together) to syntactically represent the local CTSS semantic net for human understanding. The CAI specification is made up of icons selected from IL; new icons can be created and added to IL anytime. The terms in an icon are standardized by [V]. The whole CTSS or WTS is realized from the given CAI specification by the MI paradigm, and the physical GUI of the target WTS has the same appearance as the given CAI specification.

The most engineered CTSS part is the top layer or the CAI specification because once it has been verified the whole system can be generated automatically. Working together, the three layers comprise a customized CTSS that effectively realizes the philosophical arguments of Gruber [14] in an integrated fashion. Gruber's ontology is a consensus-certified conceptualization, which is understandable to humans and meanwhile machine process-able. Guarino deepened this ontology concept by arguing that it should have a subsumption hierarchy of sub-ontologies with axiomatic associations to constrain interpretation [11].

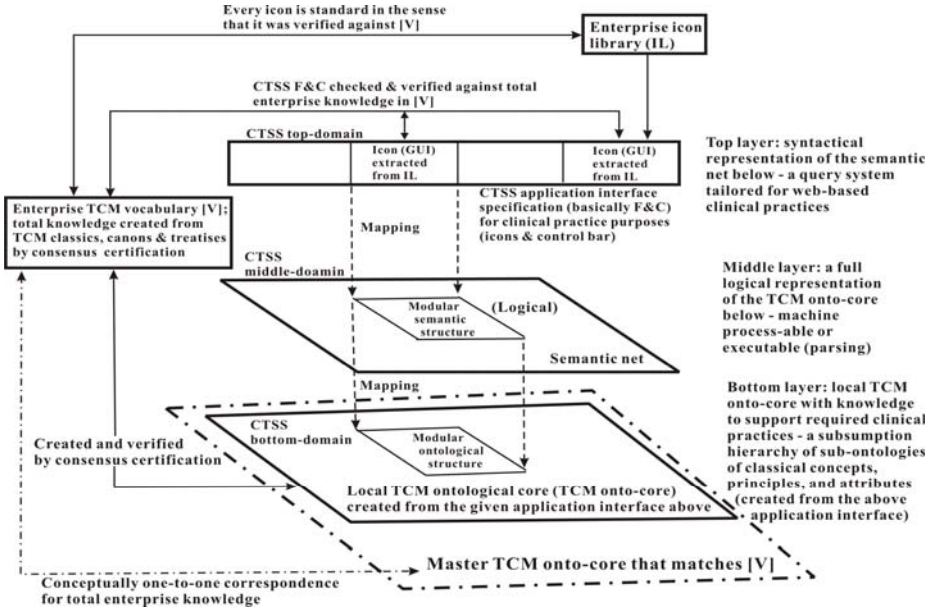

Fig. 5. The three-layer architecture of CTSS

4 Meta-Interface Paradigm – Proposed Innovative Software Development Approach

The meta-interface (MI) paradigm combines the THWADI and CAP philosophies to automate CTSS generation with the software-engineered CTSS application interface specification as the only required input; that is the *customized application interface* (CAI) specification. The physical CTSS has three layers (Figure 5). Its *graphical user interface* (GUI) has the same characteristics as the original CAI specification. Key elements in the MI paradigm include:

a) ***Enterprise TCM vocabulary:*** All CTSS terms are verified against it (i.e. [V] in Figure 5).

b) ***Unique icon library (IL):*** This contains all the graphic icons that Nong's accumulated over time. Any new icons created for customers will be added to IL as evolution. An icon in the context of the MI paradigm is a modular semantic structure backed up by its modular ontological structure (Figure 5). For machine processing, every icon is supported by a group of "control-oriented" and "data-oriented" object classes. An application interface to be customized is physically a collection of selected icons from IL that meet specific clinical functions of stated constraints. Icon creation is a formal process, for its terminology is checked and verified against the standard enterprise vocabulary [V]. This disambiguates communications within the Nong's enterprise, between Nong's and the global TCM community, and among the Nong's customers (e.g. customized WTS).

