Technical report: A preliminary hazard analysis for the GIP number entry software

Paolo Masci

School of Electronic Engineering and Computer Science
Queen Mary University of London
London, United Kingdom
`p.m.masci@qmul.ac.uk`

Abstract

The results of a preliminary hazard analysis are presented that identify common design errors in infusion pump software that may potentially cause use hazards. Many identified problems apply to other types of interactive medical devices, including ventilators and radiotheraphy machines. The identified issues may be used as a basis to define safety requirements that, if satisfied by user interface software, can substantially improve the quality and safety of broad classes of medical devices.

1 Hazard analysis

A Preliminary Hazard Analysis (PHA) based on the GIP-UI architecture [2] is performed. PHA is a top-down analysis technique: starting from postulated undesired system outcomes, the analyst works out the causes of the postulated outcomes in the system design.

Here, known use hazards and related primary causes reported in incidents and accidents involving infusion pumps are used to define the undesired system outcomes. The GIP-UI architecture is used as a reference to reason about root causes of the considered use hazards in user interface design. That is, causal factors are identified by making hypotheses about possible design errors or failures that may occur within GIP-UI components, or in information flows between GIP-UI components.

1.1 Scope of the analysis

The focus of the PHA is the *number entry software*, which is responsible for managing interaction with the user when infusion parameters or pump setting need to be configured.

Number entry software is safety-critical in the sense that design errors in this module may cause use errors (e.g., mis-programming of the pump) that can lead to potential harm to the patient (typically, overinfusion or underinfusion). The number entry module has been chosen because the analysis of incident reports suggest that overinfusion and underinfusion due to software defects and user interface issues are a major problem with infusion pumps.

Number entry software is designed to support a *number entry task*, which identifies the sequence of actions carried out by the user when entering infusion parameters or pump setting. In the current generation of infusion pumps, the typical number entry task is carried out through the following three main actions:

1. An infusion parameter or pump setting is selected by the user
2. The selected item is edited by the user
3. The value is submitted by the user

The actions described above generally involve pressing buttons and keys on the pump user interface. Whenever the pump registers a button press or a key press, the number entry

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

software performs a computation. The computation may modify the device state (e.g., a new infusion rate may be configured in the pump), and generate feedback on the user interface to present the new device state to the user.

Buttons and keys currently used for number entry can be described using two broad classes of widgets: *serial number entry widgets*, which allow the user to enter the digits of the values serially, from the most significant to the least significant digit; *incremental number entry widgets*, which allow the user to modify an initial value by incrementing or decrementing it. Serial interfaces require a full numeric keypad, and incremental interfaces typically have two or four arrow keys, depending on the exact style of interaction. A detailed description of these number entry widgets is in [3]. Notable for hazard analysis is how errors can be corrected. In a serial interface, either numbers have to be re-entered or there must be a delete key. In contrast, in incremental interfaces, as it were, the whole point of the user interface is to adjust numbers, so correcting errors does not require a separate delete key, as correction is just a special case of adjustment.

1.2 Sources of information

The analysis is informed by the following sources of information:

- Domain knowledge developed within the CHI+MED project (www.chi-med.ac.uk). This knowledge results from the analysis of commercial infusion pumps [7, 8, 12, 15, 16, 17, 19], infusion pump logs [11], incidents involving infusion pumps [9, 13, 14, 21], current and best clinical practice in hospitals and home care [20], and workshops with pump manufacturers, users and clinicians [4, 5];
- Incident reports collected through the FDA's Manufacturer and User Facility Device Experience (MAUDE) database [23];
- Guidance documents and recommendations on infusion pump design [18, 22];
- Previous hazard analysis on other components of the GIP [1, 24];
- International standards ANSI/AAMI HE75:2009 on human factors, and ISO 14971:2007 on risk management.

1.3 Terminology

According to the definitions given in ISO 14971:2007, a *hazard* is a potential source of harm, where *harm* is defined as physical injury or damage to the health of people or the environment. Use-related hazards are the focus of this work. They are caused by use error (including a mismatch between the user's actions and the design), that is, an error occurring during the use of a system.

