Journal of Behavioral Informatics
Digital Humanities & Development Research
www.behaviouralinformaticsjournal.info

Vol 6, No 1, March, 2020

HELT D: A Context-aware HEalthy Living Through Diet System

[1]Atiku, A.U. & [2]Sajoh, D.I.
[1,2]Department of Computer Science
Modibbo Adama University of Technology
Yola, Adamawa State, Nigeria
[1]ahmed.atiku@mautech.edu.ng, [2]disajoh@mautech.edu.ng

ABSTRACT

Naturally, diet is one of the most important factors in human life. In this work, a novel model that uses technology to support healthy living through diet is proposed. HELT D: A Context-aware HEalthy Living Through Diet System, is a system that provides users with diet based on their condition of health and dietary needs. A high level description of the proposed system was provided and formal specification of the system was given. For the formal specification, CCA (Calculus for Context-aware Ambients) was used. The specification was implemented using ccaPL (CCA Programming Language) to execute and study the system's behaviour. The system demonstrates both context-awareness and concurrency.

Keywords— Context-aware, Healthy living, Diet, Pervasive Systems, cca

Journal Reference Format:
Atiku, A.U. & Sajoh, D.I. (2020): HELT D: A Context-aware HEalthy Living Through Diet System. Behavioural Informatics, Digital Humanities & Development Journal Vol 6. No. 1. Pp 103-118. Available online at behaviouralinformaticsjournal.info; https://www.isteams.net/behavioralinformaticsjournal

1. INTRODUCTION

Human beings have to make various decisions in their daily lives. One of the key decisions is what to eat. The food we eat serves as source of energy for carrying out activities as well as maintaining our system. This decision will appear to be trivial, considering the abundance of food and that eating is a mastered art. However, if we consider other factors such as disease, weight management, health related risks and lifestyle, it stops being trivial.

Diabetes is a condition in which the body *1)* fails to produce *insulin* [25] or *2)* fails to produce enough insulin or fails to utilize the insulin it produced [25, 28]. Insulin is a hormone that regulates amount of glucose in the body [28]. Diabetics should be careful in choosing their food because there is universal relationship between blood glucose and food types [34] and amount of carbohydrate taken is inversely proportional to level of glycaemic control [33]. The goal is to try to obtain an ideal glucose level. An ideal amount is in the range of 3.5 - 5.5 mmol/l before meal and 8 mmol/l two hours after meal [10]; see Figure 1. A study shows restriction of carbohydrate intake to about 30g a day can result in outstanding glycaemic control [33].

Target Levels by Type	Before meals (pre prandial)	2 hours after (post prandial)
Non-diabetic	4.0 to 5.9 mmol/L	under 7.8 mmol/L*
Type 2 diabetes	4 to 7 mmol/L	under 8.5 mmol/L
Type 1 diabetes	4 to 7 mmol/L	under 9 mmol/L
Children w/diabetes	4 to 8 mmol/L	under 10 mmol/L

Figure 1: Blood sugar level ranges (The Global Diabetes Community 2012)

Classification	BMI (kg/m²)
Underweight	<18.5
Normal range	18.5–24.9
Overweight	25.0–29.9
Obese	**>30.0**
Class I	30.0–34.9
Class II	35.0–39.9
Class III severe (or 'morbid obesity' or 'super obesity')	>40.0

Figure 2: BMI (House of Commons Health Committee et al 2004)

Blood Pressure (BP) is the force which the blood exerts against artery walls, when the heart beats (*systolic pressure*) and when it relaxes (*diastolic pressure*) [18]. If the pressure rises and stayed over extended period, a person is said to have high blood pressure or hypertension [18]. This pressure forces the heart to work harder and can potentially harm the heart, kidneys, brain and eyes [18]. It is identified as one of the leading causes of heart attack, stroke and kidney disease [6]. Dietary decision is linked with chances of developing high blood pressure. For individuals with hypertension, cutting down the amount of salt consumed is essential [27, 36, 18].

Obesity is condition whereby amount of fat in the body is so high that health may be seriously affected [20]. Individuals with *body mass index* (BMI) of 30 and above are classified as being obese, see figure 2. Obesity is a worldwide epidemic [32, 20, 4]. It is emerging as the most significant contributor to ill health [20], is linked to various diseases [17] and it causes premature death [24, 19]. Obesity simply occurs when individuals consume more than their energy needs [17]. Historically, people have perception that being fat is healthy [12] and a symbol of wealth and fertility [4]. Some health personnel also believe that weight is a sign of a good living [32] even though many health personnel are finding it difficult to discharge their duty as a result of being obese [32]. Obesity is described as self-inflicted condition [24], a slow process which is fuelled by environmental factors [1].

There is scientific evidence that diet have strong impact on health during an individual's lifetime [44].It may determine if a person can contract some diseases in the future such as diabetes, cancer and cardiovascular disease [29, 44]. A strong relationship exists among the components discussed above. There is need for program that target obesity [14] and dietary habits. It is an "obvious and simple starting point in tackling obesity" [17]. Obesity is treated by controlling food intake [31, 8]. Diet can lower blood pressure for both hypertensive and non-hypertensive individuals [27, 6]. Obesity increases risk of diabetes [3, 39, 44]. Presence of diabetes increases the risk of hypertension and vice versa [15]. Eating diet which has low fat, lots of protein and carbohydrates that has low glycaemic index prevent weight gain in normal weight individuals, cause weight loss in overweight and have positive effect on risk factors associated with diabetes [3].

In this work, a novel design for a model of a system that provides diet to individuals based on their condition of health and current needs of their body is presented. The system should provide a range of healthy diets which will allow the user to have multiple safe choices.

2. DIET MONITORING AND CONTROL SYSTEMS

The types of diet monitoring and control systems were identified and categorized into three: **Passive Diet Monitoring and Control System**, **Semi-Active Diet Monitoring and Control System** and **Active Diet Monitoring and Control System**. A fourth type is **Super-Active Diet Monitoring and Control System**, which combines the features of the second and third systems already mentioned.

The criteria used is based on the extent of the system's involvement in managing diet.