c) ***Customized application interface (CAI):*** In its business plan Nong's would customize the MC based telemedicine software and remotely install it for the

client [13]. The customization process is basically fast prototyping, and the clients need only to customize the CAI specification correctly together with Nong's. With the final CAI specification the generation of the customized WTS artifact and its remote installation (client's site) are automated. Verification and validation of the final WTS can be conducted anytime and anywhere by using the *semantic TCM visualizer* (STV) – a mandatory element in the MI paradigm. In our research the customized CAI specification is the input to the automatic *meta-interface* (MI) process.

d) ***Annotated master/enterprise TCM onto-core blueprint:*** It is the huge piece of annotated code (or blueprint) for the subsumption hierarchy of the entire enterprise TCM onto-core to match the formal knowledge in the enterprise vocabulary [V]. The blueprint creation is semi-automatic to quicken rectification of errors by the group of TCM domain experts who perform consensus-certification. This semi-automatic process has two phases:
   i. ***Manual phase:*** The DOM (document object model) tree for the master TCM onto-core has to be drawn manually. The drawing helps experts visualize and verify the necessary facts quickly against the canonical information in [V]. In fact, there are usable commercial tools in the field that can be support such drawing; the DBDesigner (DBD) by Microsoft is an example.
   ii. ***Automatic phase:*** Firstly, the annotated blueprint is automatically generated from a drawn DOM tree. Annotation can be achieved by different metadata systems. For example XML, RDF, and OWL metadata systems are popular because the codes generated for them are interoperable [1]. In fact, the DBD system can generate the corresponding XML-annotated codes from its own drawings. Secondly, the GUI (graphical user interface) subsystem is automatically generated for the final WTS system for human interaction.

e) ***Automatic CTSS/WTS database generation:*** A physical CTSS/WTS is generated from the given CAI specification that indicates what portion of the enterprise TCM onto-core blueprint to be extracted automatically by the MI paradigm. The extraction, in the form of a piece of annotated code (blueprint), is then automatically instantiated into the respective local TCM onto-core.

f) ***Appropriate programming language(s) for the logical object classes:*** The executable forms of those functions in an icon in the IL are object classes. In the MI paradigm functions in an icon are instantiated as object classes selected from the main enterprise object library; the MI paradigm is object-based.

g) ***Semantic TCM visualizer (STV):*** This converts an XML-annotated code into the matching DOM tree and traces the parsing mechanism on line. In this way it verifies and validates any part of the physical CTSS anytime and anywhere.

h) ***Remote CTSS installation:*** The CTSS package contains: the GUI for human interaction; wireless communication capability for the MC; the CTSS database; object classes; and other auxiliary software tasks. It is sent via the web to remote sites for installation.

5 Experimental Results

Many experiments were carried out in the Nong's WTS (mobile clinics (MC) based) environment over the mobile Internet. The results verified that the novel MI paradigm proposed by this paper is indeed effective in customizing usable WTS. The set of results presented here include: i) the CAI specification customized for the physicians' diagnosis/prescription (D/P) procedure to treat patients; ii) the actual D/P GUI generated from a CAI specification by the MI paradigm; and iii) a partial DOM tree and its corresponding XML-annotated code or blueprint as visualized by using STV. The Chinese TCM terms in the results were translated into English by using the World Health Organization (WHO) standard [15].

Table 1. Traditional 4-step TCM diagnostic procedure and result examples

look ("望")	listen & smell ("聞")	question ("問")	pulse-diagnosis ("切")
e.g. pale face	e.g. cough, bad breath	e.g. headache? fever? loathe cold ambience? ("惡寒/怕冷")	e.g. taut and fast