As pointed out in [10], the definition of hazard given in the ISO standard is ambiguous. This is due to the fact that virtually any event within a system could be considered a potential source of harm under appropriate conditions. For example, consider the following situation: an infusion pump user interface silently ignores keystrokes from the user, and this results in mis-programming of the pump, ultimately leading to over-infusion given to the patient, and hence patient injury. Which event in the causal chain is the potential source of harm (i.e., the hazard)? Which event is the cause of the identified hazard? Different interpretations lead to different conclusions. It has been argued that the ambiguity can be mitigated using slightly refined definitions of hazard and cause of hazard [10]:

- a *hazard* is defined as any means, mode or manner by which a system might cause harm.
- a *cause of hazard* is defined as a set of circumstances (a scenario) that might reasonably lead to a hazard.

In the present paper, the terms hazard and cause of hazard are used with the above meanings. A distinction is also made between *primary cause*, which is an event linked to the hazard through a direct causal relation, and *root cause*, which is an event at the beginning of the causal chain. Obviously any search for a root cause must have stopping criteria — at some point, causes become irrelevant or too costly to fix. For example, an overdose of a drug might be fatal and one root cause is the use error leading to the infusion pump delivering the overdose. But before that was the consultant prescribing the wrong drug, or writing down the dose 15 without the intended decimal point, 1.5. Neither of these possible causes are reasonably considered part of the design hazards of the infusion pump.

By applying these definitions (hazard, cause of hazard) to the example illustrated above, overinfusion is the hazard, mis-programming of the pump is a primary cause of the hazard, and the user interface software that silently ignores keystrokes is a root cause of the hazard in software design.

1.4 Results

The analysis identified 53 potential design errors in number entry software that may cause use hazards. These design errors are summarized into a hazard analysis table that presents the identified cause-effect relationships between use hazards, primary causes due to use errors, and root causes due to design errors in infusion pump software. Each design error is exemplified using input key sequences that can be used to test real infusion pumps.

1.4.1 Hazard analysis table

	Causes of overinfusion and underinfusion hazards		
	Primary cause	**Potential root cause**	**Example**
1	The user fails to enter the intended infusion parameter value.	The device unexpectedly resets the display value without the user's awareness.	The display value is erroneously reset to 0 when the device registers the first key press for number entry. For example, if the display is 9 when the user starts number entry, the input sequence $\boxed{\bullet}\boxed{2}$ results in 0.2, instead of 9.2, without any warning or notification.
2	The user fails to enter the intended infusion parameter value.	The device erroneously discards key presses without the user's awareness.	If the minimum legal value for the entered parameter is 0.1, the input sequence $\boxed{0}\boxed{\bullet}\boxed{0}\boxed{1}$ is erroneously registered as 0.1 without any warning or notification.
3	The user fails to enter the intended infusion parameter value.	The device erroneously delays feedback without the user's awareness.	The display is updated after a delay because the software routines are still processing the last keyed command.
4	The user fails to enter the intended infusion parameter value.	The device unexpectedly registers or discards the entered value before the user finishes entering it without the user's awareness.	The entered value is erroneously registered or discarded when the user exceeds x seconds to enter the value without any warning or notification.