2.1 Passive Diet Monitoring and Control System

Passive Diet Monitoring and Control Systems are those systems whereby the major activities of diet monitoring and control are undertaking by an individual using them. An outline or specification will be given and the person will follow the specification in order to have both healthy diet and lifestyle. The system could be as easy as a user adopting some known healthy ways without using any tool for measuring, storing or carrying out analysis about diet, and other physical activities. It could also include some devices for storing and processing information about user's diet and activities. Information is usually entered by the user. Systems under this category include a successful effort was made by a type II diabetic patient [34], coloured cards to record dietary information used in Diet Planning and Control System [21], tracking user calories intake using a card with a chip that tracks user calories intake based on purchases and through data entry by the user [13], and smart kitchen that can detect nutrients and energy content of a diet [7].

2.2 Semi-Active Diet Monitoring and Control System

A Semi-Active Diet Monitoring and Control System is a system that gives specifications for diet monitoring and control and also has the ability to directly monitor how the body uses its energy. It may measure the parameters it uses in its analysis. With semi-active system, the user still controls the consumption aspect. What the system does is to provide a more accurate representation of individual's dietary usage. For example instead of relying totally on the individual to enter calories consumed or dispensed, a semi-active system has the ability to measure that directly. BALANCE [9], is a device that detect and calculate individual's caloric usage. PmEB [40] serves the purpose of both storing user's caloric intake and calculating caloric expenditure in real time. Another system that aims at determining the user's caloric expenditure in real time was presented in [22].

2.3 Active Diet Monitoring and Control System

This type of system provides individual with diet based on current needs of the individual's body. It actively reads user's information and determines what kind of diet the user should have. Ultimately, this system should have the ability to keep track of the entire dietary intake of an individual. That is, it should monitor the fact that a user is getting the *recommended daily allowance* based on the user's body and lifestyle. It targets source of the problem; intake. The implication is that if we eat more than we can burn, then the result will be net gain in weight [31], which could lead to all other problems. Therefore it is imperative that we actively control the amount of energy intake. People need help with that control. Because they are aware of the need for a good diet [17], yet they do not adopt it.

Similarly, providing a system that provide instructions alone does not solve the problem as seen in [7]. An ideal system is the one that controls the amount of energy intake and make adjustment when it detect any problem. ADM [2], is system which aims at monitoring diet intake in real time. The system tries to measure changes in human body during food intake. It tracks food intake right from chewing, swallowing, stomach, change in heart rate, weight etc. The system measures all the physiological changes that takes place during the eating process as well as the after effect. It has different sensors that measures all the various activities. The aim of this system, among others, is to eliminate the bias associated with self-reporting and the difficulties in managing those type of data.

An active system is needed to minimize and discourage over consumption which leads to a lots of health related problems and risks.

3. PROPOSED SYSTEM

Consider how a customer uses a Vending Machine. The customer walks to the machine, checks through the available food items, makes a selection and pays. The machine gives out the item and the customer picks it; and that is all. That food item could pose a great risk to the person (for example due to the person's health condition), but neither the person nor the machine knows that. A hypertensive patient might purchase food with high salt content, a diabetic might purchase food with high sugar content as well as high glycaemic index. It is desirable to have a system that can avoid this situation. A system that can know person's condition and the type of diets that are suitable to the person. A system that enables the user to make an informed diet choice, by providing multiple correct (healthy) options.

In response to the above mentioned scenario, we proposed a context-aware system called HELT D (HEalthy Living Through Diet) System. We envisioned a system that would be loaded with *healthy diets* that can serve wide range of people, based on their needs, health and lifestyle. When a user walks up to the system, it should provide multiple food choices to the user that are all correct. This system should measure the user's body to determine key factors in its decision making; weight, diabetes and hypertension. To be able to do this, the system would use sensors.

Like the vending machine, this system would be loaded with food items. But unlike the vending machine, users can only have access to subset of the items, depending on their condition. The reason for that is the fact that not all kinds of food will be suitable for every individual. It is the responsibility of the system to identify those items that are healthy for each user. The list of *available* items will be displayed to the user. Food items that would be loaded would be standardized both in content and packaging. The packaging will enable the system to identify the type of food and its content, while standards in content is to ensure that the identification is correct.

3.1 Operation

The system would be fitted with sensors that would read user's data, using non-invasive methods. A user approaches the system and gets within the sensor, which then reads the user's data. The required data include height, weight, blood glucose and blood pressure. Height and weight would be used in determining body mass index of the individual. The system checks values obtained from the user against *normal values* in order to determine suitable class of diets for the individual. For example a user with high body mass index would be given diet from the low energy density class. Once the class is identified, a list of available diets in that class would be *provided* to the user. The user choose from the list and the system dispense that item to the user. Note that the number of items that would be available to a user depends on the user's data. Also, all the options provided to user are correct and healthy diet choices.

3.2 User Parameters

Body Mass Index (BMI): is an index that is widely used to categorize individuals as underweight, normal, overweight and obese based on their weight and height ($BMI = weightinkg/(heightinmeters)^2$) [20].

Although BMI does not measure amount of fat directly, it correlates strongly with amount of fat in the body [20], which in turn relates to bodyweight [31]. BMI provides a good estimate for measuring obesity [20, 23].

Blood Glucose (BG): blood glucose or *blood sugar* is the amount of glucose present in the blood [11]. It can be measured by pricking the skin to obtain blood to be measured (invasive) [11] or methods that does not require pricking the skin (noninvasive) [41]. While a lot need to be done, we find *Glucoband* [5], very promising. Based on "Bio-Electromagnetic Resonance™" technology, the system can measure glucose from the wrist without the need of disposable parts or additional units.

Blood Pressure (BP): blood pressure can be measured noninvasively using finger, e.g. *CNOGA's TensorTip Vital Signs Monitor* [26].

Calories: as a measure of the total energy in food, calories help us in knowing how to regulate our energy intake [30]. Although energy is required to perform activities, problem would arise when intake is always greater than expenditure. Among other factors, age & sex (see Figure 3) and level of physical activities are very important in determining our caloric needs [30]. By controlling energy intake, we limit the effort that would be needed to expend it. The control is necessary, otherwise the excess energy will be stored as fat [30]. Too much fat result into obesity and all the associated problems.