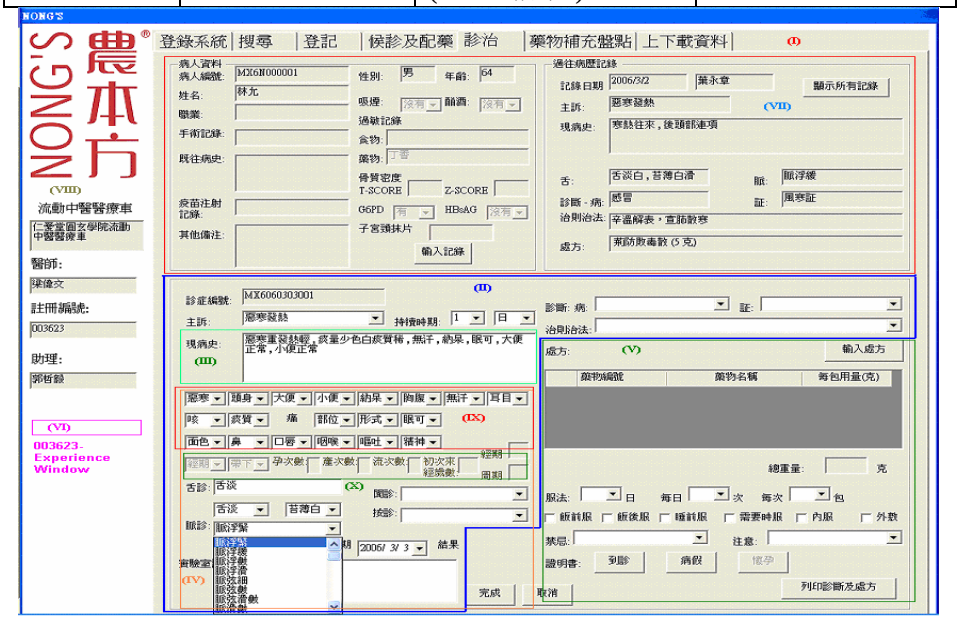

Fig. 6. The GUI generated from a CAI of chosen icons from the IL

5.1 The CAI Example

Figure 6 is a CAI specification (for generating the corresponding physical GUI) that includes: a) icon (I) – control bar; b) icon (II) – patient registration number (i.e. MX6060303001) and fields to be filled in the D/P process by the physician, including: patient's complaint ("主訴"), and diagnosis ("診斷") (e.g. illness/type ("病"/"証") and treatment principle ("治則治法")); c) icon (III) – symptoms ("現病史") obtained by a standard TCM diagnostic procedure; d) icon (IV) – pulse diagnosis ("脈診"); e)

icon (V) – prescription(s) (“處方”) to be dispensed with respect to the diagnosis; f) icon (VI) – experience window (record) entrance specific to the logon physician with medical practice registration (e.g. 003623); g) icon (IX) –diagnostic questions (e.g. Do you loathe cold ambience conditions (“惡寒/怕冷”)?) and general physical inspections (e.g. complexion (“面色”) – pale or red); h) icon (X) – tongue diagnosis (“舌診”) (e.g. texture and coating color). Table 1 shows four steps in the traditional TCM diagnostic procedure: look (“望”), listen & smell (“聞”), question (“問”), and pulse-diagnosis (“切”).

5.2 A Customized WTS Example

This example shows the operation of the physical WTS generated automatically from a given CAI (e.g. Figure 6) by the MI paradigm. The physical GUI for the operating WTS will appear the same as the parent/input CAI. In the GUI the set of symptoms, $S\{$怕冷重 (dislike cold ambience), 發熱輕 (light fever), 無汗 (no perspiration$\}$, obtained from the patient were keyed-in and echoed in the symptoms window “現病史”. Normally the parser will works automatically with the input query (e.g. $Q\{$怕冷重, 發熱輕, 無汗$\}$). The parsing process can also be visualized by pressing the Parse button that invokes the STV (as shown in Figure 7). The parsed result for the symptoms (in the “現病史” window) includes: a) Diagnosed (診斷) illness (病): Flu (感冒) & type (証) is “wind cold” (風寒); b) Treatment principle (治則): heating & sweating (辛温解表); and c) Prescription (處方): “荊防敗毒散”.