4 A preliminary hazard analysis for the GIP number entry software

5	The user fails to enter the intended infusion parameter value.	The device automatically changes the out-of-range value entered by the user without the user's awareness.	If the maximum value for the input field is 10, the input sequence $\boxed{5}\boxed{0}$ is changes into 10 without any warning or notification.
6	The user fails to enter the intended infusion parameter value.	The device uses default values (.e.g., provided by DERS) without the user's awareness.	The default value suggested by DERS is used for drug concentration without any warning or notification
7	The user fails to enter the intended infusion parameter value.	The device fails to distinguish between consecutive input actions (e.g., double clicks) and single input actions made by the user.	The key sequence $\boxed{2}\boxed{2}\boxed{\bullet}\boxed{3}$ is erroneously registered as 2.3 or 0.3 without any warning or notification when the key sequence $\boxed{2}\boxed{2}$ is performed too quickly.
8	The user fails to enter the intended infusion parameter value.	The device erroneously accepts key combinations without the user's awareness.	Pressing key $\boxed{2}$ and $\boxed{3}$ simultaneously erroneously results in either 23 or 32 without any warning or notification.
9	The user fails to enter the intended infusion parameter value.	The device erroneously ignores key combinations without the user's awareness.	If the device has multiple up/down keys for editing the integer and the fractional part of the parameter value (e.g., ▲for the integer part, and △for the fractional part), simultaneous key presses of ▲and △are erroneously ignored without any warning or notification.
10	The user fails to enter the intended infusion parameter value.	Virtual keys rendered on touchscreen displays are too small for the user to press or read correctly.	Virtual keys are smaller than 23mm (which is the minimum size recommended by HF75:2009)
11	The user fails to enter the intended infusion parameter value.	The device fails to correctly update the value of a field (e.g., because of unsound computation or logic) based on the values of other relevant fields.	The device requires to enter volume to be infused and duration of infusion; the rate value is computed using arithmetic division with single precision floating point, which may lead to unsound results.
12	The user fails to read the infusion infusion parameter value.	The device uses inappropriate fonts or formats to render values or units.	The device renders fractional values without a leading zero (e.g., .9 instead of 0.9), or integer values with leading zeros (e.g., 09 instead of 9)
13	The user fails to enter the intended infusion parameter value.	The number entry software allows numeric wrap-around[1] with single key presses.	Devices with a cursored display enable moving the cursor from the least significant digit to the most significant digit with a single key press.

[1] Numeric wrap-around refers to the way ▲and ▼behave at boundary conditions for maximum and minimum values: ▲changes the maximum value into the minimum; ▼changes the minimum value into the maximum. For cursored displays, numeric wrap-around affects also the way the cursor position is

14	The user fails to enter the intended infusion parameter value.	The device is unable to verify or omits to check whether the hardware display is defective, incorrectly mounted, or incorrectly configured.	Display elements are erroneously mounted upside down without the user awareness.
15	The user fails to enter the intended infusion parameter value.	Infusion parameters values or units required by the pump are inconsistent with those provided in the prescription order used to program the pump.	The pump requires infusion rates in mL per hour, but the prescription order reports rates in mg per hour.
16	The user fails to enter the intended infusion parameter value.	The device positions labels and values of infusion parameters on the display inappropriately.	The label for volume to be infused is abbreviated as "VTBI" and shown next to the volume value — the last letter of the acronym can be misread as a 1. For instance, the text "VTBI 9 mL" can be misread as "VTBI 19 mL".
17	The user fails to enter the intended infusion parameter units.	The device displays inappropriate abbreviations for units.	The device abbreviates *micrograms* into "mg" (which can be easily mistaken as *milligrams*)
18	The user fails to enter the intended infusion parameter units.	The device does not display the units during number entry.	A rate value is erroneously rendered as 9 without specifying the units.
19	The user fails to enter the intended infusion parameter units.	The device uses inappropriate font type and/or size.	The device uses *serif* fonts, or font size less than 12pt.
20	The user fails to select the correct fluid, drug name, or patient profile.	The device displays similar-looking names for different information elements, e.g., fluid names, drug names, or patient profiles.	The device shows, e.g., two similar-looking abbreviations "UAC" and "UVC" for labeling two different fluids "Umbilical Artery Catheter" and "Umbilical Venous Catheter".
21	The user fails to resolve the error condition.	The device uses uninformative or incorrect notifications.	The device shows a generic message "Out of range" when the user enters a value that is either too high or too low, but omits to show the actual value registered by the device. Another example is: the device beeps and clears the display during number entry when the key pressed by the user results in a value that is either too high or too low.

changed: ◀changes the cursor position from the left-most position to the right-most position; the ▶key changes the cursor position from the right-most position to the left-most position.