Age	Males (kcal)	Females (kcal)	Age	Males (kcal)	Females (kcal)
0-3 mo	545	515	11-14 yr	2220	1845
4-6 mo	690	645	15-18 yr	2755	2110
7-9 mo	825	765	19-50 yr	2550	1940
10-12 mo	920	865	51-59 yr	2550	1900
1-3 yr	1230	1165	60-64 yr	2380	1900
4-6 yr	1715	1545	65-74 yr	2330	1900
7-10 yr	1970	1740	74+ yr	2100	1810

Figure 3: Energy Requirement (House of Commons Health Committee et al 2004)

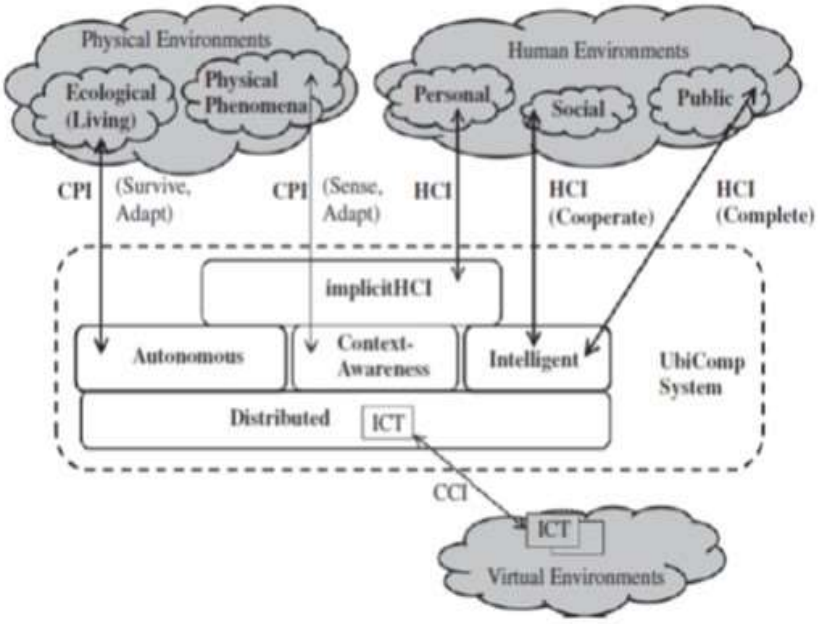


Figure 4: Pervasive system model, dotted line indicates system's boundary (Poslad 2009)

4. PERVASIVE SYSTEM

Pervasive system, also known as ubiquitous system, is defined as "a new paradigm for next generation distributed systems where computers disappear in the background of the users' everyday activities, making data and services readily available at any time and everywhere" [37]. The term ubiquitous computing, first described by [43], aims at providing a fundamental shift in how technology is used in the human environment. Weiser noted that people spend time in learning information about a system that has nothing to do with the task they want to carry out. Ideally, technology should not impose any additional overhead to people using it. According to him, "the most profound technologies are those that disappear". Therefore in pervasive system, the focus of the user is just the task at hand because the system has integrated with the environment that it becomes almost *invisible*. The responsibility of identifying the user, environment, scenario and the type of services to provide lies with the system [35].

To be able to achieve these requirements, pervasive systems use five properties: automation, contextaware, intelligence, implicit Human Computer Interaction (iHCI) and distributed [35]. A pervasive system model is shown in figure 4; CPI, HCI and CCI denotes Computer Physical environment Interaction, Human Computer Interaction and Computer Computer Interaction respectively.

4.1 HELT-D as a Pervasive System

The goal of HELT D is to provide the right diet to the right individual. As an *autonomous* system, it has the responsibility to achieve this goal based on predefined rules and inferred rules. The predefined rules are the ones built into the system, while inferred rules are the ones the system learns based on interactions with the user. The learning ability would be made possible through the system's *intelligence* properties. Intelligence would enable the system to provide services when unexpected condition is encountered (e.g. user responding negatively to a particular diet). HELT D would be able to perform this action because it would be *context-aware*. The system would know user, diet and the result of interaction (how user respond to diet overtime) between user and diet. This implies that the system monitors its context and learns from it.

As a pervasive system, the context is not limited to a single system's location. This is because the system would be *distributed*. This feature would allow the system to monitor and provide services to user at different locations and also enable systems to learn from occurrences at different locations. The result would be a network of interconnected system that interact with the user, environment and one another to provide effective service to the user. When the system is fully functional, it would keep track of user's dietary activities. For example amount of calories consumed and changes in the parameters used. If the system recognizes a problem with the user's health, for example, it would have the ability to alert his/her doctor, providing *implicit Human Computer Interaction*. Analysis could be conducted using these stored records. A conceptual representation of HELT D is shown in Figure 5.

Sensors (e.g. weight sensor) would be responsible for obtaining context data. The *Context Aggregator*, collects and transform these data into a format suitable for the system and store in the *Context Server*. Context Server ensures that the system would continue to function even when there is disruption that prevents it from accessing distributed resources. *Diet* and *Health* components would provide decision making capability on diets and health issue respectively. *Rule Engine* serves as the brain of the system and interact with other components to provide required services. The *Service Provider* component prepares the output to suit user/media. *Application* component serves as an interface between users and the system.

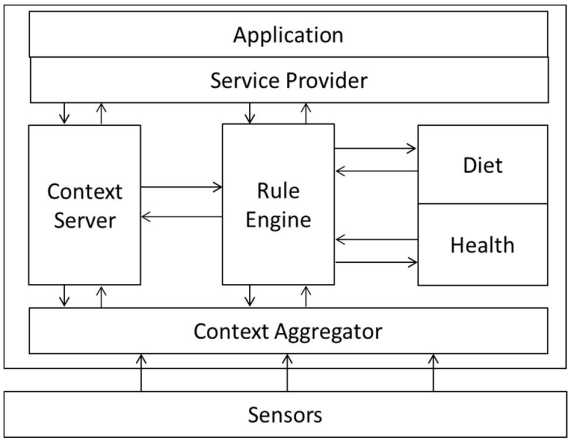


Figure 5: HELT D Conceptual representation

4.2 Formal Specification

Formal specification is a specification made using *formal specification languages*. A formal specification language is a mathematically based language that is used to describe a system [16]. With roots in mathematics, formal specification can be subjected to mathematical operations. Formal specification is explicit and hence avoids ambiguity [16]. It gives us precision and the ability to verify, with high degree of certainty, the completeness of our specification [16].

The existence of tools support for formal specification languages simplifies the process of specification and analysis. There are various types of these tools and can differ in approach and/or suitability to different types of systems. Examples include those that are based on: model (e.g. Z, VDM), Finite State (e.g. FSM, X-Machine), Algebraic (e.g. OBJ, CASL), Process Algebra (e.g. CSP, CCS) [16].

