CM10251 Computing as a Science and Engineering Discipline: Coursework 2 Specification 2018/19

Dr Simon Jones & Dr Leon Watts

Date Set	Friday 8th February 2019
Submission Date	4pm Tuesday 16th April 2019 (Moodle)
Estimated effort per person	50 hours
Proportion of units assessment	30%

Abstract

This document describes the second semester coursework for students who are taking CM10251. Students must work in groups and follow an Agile software process that will result in the production and testing of a software system for Personal Informatics (PI).

1 Assignment Information

This group coursework assignment involves the production and experimental testing of a working software system. Your group will be required to produce a written report that explains a problem to be solved, your approach to it and a critical analysis of your work. You must also produce a short video (max. 3 minutes) that shows your working software system.

The purpose of the assignment is to exercise and demonstrate an Agile approach to the lifecycle of a software engineering project, but with a focus on software system verification and experimental testing. You will need to think carefully about the validity of your requirements data gathering and analysis, and the empirical analysis of your software system, as it evolves through its lifecycle.

The domain of the project is Personal Informatics (PI), a field that relates to the design of tools that help people collect personally relevant information for the purposes of self-reflection, self-monitoring and changing behaviour [3, 4]. This information may relate to many different aspects of their behaviours, habits, health and lifestyles. Personal Informatics 'tracking data' can be collected from many different sources, such as wearable fitness trackers (e.g. Fitbit[1]), sensing devices (e.g. smartphones, Arduinos) productivity monitoring tools (e.g. RescueTime[2]), biometric devices (e.g. glucose monitors[3]) and manual inputs (e.g. activity logging apps and questionnaires). Personal Informatics is sometimes also referred to as 'living by numbers', 'personal analytics', 'quantified self', and 'self-tracking' [3].

These systems are intended to offer a wide variety of benefits to their users, from motivating physical fitness [1] and improving self-management of health conditions [5], to providing a better understanding of one's lifestyle [4] and fostering sustainable or environmentally-friendly behaviours [2].

[1] www.fitbit.com
[2] www.rescuetime.com
[3] www.gluco-wise.com

1.1 What to Do - Overview

Your group must work together to research, specify, design, develop and test (verify and validate) software for Personal Informatics.

You are required to develop a Personal Informatics tool for a target audience of your choosing. We encourage you to be adventurous! Your system could be for anything from helping people to manage a coffee addiction, to improving the productivity of software developers.

This task will require the consideration of material covered in lectures, your prior C-SED coursework experience, and additional research in the domain of Personal Informatics. This additional research must involve gathering evidence by interviewing potential users, reading articles on Personal Informatics, and examining existing Personal Informatics systems. You are expected to compare and contrast ideas from these sources, whilst paying attention to the validity of their claims.

Scientists and engineers both rely on the quality of the data they gather to (a) set the scope of the problems they wish to address and (b) to determine the validity of their work. In this coursework, you will address the challenge of 'realism' in dealing with various viewpoints on the value of a software system, as well as 'rationality' as a principle for connecting statements in a coherent manner, via an explanatory frame. You will combine an Agile engineering approach with a scientific mindset to test the claims you make about the value of your software system.

As in Semester 1, we expect you to make use of the Computer Science subject resources linked from the University of Bath Library at http://www.bath.ac.uk/library/subjects/comp-sci/. Accessing resources such as the ACM Digital Library via this page will ensure you are able to download full-text articles without charge. If you are working off campus, make sure you set up a VPN connection so that your machine is recognised by the ACM servers as a legitimate UoB user.

1.2 Programming

Unlike in Semester 1, you will be implementing your designs, carrying out software testing and conducting an experiment using your system. Programming in this unit is treated only as a means to an end — enabling you to experience a test-driven software development lifecycle and scientific experimentation. Consequently your programming ability and code structure will not be assessed as part of this assignment, but you will be required to produce a testable software system. You must submit a video that shows your system working (see Section 6). Tutors will assist you in translating your designs (high and low level) into your implementations and in putting your test plans into action. Lab sessions are timetabled for CM10251 students throughout the semester. You are expected to take advantage of these sessions to implement and test your system.

You are free to choose any platform (e.g PC, Smartphone, Tablet, Arduino) and any object-oriented programming language to develop your Personal Informatics software, providing that you feel competent regarding your selection. Course tutors will only be able to provide support for languages and tools that you have been exposed to on your course so far. You MUST NOT choose tools that cannot implement your designs. Doing so would in effect be choosing to fail because you would no longer be engineering a software system. You would be hacking software without integrity.

2 Initial System Requirements

We have specified an initial set of requirements for your Personal Informatics software. These initial requirements are to be developed and expanded by your group in order to produce a detailed requirements specification as part of your coursework. If you find any ambiguity in these initial requirements this is intentional. You should add further detail by using relevant processes to generate knowledge about your system. All requirements that you produce should be unambiguous and verifiable. When incorporating requirements that we have provided into your Software Requirements Specification you should cite 'Coursework Specification' as their source.