22	The user fails to resolve the error condition.	The device unexpectedly halts number entry without providing feedback about the detected error.	The device halts number entry when the key sequence registered by the device results into a number that exceeds the legal limits of the infusion parameter, but does not provide informative feedback about the detected error.
23	The user fails to enter the intended infusion parameter, or fails to resolve an error condition.	The device erroneously shows incomplete information without the user's awareness.	The device automatically scrolls or trims text messages that overflow the display capabilities.
24	The user commits a decimal point error.	The device ignores decimal point key presses without the user's awareness.	The key sequence $\boxed{1}\boxed{0}\boxed{0}\boxed{\bullet}\boxed{1}$ is erroneously registered as 1001 without any warning or notification if the infusion parameter value must be an integer.
25	The user commits a decimal point error.	The device erroneously displays ill-formed integer values.	Integer values are rendered with leading zeros (e.g., 09, which can be easily misread as 0.9).
26	The user commits a decimal point error.	The device erroneously displays ill-formed fractional values.	Fractional values are rendered with "naked" decimal point (e.g., .9, which can be easily misread as 9).
27	The user commits a decimal point error.	The device erroneously accepts ill-formed integer values without the user's awareness.	The key sequence $\boxed{0}\boxed{9}$ is erroneously accepted and registered as 9 without any warning or notification.
28	The user commits a decimal point error.	The device erroneously accepts ill-formed fractional values without the user's awareness.	The key sequence $\boxed{\bullet}\boxed{9}$ is erroneously accepted and registered as 0.9 without any warning or notification.
29	The user commits a decimal point error.	The device fails to display numeric values distinguishing the integral and fractional parts.	Fractional digits are rendered using the same font size and color used for integer digits.
30	The user commits a decimal point error.	The device does not enhance the visibility of the decimal point.	The decimal point is rendered using a small symbol . instead of a more visible ● symbol.
31	The user commits a decimal point error.	The device erroneously renders integer digits as if they were fractional digits.	The integer value 1001 is erroneously rendered as $100{\scriptstyle 1}$.
32	The user commits a decimal point error.	The device always displays a decimal point even if the decimal point has not been registered.	The device renders 0. as initial value, but the decimal point has not been registered.
33	The user commits a decimal point error.	The device uses inappropriate layouts for number entry keys.	The number entry layout rendered on the touchscreen places the decimal point key next to numeric keys.

34	The user commits a decimal point error.	The device erroneously accepts ill-formed values without the user's awareness.	The illegal input sequence $\boxed{9}\boxed{.}\boxed{0}\boxed{.}\boxed{9}$ is erroneously accepted without any warning or notification, and registered as 9.09
35	The user commits a decimal point error.	The device erroneously clears the fractional part of the value when a decimal point key press is registered.	The illegal input sequence $\boxed{9}\boxed{.}\boxed{0}\boxed{.}\boxed{9}$ is erroneously accepted and registered as 9.9 without any warning or notification.
36	The user fails to confirm the entered value.	The device requires a final confirmation but feedback suggests value accepted.	The entered value is confirmed with $\boxed{\text{OK}}$, but the confirmed value is not applied until $\boxed{\text{Confirm}}$ is pressed.
37	The user fails to select the intended infusion parameter.	The input fields for one or more infusion parameters are not shown in the visible screen area without the user's awareness.	The input field for rate falls out of the visible screen area, but nothing suggests that the screen area can be scrolled.
38	The user fails to select the intended infusion parameter.	The device uses inappropriate selection symbols or colors.	The device uses a "cross-out" symbol (X) to indicate selected items, instead of a "check" symbol ($\surd$).
39	The user fails to select the intended infusion parameter.	The device aligns informative text inappropriately, so that it can be read as labels of soft buttons.	The device displays the programmed rate value (e.g., 5 mL per hour) as a soft button label, but the soft button cannot be used to select the input field for rate.
40	The user fails to enter the intended infusion parameter value.	The device displays multiple information inconsistently.	The device displays information about the current action (e.g., "Enter Rate") and instructions for the next action (e.g., "Select volume to be infused") at the same time.
41	The user fails to select the intended infusion parameter.	Infusion parameters required by the pump are inconsistent with those provided in the prescription order used to program the pump.	The pump requires to enter infusion rate and then volume to be infused, but the prescription order reports volume first and then rate.
42	The user fails to modify the entered value.	The device is not responsive to user actions.	The device incorrectly locks the keys without the user's awareness. Another example is: the device software "freezes" because of infinite loops caused by program errors.
43	The user fails to modify the entered value.	The same soft key is rendered in different positions in different device modes.	The soft key "OK" is either the first soft key on the right or the first on the left depending on the device mode.
44	The user fails to modify the entered value.	The same soft key is displayed with different labels in different device modes.	The soft key used to cancel the entered value is either "Cancel" or "Back" depending on the device mode.