4.3 CCA

CCA Calculus of Context-aware Ambients) is a process calculus for modelling mobile and context-aware systems [38]. Its major attributes are *mobility, context-awareness* and *concurrency*. Entities are modelled as *ambients*. Ambient is defined as "an abstraction of a bounded place where computation happens. It can be mobile, non-mobile, can communicate with peers and can be nested inside another ambient" [38]. For an ambient to perform any functionality or exhibit any behaviour, the functionality has to be provided as a *process*. A process define how an ambient can conduct itself and how it can interact with other ambients. The definition of ambients and processes is done using CCA's syntax which has four main aspects. These are: process (denoted by *P* or *Q*), location(denoted by *α*), capabilities (denoted by *M*) and contextexpression (denoted by *E*). CCA allow us to specify our context-aware system and an interpreter ccaPL (CCA Programming Language) [37], enable us to execute and observe the behaviour of our specification.

The syntax and brief explanation is given below; see [38] for details.

$$P, Q ::= 0 | P | Q | !P | n[P] | \{P\}$$

$$\alpha ::= \uparrow | n \uparrow | \downarrow | n \downarrow | :: | n :: | \epsilon$$

$$M ::= \mathbf{in} n | \mathbf{out} | \alpha \mathbf{send}(\tilde{y}) | \alpha \mathbf{recv}(\tilde{z}) | \mathbf{del} n$$

$$E ::= \mathbf{True} | n = m | \neg E | \Diamond$$

4.4 Processes

The process 0 denotes inactivity, and it halts execution. The parallel processes are expressed as $P|Q$; "—" indicates parallel composition. The symbol "!" define replication. Given a process *P*, the expression !*P* specifies an operation that continuously create an exact copy of *P*. The expression n[P] denotes an ambient named *n* with process "P" modelling the ambient's behaviour. The symbols "[" and "]" indicate boundary of the ambient. For any given process *P*, $P = \{P\}$

4.5 Location

For any given ambient that wants to communicate, there are various destination. The destination could be: $n\uparrow$, named parent *n*; $\uparrow$, any parent; $n\downarrow$, named child*n*; $\downarrow$, any child; $n::$, named sibling *n*; $::$, any sibling; or ε – empty string, within the ambient.

4.6 Capabilities

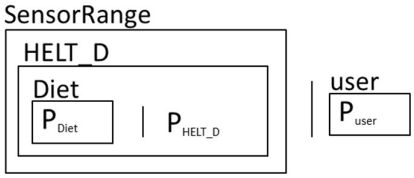
Capabilities are the operations that an ambient has the ability to and is willing to carry out. Ambient can carry out a message exchange capability α **send**$(\tilde{y})$ to send or α **recv**$(\tilde{z})$ to receive list to/from location α. An ambient can also possess mobility – **in** *n*, **out**, capability. The capability **in** *n* denotes moving into sibling ambient *n* and **out** denotes moving out of a parent ambient. To delete an ambient *n* with process 0, that is $n[0]$, we use **del** $n[0]$.

Context-expression: This represents statements and condition, expressed in processes that are correct based on the language rules and would be executed when environment context is right. **True** implies context that is always satisfied, *E* represents context expression, $\neg$ negation and $\Diamond$ denotes *somewhere modality*.

HELT D: There are two ways to represent ambient: textual and graphical. While both of them represent the same thing, the graphical representation gives more vivid picture of the ambient's structure. It is easier to see boundaries and nested structure of ambients if they are presented graphically.

The model of Helt D system is made up of the range which its sensors cover, *SensorRange*, the main system and the user of the system. This is shown both in textual and diagrammatic representation below.

SensorRange[*HELT D*[$P_{HELT\ D}$|*Diet*[P_{Diet}]]]
— *user*[P_{user}]



It can be seen that SensorRange serves as a *parent* ambient of HELT D and HELT D is a parent ambient of Diet ambient. Another way of saying this is that ambient SensorRange contains ambient HELT D and ambient HELT D contains ambient Diet. On the other hand, Diet is a *child* ambient of HELT D and

HELT D is a child ambient of SensorRange ambient. The processes $P_{HELT\ D}$, P_{Diet} and P_{user} model the behaviours which these ambients should demonstrate.

4.7 User Process

If a user is close to SensorRange area, the user can get in to the area. To determine if first ambient is in the same location (same level) with a second ambient, context expression $with_{(n)}$ is used, where *n* represents the second ambient. The expression evaluates to true if both ambients are *siblings* – they are both at the same level. Modelling the user's knowledge of the area, before getting in, is achieved through the use of *context-guarded capability*.

It is represented as $\kappa?M$, where κ denotes expression and M action to be performed. In this case we want the user to perform the action of going into the SensorRange ambient. The user, therefore will use *in* expression, which takes the ambient executing the operation into a specified ambient. The process is as follows

$$with_{(SensorRange)}?in\ SensorRange.0$$

The symbol "." in above process represents *sequential composition* and "0" is *inactivity* – termination of process execution. Once in the SensorRange ambient, the user becomes a sibling of HELT D, remember they are now at the same level. Sibling relationship expressed using the sibling operator "::" and for communication **send**() & **recv**() are used to send and receive message respectively. In the SensorRange area, the user can send his/her identity, BMI, blood glucose and blood pressure. User sending values models system use of sensors to read data. This is expressed as

$$HELT\ D :: \mathbf{send}_{(ID,data)}.0$$

The user then receives list of available diets from the system, send a response of chosen diet and receives a food item.

$$HELT\ D :: \mathbf{recv}_{(AvailableDiets)}.HELT\ D :: \mathbf{send}_{(choice)}.HELT\ D :: \mathbf{recv}_{(fooditem)}.0$$

Finally, the user can get out of SensorRange by performing the *out* operation, which takes ambient out of its parent ambient. User behaviour can be define as follows

$$P_{user} \triangleq with_{(SensorRange)}?in\ SensorRange.HELT\ D :: \mathbf{send}_{(ID,data)}.0$$
$$|HELT\ D :: \mathbf{recv}_{(AvailableDiets)}.HELT\ D :: \mathbf{send}_{(choice)}.HELT\ D :: \mathbf{recv}_{(fooditem)}.0|out.0$$

4.8 HELT D Process

The HELT D system must accept identity and data from any person that wants to use the system and not just particular user. Not specifying any ambient name on the left side of the sibling symbol means from any sibling. The symbol "!" indicates repetition, the system continue to receive as long as there is a sibling that sends (sense user and read in values).

$$!\ ::_{(ID,data)}.0$$

The HELT D system then checks if the user is somewhere in the parent ambient using the *somewhere modality* which is represented by the symbol "✧". $at_{(n,m)}$ context expression checks if two ambients are siblings. An ambient can provide context service or information using a *process abstraction*, which is similar to *method* in an object oriented programming or *procedure* in a procedural programming language. The service would then be obtained through *process call*. The syntax for the process abstraction and process call are $x \triangleright \mathbf{recv}(\tilde{y}).P$ and x **send**$(\tilde{z})$ where x is the process name, $(\tilde{y})$ is the formal parameters and $(\tilde{z})$ actual parameters.