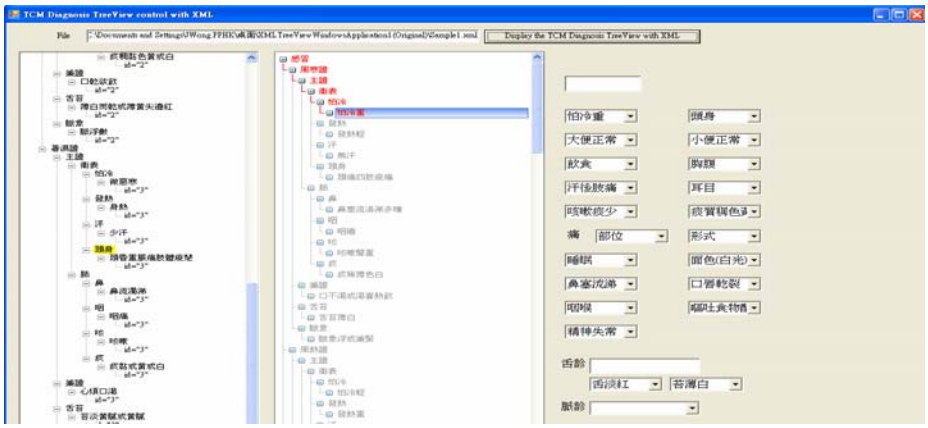

Fig. 7. Invoked STV to visualize a parsing operation

6 Conclusion

In this paper the meta-interface (MI) paradigm, which combines the THWADI and CAP philosophies, is proposed. It automates the generation of a client’s CTSS directly from the given *customized application interface* (CAI) specification. This

specification is constructed by “gluing” together icons selected from the enterprise icon library (IL) as part of a fast prototyping process. The automatic process extracts the portion of the subsumption hierarchy in the enterprise TCM onto-core that corresponds to the given CAI specification. The next step is to perfect the STV so that it visualizes and debugs more effectively.

Acknowledgement

The authors thank the Hong Kong Polytechnic University and the PuraPharm Group for funding the research, grants A-PA9H and ZW93.

References

1. R. Rifaieh and A. Benharkat, From Ontology Phobia to Contextual Ontology Use in Enterprise Information System, in Web Semantics & Ontology, ed. D. Taniar and J. Rahayu, Idea Group Inc., 2006
2. L.J. Osterweil, C. Ghezzi, J. Kramer and A.L. Wolf, Determining the Impact of Software Engineering Research on Practice, IEEE Comp., March 2008, 39-49
3. C. Chan, Ontological Methodologies, - From Open Standards Software Development to Open Standards Organizational Project Governance, Journal of Computer Science and Network Security, 7(3), March 2007
4. Allan K.Y. Wong, Wilfred W.K. Lin and Tharam S. Dillon, Local Compilation: A Novel Paradigm for Multilanguage-Based and Reliable Distributed Computing over the Internet, Special Issue: Mobile & Wireless Communications & Information Processing, Journal of Simulation, 75(1), July 2000, 18-31
5. A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis and E. Giannopoulou, Ontology Visualization Methods – A Survey, ACM Surveys, 39(4), October 2007
6. B. Boehm, Making a Difference in the Software Century, IEEE Computer, March 2008, 32-38
7. J.E. Bardram and H.B. Christensen, Pervasive Computing Support for Hospitals: An overview of the Activity-Based Computing Project, IEEE Pervasive Computing, 6(1), 44-51
8. Wilfred W.K. Lin, Allan K.Y. Wong and Tharam S. Dillon, Application of Soft Computing Techniques to Adaptive User Buffer Overflow Control on the Internet, IEEE Transactions on Systems, Man and Cybernetics, Part C, 36(3), May 2006, 397-410
9 M. Uschold, M. King, S. Moralee and Y. Zorgios, The Enterprise Entology, Artificial Intelligence Applications Institute, University of Edinburg, UK,
http://citesee.ist.psu.edu/cache/papers/cs/11430/ftp:zSzzSzftp.aiai.ed..ac.ukzSzpubzSzdocumentszSz1998zSz98-kerent-ontology.pdf/uschold95enterprise.pdf
10. UMLS, http://www.nlm.nih.gov/research/umls/
11. D. Taniar and J.W. Rahayu, Web Semantics & Ontology, Idea Group Publishing, 2006
12. J.F. Kaar, International Legal Issues Confronting Telehealth Care, Telemedicine Journal, March 1999
13. Jackei H.K. Wong, Allan K.Y. Wong, Wilfred W.K. Lin and Tharam S. Dillon, Dynamic Buffer Tuning: An Ambience-Intelligent Way for Digital Ecosystem, Proc. of the 2nd IEEE International Conference on Digital Ecosystems and Technologies (IEEE-DEST 2008), February, 2008, Phitsanulok, Thailand
14. T. R. Gruber, A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), 1993, 199-220
15. WHO International Standard terminologies on traditional medicine in the Western Pacific Region, ISBN 978 92 9061 248 7, World Health Organization, 2007