45	The user fails to enter the intended value.	Legal gestures on touchscreens are erroneously discarded without the user's awareness.	The device erroneously ignores press and hold gestures on virtual buttons.
46	The user fails to confirm the entered value.	Illegal gestures on touchscreen are erroneously registered without the user's awareness.	The device erroneously de-selects input field areas if the user taps outside any input area.
47	The user fails to confirm the entered value.	Similar gestures on touchscreen are erroneously associated with logically different functions.	The device associates "slide left" to *cancel*, and "slide right" to *confirm*.
48	The user fails to enter the intended value.	The device uses different fractional precision for different infusion parameters without the user's awareness.	The device accepts rate values with one fractional digit, and volume values with two fractional digits.
49	The user fails to enter the intended value.	For the same infusion parameter, the device accepts inconsistent precision or format for different value ranges without the user's awareness.	The device accepts rate values with fractional digits only if the value is less than 100 .
50	The user fails to enter the intended value.	The new device firmware changes the way the device handles input and output.	For devices with ▲and △keys, the increment step is changed without the user's awareness (e.g., without updating user manuals).
51	The user fails to predict the effect of key presses.	The device overloads number entry keys without the user's awareness.	Keys ▲and ▼are overloaded with MR (Memory Recall).
52	The user fails to select the intended units for infusion parameters.	The device automatically selects the wrong units without the user's awareness.	The device automatically selects the factory default units and omits to display the units during data entry.
53	The user fails to predict the effect of key presses.	The device unexpectedly changes the behavior of repeating keys.	The device changes the increment step of keys when they are pressed and held down for more than x seconds.

Table 1 PHA hazard table.

2 Discussion and conclusions

The analysis identified a substantial set of root causes of use hazards in software design. However, the hazard analysis results are not exhaustive. To improve coverage, best practice is to employ several hazard analysis techniques together in a complementary way, as no single technique guarantees the identification of all potential hazards and related causes in complex systems [10]. PHA is usually the first hazard analysis technique applied to a system, and the results of the analysis constitute an initial inventory that can be used to inform other hazard analysis techniques, such as Failure Mode and Effect Analysis (FMEA) [6]. Extending the PHA with other hazard analysis techniques is, however, beyond the scope of the present work.

The hazards analysis presented in this work is general in the sense that the problematic functionalities are common in broad classes of infusion pumps. Manufacturers should therefore be able to map the functionalities of their specific make and models to the GIP-UI, and establish a mapping between the identified design issues and implemented software routines. Under this perspective, the GIP-UI architecture and the hazard analysis can be used as a reference to establish whether a software implementation meets basic levels of quality and safety. Manufacturers who reference this generic analysis in their design process may benefit from checking their results against this independent work. However, it is worth noting that manufacturers claiming to use this preliminary hazard analysis in their design process still have to establish sufficient evidence to the FDA that their device is safe and effective.

Acknowledgement

This work is support by CHI+MED (Computer-Human Interaction for Medical Devices, EPSRC research grant [EP/G059063/1]). We thank Paul Curzon (Queen Mary University of London), Michael Harrison (Newcastle University), Harold Thimbleby (Swansea University) and Chris Vincent (University College London) for their valuable comments.

References

1 The Generic Patient Controlled Analgesia Pump Hazard Analysis. `http://rtg.cis.upenn.edu/gip-docs/Hazard_Analysis_GPCA.doc`.

2 A generic user interface architecture for analyzing use hazards in infusion pumps. 2014. under review.

3 A. Cauchi, P. Curzon, P. Eslambolchilar, A. Gimblett, H. Huang, P. Lee, Y. Li, P. Masci, P. Oladimeji, R. Rukšėnas, and H. Thimbleby. Towards dependable number entry for medical devices. In *EICS4Med, 1st International Workshop on Engineering Interactive Computing Systems for Medicine and Health Care*. ACM Digital Library, 2011.

4 CHI+MED. Guidelines for number entry interface design (infusion devices). `http://www.chi-med.ac.uk/researchers/bibdetail.php?docID=684`, 2013.

5 CHI+MED. Personas and scenarios (infusion devices). `http://www.chi-med.ac.uk/researchers/bibdetail.php?docID=685`, 2013.