The system needs to login a user to ensure that the diet list to be provided is appropriate to that user.

$$✧at2_{(SensorRange,\ ID)}?login\ \mathbf{send}(\mathrm{ID}).0$$

After login the user, HELT D checks that user is still present, using *somewhere has(n)* expression. If user is present, the system forwards user data and ID to its child (indicated by ↓), Diet ambient, for processing.

$$✧has_{(ID)}?Diet \downarrow \mathbf{send}_{(data,ID)}.0$$

After receiving user data and ID, the Diet ambient sends a reply to its parent, the parent in turn sends a reply to its sibling; the user. The user respond with a choice and the system replies with a food item.

Diet $\downarrow$ **recv**$_{(}$*reply,ID*$_{)}$.*ID* :: **send** $_{(}$*reply*$_{)}$.*ID* :: **recv**$_{(}$*choice*$_{)}$.*ID* :: **send** $_{(}$*fooditem*$_{)}$.0

When user leaves the area, the system needs to log him/her out. The system uses "¬", not operator, and $at_{(n,m)}$ context expression to determine this.

(¬ ✧'*at2*$_{(}$*SensorRange, ID*$_{)}$)?*logout* **send** $_{(}$*ID*$_{)}$.0

What the login process abstraction does is to create an instance of the user ambient, passed as parameter, in the system ambient. The logout ambient erases or deletes the instances using the *del* operation.

login ⊳ (*ID*).{*ID*[0]}
logout $\triangleright_{(}$*ID*$_{)}$.{*del ID*.0}
The HELT _D process definition is shown below:
$P_{HELT\ D} \triangleq$! :: (*ID,data*).0
|(✧'*at2*$_{(}$*SensorRange, ID*$_{)}$)?*login* **send**(ID).0
|✧'*has*$_{(}$*ID*$_{)}$?*Diet* $\downarrow$ **send**$_{(}$*data,ID*$_{)}$.*Diet* $\downarrow$ **recv**$_{(}$*reply,ID*$_{)}$.*ID* :: **send** $_{(}$*reply*$_{)}$.*ID* :: **recv**$_{(}$*choice*$_{)}$.*ID* :: **send** $_{(}$*fooditem*$_{)}$.0
|(¬ ✧'*at2*$_{(}$*SensorRange, ID*$_{)}$)?*logout* **send** $_{(}$*ID*$_{)}$.0
|*login* ⊳ (*ID*).{*ID*[0]}
| *logout* $\triangleright_{(}$*ID*$_{)}$.{*del ID*.0}

4.9 Diet Process

The diet process is responsible for identifying user's condition and selecting appropriate diet for the individual. In this model, user's condition is categorized as *Low, Normal* or *High*. Similarly diets are categorized into either Normal or Low in: *Density* – total energy, *Carbs* – carbohydrate and *Salt*. Based on this arrangement, a total of 21 user conditions is checked and in each case a suitable diet is selected.

The Diet ambient receives user's data and ID from its parent ambient, indicated by "$\uparrow$" symbol.

HELT D $\uparrow$ **recv**$_{(}$*data,ID*$_{)}$.0

The Diet ambient checks for user's condition and sends the correct diet type list to HELT D ambient. The ambient process compares the data received with available conditions. For example, the following process corresponds to a user that has risk of obesity, diabetes and high blood pressure.

(*data* = *BmiHigh BGHigh BPHigh*)?*HELT D* $\uparrow$ **send**$_{(}$*LowDensityCarbsSalt,ID*$_{)}$.0

LowDensityCarbsSalt model list of available diets that target this particular condition. "Low" followed by Density, Carbs, Salt or any combination of the three signifies list targeting a given condition. Density targets BMI, Carbs targets blood glucose and Salt targets blood pressure.

The whole process is defined as follows:

$P_{Diet} \triangleq !HELT\ D \uparrow (data,ID).\{$
$(data = BmiHigh\ BGHigh\ BPHigh) ?HELT\ D \uparrow \mathbf{send}(LowDensityCarbsSalt,ID).0$
$|(data = BmiHigh\ BGHigh\ BPNormal) ?HELT\ D \uparrow \mathbf{send}(LowDensityCarbs,ID).0$
$|(data = BmiHigh\ BGNormal\ BPHigh) ?HELT\ D \uparrow \mathbf{send}(LowDensitySalt,ID)0$
$|(data = BmiHigh\ BGNormal\ BPNormal) ?HELT\ D \uparrow \mathbf{send}(LowDensity,ID).0$
$|(data = BmiNormal\ BGHigh\ BPHigh) ?HELT\ D \uparrow \mathbf{send}(LowCarbsSalt,ID).0$
$|(data = BmiNormal\ BGHigh\ BPNormal) ?HELT\ D \uparrow \mathbf{send}(LowCarbs,ID).0$
$|(data = BmiNormal\ BGNormal\ BPHigh) ?HELT\ D \uparrow \mathbf{send}(LowSalt,ID).0$
$|(data = BmiNormal\ BGNormal\ BPNormal) ?HELT\ D \uparrow \mathbf{send}(Normal,ID).0$
$|(data = BmiHigh\ BGHigh\ BPLow) ?HELT\ D \uparrow \mathbf{send}(LowDensityCarbs,ID).0$
$|(data = BmiHigh\ BGLow\ BPHigh) ?HELT\ D \uparrow \mathbf{send}(LowDensitySalt,ID).0$
$|(data = BmiHigh\ BGLow\ BPLow) ?HELT\ D \uparrow \mathbf{send}(LowDensity,ID).0$
$|(data = BmiLow\ BGHigh\ BPHigh) ?HELT\ D \uparrow \mathbf{send}(LowCarbsSalt,ID).0$
$|(data = BmiLow\ BGHigh\ BPLow) ?HELT\ D \uparrow \mathbf{send}(LowCarbs,ID).0$
$|(data = BmiLow\ BGLow\ BPHigh) ?HELT\ D \uparrow \mathbf{send}(LowSalt,ID).0$
$|(data = BmiLow\ BGLow\ BPLow) ?HELT\ D \uparrow \mathbf{send}(Normal,ID).0$
$|(data = BmiNormal\ BGNormal\ BPLow) ?HELT\ D \uparrow \mathbf{send}(Normal,ID).0$
$|(data = BmiNormal\ BGLow\ BPNormal) ?HELT\ D \uparrow \mathbf{send}(Normal,ID).0$
$|(data = BmiNormal\ BGLow\ BPLow) ?HELT\ D \uparrow \mathbf{send}(Normal,ID).0$
$|(data = BmiLow\ BGNormal\ BPNormal) ?HELT\ D \uparrow \mathbf{send}(Norma,ID).0$
$|(data = BmiLow\ BGNormal\ BPLow) ?HELT\ D \uparrow \mathbf{send}(Normal,ID).0$
$|(data = BmiLow\ BGLow\ BPNormal) ?HELT\ D \uparrow \mathbf{send}(Normal,ID)0$
$\}$

5. ccaPL IMPLEMENTATION

ccaPL (CCA Programming Language) [37], is an interpreter for CCA. It serves as a tool which is used to animate and observe the behaviour of contex-aware systems. That is what we used execute our CCA specification. Screen shots of the system execution (see appendix for ccaPL code) is presented and discussed below. The symbol – – –¿ indicates *reduction* operation – process transition.

5.1 Scenarios

In Figure 6 a single user, Bob who is hypertensive is seen using the system. He entered the sensor range of the system and the system read his data (normal BMI, normal BG, High BP) and ID. The system then logs him in using the *login* process abstraction. Both Bob's data and ID is send to Diet ambient which process and reply to the parent with list (grouped under single name) of suitable diets. The list is presented to Bob. Because Bob's BP is high, the list contains *low salt* diets. Bob then chooses an item and the system dispenses the item. When Bob leaves the sensor range, the system logs him out; using *logout* process abstraction.

Figures 7 and 8 show Alice, with normal health condition and Charles who is diabetic and hypertensive respectively. The operation is the same as described above, except for the type of diets issued. In the case of Alice, the system provides her with normal diet. Due to his condition, Charles is given diet that is low in carbohydrate and salt. A very interesting situation is shown in Figure 9. Alice, Bob and Charles are seen using the system at the same time – *concurrently*. The users are accessing the system at the same time, they all have different conditions and the system is able to provide the correct service to all the users. Alice has normal values, therefore she was given list that contains normal diets. Bob has high blood pressure and was given list that contains low salt diet. Charles has high blood pressure and high blood glucose values; and was given list that contains low salt and carbohydrate diets.

Sequential composition (*P.Q*) was used in user ambient process to obtain the scenarios described above. This is to force the system to execute all the processes. But user should be free to *behave* as he/she wants. This is achieved through *parallel composition*. Figure 10 shows Alice, Bob and Charles exercising their *free will* – every user does what he/she wants (the execution is random, it is possible to have execution that have all of them behaving in the same way). Bob first moved in within sensor range and move out with no interaction with the system.

Charles then move in, the system gets his data, logs him in and passed his data to Diet ambient. Alice also came in, the system gets her data logs her in (not shown), but she moved out. Diet ambient sends reply to its parent with Charles diet. Since Alice has left, the system logs her out. Charles receives his diet list, makes his choice, gets his diet and moves out. The system logs him out. Note that with the parallel composition, the user have absolute control of their behaviour and one cannot predict outcome with certainty.

```
---> {ambient "Bob" moves into ambient "SensorRange"}
---> {Sibling to sibling: Bob ===(Bob,BmiNormal_BGNormal_BPHigh)===> HELT_D}
---> {local call to the abstraction "login" in the ambient "HELT_D"}
---> {Parent to child: HELT_D ===(BmiNormal_BGNormal_BPHigh,Bob)===> Diet}
---> {Child to parent: Diet ===(LowSalt,Bob)===> HELT_D}
---> {Sibling to sibling: HELT_D ===(LowSalt)===> Bob}
---> {Sibling to sibling: Bob ===(choice)===> HELT_D}
---> {Sibling to sibling: HELT_D ===(fooditem)===> Bob}
---> {ambient "Bob" moves out of ambient "SensorRange"}
---> {local call to the abstraction "logout" in the ambient "HELT_D"}
---> {delete ambient "Bob" in ambient "HELT_D"}
```

Figure 6: HELT D System operation with user Bob

```
---> {ambient "Alice" moves into ambient "SensorRange"}
---> {Sibling to sibling: Alice ===(Alice,BmiNormal_BGNormal_BPNormal)===> HELT_D}
---> {local call to the abstraction "login" in the ambient "HELT_D"}
---> {Parent to child: HELT_D ===(BmiNormal_BGNormal_BPNormal,Alice)===> Diet}
---> {Child to parent: Diet ===(Normal,Alice)===> HELT_D}
---> {Sibling to sibling: HELT_D ===(Normal)===> Alice}
---> {Sibling to sibling: Alice ===(choice)===> HELT_D}
---> {Sibling to sibling: HELT_D ===(fooditem)===> Alice}
---> {ambient "Alice" moves out of ambient "SensorRange"}
---> {local call to the abstraction "logout" in the ambient "HELT_D"}
---> {delete ambient "Alice" in ambient "HELT_D"}
```

Figure 7: HELT D System operation with user Alice

```
---> (ambient "Charles" moves into ambient "SensorRange")
---> (Sibling to sibling: Charles ===(Charles,BmiNormal_BGHigh_BPHigh)===> HELT_D)
---> (local call to the abstraction "login" in the ambient "HELT_D")
---> (Parent to child: HELT_D ===(BmiNormal_BGHigh_BPHigh,Charles)===> Diet)
---> (Child to parent: Diet ===(LowSaltCarbs,Charles)===> HELT_D)
---> (Sibling to sibling: HELT_D ===(LowSaltCarbs)===> Charles)
---> (Sibling to sibling: Charles ===(choice)===> HELT_D)
---> (Sibling to sibling: HELT_D ===(fooditem)===> Charles)
---> (ambient "Charles" moves out of ambient "SensorRange")
---> (local call to the abstraction "logout" in the ambient "HELT_D")
---> (delete ambient "Charles" in ambient "HELT_D")
```