6 P.L. Goddard. Software fmea techniques. In *Reliability and Maintainability Symposium, 2000. Proceedings. Annual*, pages 118–123. IEEE, 2000.

7 M.D. Harrison, J. Campos, and P. Masci. Reusing models and properties in the analysis of similar interactive devices. *Innovations in Systems and Software Engineering, Springer-Verlag London*, 2013.

8 M.D. Harrison, P. Masci, J.C. Campos, and P. Curzon. Automated theorem proving for the systematic analysis of interactive systems. 2013.

9 ISMP Canada. Fluorouracil incident root cause analysis report. `http://www.ismp-canada.org/download/reports/FluorouracilIncidentMay2007.pdf`.

10 P. Jones, J. Jorgens III, A.R. Taylor Jr., and M. Weber. Risk management in the design of medical device software systems. *Journal of Biomedical Instrumentation and Technology*, 36(4):237–266, 2002.

11 P. Lee, F. Thompson, and H. Thimbleby. Analysis of infusion pump error logs and their significance for health care. *British Journal of Nursing*, 21(8):S12–S22, 2012.

12 P. Masci, A. Ayoub, P. Curzon, M.D. Harrison, I. Lee, and H. Thimbleby. Verification of interactive software for medical devices: Pca infusion pumps and fda regulation as an example. In *EICS2013, 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM Digital Library, 2013.

13 P. Masci and P. Curzon. Checking user-centred design principles in distributed cognition models: a case study in the healthcare domain. In *Proceedings of the 7th conference on Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society: information Quality in e-Health*, USAB'11, pages 95–108, Berlin, Heidelberg, 2011. Springer-Verlag.

14 P. Masci, H. Huang, P. Curzon, and M.D. Harrison. Using pvs to investigate incidents through the lens of distributed cognition. In Alwyn E. Goodloe and Suzette Person, editors, *Proceedings of the 4th NASA Formal Methods Symposium (NFM 2012)*, volume 7226, pages 273–278, Berlin, Heidelberg, April 2012. Springer-Verlag.

15 P. Masci, R. Rukšėnas, P. Oladimeji, A. Cauchi, A. Gimblett, Y. Li, P. Curzon, and H. Thimbleby. On formalising interactive number entry on infusion pumps. *ECEASST*, 45, 2011.

16 P. Masci, R. Rukšėnas, P. Oladimeji, A. Cauchi, A. Gimblett, Y. Li, P. Curzon, and H. Thimbleby. The benefits of formalising design guidelines: A case study on the predictability of drug infusion pumps. *Innovations in Systems and Software Engineering, Springer-Verlag London*, 2013.

17 P. Masci, Y. Zhang, P. Jones, P. Curzon, and H. Thimbleby. Formal verification of medical device user interfaces using PVS. In *ETAPS/FASE2014, 17th International Conference on Fundamental Approaches to Software Engineering*. Springer Berlin Heidelberg, 2014.

18 National Patient Safety Agency (NHS/NPSA). Design for patient safety: A guide to the design of electronic infusion devices. `http://www.nrls.npsa.nhs.uk/resources/?EntryId45=68534`, March 2010.

19 P. Oladimeji, H. Thimbleby, and A. Cox. Number entry interfaces and their effects on error detection. In *Human-Computer Interaction – INTERACT 2011*, volume 6949 of *Lecture Notes in Computer Science*, pages 178–185. Springer Berlin Heidelberg, 2011.

20 R. Rukšėnas, P. Curzon, A. Blandford, and J. Back. Combining human error verification and timing analysis: a case study on an infusion pump. *Formal Aspects of Computing*, pages 1–44, 2013.

21 H. Thimbleby. Is it a dangerous prescription? *BCS Interfaces*, 84, 2010.

22 US Food and Drug Administration, Center for Devices and Radiological Health (FDA/CDRH). An Introduction to Human Factors in Medical Devices. `http://www.fda.gov/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm259748.htm`.

23 US Food and Drug Administration (FDA). Manufacturer and User Facility Device Experience Database (MAUDE). `http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/PostmarketRequirements/ReportingAdverseEvents/ucm127891.htm`.

24 Y. Zhang, P. Jones, and R. Jetley. A hazard analysis for a generic insulin infusion pump. *Journal of diabetes science and technology*, 4(2):263, 2010.