Figure 8: HELT D System operation with user Charles

```
---> (ambient "Bob" moves into ambient "SensorRange")
---> (ambient "Alice" moves into ambient "SensorRange")
---> (ambient "Charles" moves into ambient "SensorRange")
---> (Sibling to sibling: Bob ===(Bob,BmiNormal_BGNormal_BPHigh)===> HELT_D)
---> (local call to the abstraction "login" in the ambient "HELT_D")
---> (Parent to child: HELT_D ===(BmiNormal_BGNormal_BPHigh,Bob)===> Diet)
---> (Sibling to sibling: Alice ===(Alice,BmiNormal_BGNormal_BPNormal)===> HELT_D)
---> (Child to parent: Diet ===(LowSalt,Bob)===> HELT_D)
---> (local call to the abstraction "login" in the ambient "HELT_D")
---> (Sibling to sibling: HELT_D ===(LowSalt)===> Bob)
---> (Sibling to sibling: Bob ===(choice)===> HELT_D)
---> (Parent to child: HELT_D ===(BmiNormal_BGNormal_BPNormal,Alice)===> Diet)
---> (Child to parent: Diet ===(Normal,Alice)===> HELT_D)
---> (Sibling to sibling: Charles ===(Charles,BmiHigh_BGHigh_BPHigh)===> HELT_D)
---> (Sibling to sibling: HELT_D ===(fooditem)===> Bob)
---> (ambient "Bob" moves out of ambient "SensorRange")
---> (local call to the abstraction "logout" in the ambient "HELT_D")
---> (Sibling to sibling: HELT_D ===(Normal)===> Alice)
---> (delete ambient "Bob" in ambient "HELT_D")
---> (Sibling to sibling: Alice ===(choice)===> HELT_D)
---> (Sibling to sibling: HELT_D ===(fooditem)===> Alice)
---> (local call to the abstraction "login" in the ambient "HELT_D")
---> (ambient "Alice" moves out of ambient "SensorRange")
---> (local call to the abstraction "logout" in the ambient "HELT_D")
---> (Parent to child: HELT_D ===(BmiHigh_BGHigh_BPHigh,Charles)===> Diet)
---> (Child to parent: Diet ===(LowDensityCarbsSalt,Charles)===> HELT_D)
---> (delete ambient "Alice" in ambient "HELT_D")
---> (Sibling to sibling: HELT_D ===(LowDensityCarbsSalt)===> Charles)
---> (Sibling to sibling: Charles ===(choice)===> HELT_D)
---> (Sibling to sibling: HELT_D ===(fooditem)===> Charles)
---> (ambient "Charles" moves out of ambient "SensorRange")
---> (local call to the abstraction "logout" in the ambient "HELT_D")
---> (delete ambient "Charles" in ambient "HELT_D")
```

Figure 9: HELT D System operation multiple users

Choosing CCA for the formal specification is a good decision because it has context-awareness and mobility as natives in the language and Helt D system is based on context-awareness. But CCA does not have model checker, which means we could not verify or proof properties of the system. However, CCA has associated implementation tool, ccaPL, which can be used to implement the CCA specifications. CCA supports random processes but you have to hard code number of processes that you want with their values, if any. CCA also supports concurrency which allows us to observe the behaviour of many ambients at the same time. For context-aware system, time is important. But CCA focuses on awareness, mobility and concurrency. This implies that temporal aspect of the system could not be modeled using this language. There are other languages that provide that capability, e.g. *Temporal Logic* [42]. It is important to note that the inability of CCA to model a given aspect is not peculiar to it. Therefore during specification, the ideal thing to do is to identify the area of coverage of that language and used it accordingly.

While the model presented in this work communicate the conceived concept clearly, we acknowledged the fact that a lot still need to be done before a concrete implementation is obtained.

```
--> {ambient "Bob" moves into ambient "SensorRange"}
--> {ambient "Bob" moves out of ambient "SensorRange"}
--> {ambient "Charles" moves into ambient "SensorRange"}
--> {Sibling to sibling: Charles ===(Charles,BmiNormal_BGHigh_BPHigh)===> HELT_D}
--> {local call to the abstraction "login" in the ambient "HELT_D"}
--> {Parent to child: HELT_D ===(BmiNormal_BGHigh_BPHigh,Charles)===> Diet}
--> {ambient "Alice" moves into ambient "SensorRange"}
--> {Sibling to sibling: Alice ===(Alice,BmiNormal_BGNormal_BPNormal)===> HELT_D}
--> {ambient "Alice" moves out of ambient "SensorRange"}
--> {Child to parent: Diet ===(LowSaltCarbs,Charles)===> HELT_D}
--> {local call to the abstraction "logout" in the ambient "HELT_D"}
--> {Sibling to sibling: HELT_D ===(LowSaltCarbs)===> Charles}
--> {Sibling to sibling: Charles ===(choice)===> HELT_D}
--> {Sibling to sibling: HELT_D ===(fooditem)===> Charles}
--> {ambient "Charles" moves out of ambient "SensorRange"}
--> {local call to the abstraction "logout" in the ambient "HELT_D"}
--> {delete ambient "Charles" in ambient "HELT_D"}
```

Figure10:HELT- DSystemoperationmultipleusers(parallel)

6. CONCLUSION

Active Diet Monitoring and Control System would be important in helping individuals to control diet intake. It would free user from the task of identifying the appropriate diet. Similarly, it would prevent situation of users falling back to bad dieting. This research work serves as a step towards providing a healthy diet system to individuals. We must eat, but with limit. Exceeding this limit moves us close to health problems. Therefore systems that would help us stay within our limits are essential.

In this paper, we have proposed a novel approach of dealing with dietary problem based on a new system that is context-aware. This work demonstrated that the idea is sound. We believe it is not complete and a lot of work still needs to be done but, the methodology used is valid. Hence our work forms a basis of further studies that would be carried out to fully realize the system.

REFERENCES

[1] MR Akpa and CN Mato. Obesity in nigeria: current trends and management. *Nigerian Medical Practitioner*, 54(1):11–15, 2008.
[2] O. Amft and G. Troster. On-body sensing solutions for automatic dietary monitoring. *Pervasive Computing, IEEE*, 8(2):62–70, 2009.
[3] A. Astrup et al. Healthy lifestyles in europe: prevention of obesity and type ii diabetes by diet and physical activity. *Public health nutrition*, 4(2B; SPI):499–516, 2001.
[4] Y. D. Buowari. Obesity among students attending a tertiary institution in nigeri. *The Internet Journal of Tropical Medicine*, 7(1), 2010.
[5] CalistoMedical. Glucoband, 2012. from http://www.garlic.com/ lynn/secure.htm Accessed: [15/07/2012].
[6] C.M. Champagne. Dietary interventions on blood pressure: the dietary approaches to stop hypertension (dash) trials. *Nutrition reviews*, 64:S53–S56, 2006.
[7] J.H. Chen et al. A smart kitchen for nutrition-aware cooking. *Pervasive Computing, IEEE*, 9(4):58–65, 2010.
[8] A.V. Chobanian et al. Seventh report of the joint national committee on prevention, detection, evaluation, and treatment of high blood pressure. *Hypertension*, 42(6):1206–1252, 2003.
[9] T. Denning et al. Balance: towards a usable pervasive wellness application with accurate activity inference. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, page 5. ACM, 2009.
[10] DiabetesUK. Blood glucose targets. WWW, 2009. from http://www.diabetes.org.uk/ Accessed: [15/07/2012].
[11] DiabetesUK. Blood glucose. WWW, 2012. from http://www.diabetes.co.uk/diabetes Accessed: [15/07/2012].
[12] C.B. Ebbeling, D.B. Pawlak, and D.S. Ludwig. Childhood obesity: public-health crisis, common sense cure. *The lancet*, 360(9331):473–482, 2002.
[13] G.M. Ecer. System and method for diet control, May 2 1995. US Patent 5,412,564.
[14] E.A. Finkelstein, I.C. Fiebelkorn, and G. Wang. National medical spending attributable to overweight and obesity: how much, and who's paying? *HEALTH AFFAIRS-MILLWOOD VA THEN BETHESDA MA-*, 22(3; SUPP):3–219, 2003.
[15] G. Govindarajan, J.R. Sowers, and C.S. Stump. Hypertension and diabetes mellitus. *European Cardiovascular Disease [serial on line], May*, pages 1–7, 2006.
[16] R.M. Hierons et al. Using formal specifications to support testing. *ACM Computing Surveys (CSUR)*, 41(2):9, 2009.
[17] HouseofCommons et al. Obesity: third report of session 2003-04. *London: The Stationery Office*, 2004.
[18] NationalHeartLungandBlood Institute et al. Your guide to lowering your blood pressure with dash. *NIH Publication*, (06-4082), 2006.
[19] K. Kiefer, L. Shirey, and L. Summer. *Childhood obesity: A lifelong threat to health*. Center on an Aging Society, 2002.
[20] P.G. Kopelman et al. Obesity as a medical problem. *NATURE-LONDON-*, pages 635–643, 2000.
[21] D.L. Lennon-Thompson and K.R. Raneri. Diet planning and control system and method, August 21 1990. US Patent 4,950,164.
[22] J. Lester et al. Validated caloric expenditure estimation using a single body-worn sensor. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 225–234. ACM, 2009.
[23] A. Luke et al. Relation between body mass index and body fat in black population samples from nigeria, jamaica, and the united states. *American journal of epidemiology*, 145(7):620–628, 1997.
[24] J.A.E. Manson et al. The escalating pandemics of obesity and sedentary lifestyle: a call to action for clinicians. *Archives of Internal Medicine*, 164(3):249, 2004.

[25] P. McArdle. Your food choices and diabetes. WWW, 2007. from http://www.bda.uk.com/foodfacts/Diabetes.pdf Accessed: [15/07/2012].
[26] CNOGA Medical. Tensortip vital signs monitor. WWW, 2012. from http://www.cnoga.com/Product/vsm.aspx Accessed: [10/8/2012].
[27] Malaysia MinistryofHealth. Clinical practice guidelines management of hypertension. WWW, 2008. from http://www.moh.gov.my/attachments/3885 Accessed: [15/07/2012].
[28] NationalKidneyFoundation. Diabetes and chronic kidney disease. WWW, 2007. from http://www.kidney.org/atoz/pdf/diabetes.pdf Accessed: [15/07/2012].
[29] NHMRC. Dietary guidelines for australian adults. WWW, 2003. from http://www.nhmrc.gov.au/files/nhmrc/publications/attachments/n33.pdf Accessed: [15/07/2012].
[30] NHS. Understanding calories. WWW, 2010. from http://www.nhs.uk/Livewell Accessed [15/07/2012].
[31] NHS. Obesity. WWW, 2011. from http://www.nhs.uk/Conditions/Obesity/Pages/Introduction.aspx Accessed [19/12/2011].
[32] LO Ogunjimi, M.M. Ikorok, and YO Olayinka. Prevalence of obesity among nigerian nurses: the akwa ibom state experience. *Int NGO J*, 5:45–49, 2010.
[33] D.F. O'Neill, E.C. Westman, and R.K. Bernstein. The effects of a low-carbohydrate regimen on glycemic control and serum lipids in diabetes mellitus. *Metabolic Syndrome and Related Disorders*, 1(4):291–298, 2003.
[34] D.A. Paice. Diabetes and diet. *Substance*, 200:220, 1997.
[35] S. Poslad. Ubiquitous computing smart devices, smart environments and smart interaction, 2009.
[36] F.M. Sacks et al. Effects on blood pressure of reduced dietary sodium and the dietary approaches to stop hypertension (dash) diet. *New England Journal of Medicine*, 344(1):3–10, 2001.
[37] F Siewe. Pervasive systems. BlackBoard, 2011. from DMU BlackBoard.
[38] F. Siewe, H. Zedan, and A. Cau. The calculus of context-aware ambients. *Journal of Computer and System Sciences*, 77(4):597–620, 2011.
[39] N.P. Steyn et al. Diet, nutrition and the prevention of type 2 diabetes. *Public health nutrition*, 7(1A; SPI):147–166, 2004.
[40] C.C. Tsai et al. Usability and feasibility of pmeb: a mobile phone application for monitoring real time caloric balance. *Mobile networks and applications*, 12(2):173–184, 2007.
[41] A. Tura, A. Maran, and G. Pacini. Non-invasive glucose monitoring: assessment of technologies and devices according to quantitative criteria. *Diabetes research and clinical practice*, 77(1):16–40, 2007.
[42] Y. Venema. Temporal logic in The Blackwell guide to philosophical logic. pages 203–223, 2001.
[43] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.
[44] WHO. *Diet, nutrition, and the prevention of chronic diseases: report of a joint WHO/FAO expert consultation*. World Health Organization, 2003.