It is important for you to note that whilst you may not ignore the initial requirements, we may add/delete/amend them without notice. As you have been told, a client's needs or the expected environment of operation often change during the software lifecycle. It is therefore important that software engineers follow a software development lifecycle, which allows flexibility in accommodating new requirements or changes thereof. For this reason we are requiring you to follow an Agile Scrum approach. You will be expected to explain in your deliverable how you dealt with any new requirements and how the new requirements are reflected in your design and testing strategy.

Sections 2.1 and 2.2 specify the initial Non-Functional and Functional requirements for your system, respectively.

2.1 Non-Functional Requirements

1. Software Development Process		
1.1	Must adhere to an Agile Scrum software development methodology.	Priority: High Author: SJ
1.2	Your software development should include at least three sprints	Priority: Medium Author: SJ Dependencies: 1.1, 1.3
1.3	Each sprint should last between 1 and 3 weeks.	Priority: Medium Author: LW Dependencies: 1.1, 1.2
1.4	Must regularly review the functional requirements.	Priority: High Author: SJ
1.5	Must incorporate risk management into your software process.	Priority: High Author: LW Dependencies: 1.1

2. Expanding Initial Requirements		
2.1	Must expand upon the initial Non-Functional requirements.	Priority: High Author: LW
2.2	Must expand upon the initial Functional requirements and add additional functionality to the system.	Priority: High Author: SJ
2.3	Additional requirements should be established using appropriate requirement gathering techniques	Priority: High Author: SJ Dependencies: 2.1, 2.2

3. Background Research		
3.1	Must read and cite at least four core articles in the area of Personal Informatics.	Priority: High Author: SJ Dependencies: 3.3
3.2	Should read and cite at least eight articles.	Priority: Medium Author: LW
3.3	Citations of core articles must be to peer-reviewed articles.	Priority: High Author: LW Dependencies: 3.1
3.4	Additional citations may be made to web articles of unknown quality.	Priority: Low Author: SJ

4. Ethical Issues		
4.1	Must account for ethical issues in the specification, design, development and testing of the system.	Priority: High Author: LW

5. Testing		
5.1	Must adopt a test-driven development approach, including production of test plans	Priority: High Author: LW
5.2	Must provide evidence of testing (e.g. static verification bug reports, JUnit output, code coverage metrics)	Priority: High Author: SJ
5.3	Must state hypotheses that clearly link claims to observable behaviours of your system in operation	Priority: High Author: LW
5.4	Must conduct a valid analysis of empirical data generated from an experiment. This should include use of Student's t-test	Priority: High Author: LW

2.2 Functional Requirements

Note that the priority levels and relationships for the following requirements are not specified (i.e. there are also no MUST or SHOULD statements, dependencies, etc.). These will therefore need to be decided by your group.

6. Viewing and Collecting Tracking Data		
6.1	Store relevant tracking data collected by the user (see Section 3 'Tracking Data' for more detail).	Author: LW
6.2	Allow the the user to access tracking data that is stored.	Author: SJ
6.3	Permit the user to manually enter any tracking data which can not be obtained automatically from a tracking device or service (e.g. allowing the user to enter the number of cups of coffee consumed each day, or to enter a rating that represents their mood for the day).	Author: LW
6.4	Permit a user to sort tracking data using a known sorting algorithm (e.g. to rank days by number of steps taken, average temperature, humidity etc.).	Author: SJ

7. Identifying Trends in Data		
7.1	Permit a user to compare their data over different periods of time (e.g. to find out if their physical activity has increased in the last week, or if their productivity is often worse on a Monday).	Author: SJ

8. Goals and Achievements		
8.1	Permit a user to manage targets (goals) for a tracked activity.	Author: SJ
8.2	Permit a user to set a daily or weekly goal value for a particular data variable (e.g. Steps Goal >= 10000, Productivity Goal >= 60%, Cups of Coffee Goal <= 2).	Author: LW
8.3	Permit a user to update or change a goal.	Author: SJ
8.4	Incorporate a feature to motivate the user to achieve their goals (e.g. scoring points, receiving trophies/badges, competing with other users)	Author: SJ

3 'Tracking Data' for your Personal Informatics Software

Your system can make use of any source of Personal Informatics data at all. For example, you may simply ask users to manually enter data by responding to questions in a form presented by the user interface. You may also obtain data from sensors (e.g. using Arduino components) or devices like Fitbit (using their APIs).

You are **not** required to develop working interfaces with existing tracking technologies in order to acquire 'real' data (e.g. downloading data from Fitbit API), however you are encouraged to do so if it is within your abilities. You are welcome to 'mock-up' some tracking data to be used by your Personal Informatics software. For example, you might write some software to automatically generate artificial tracking data, simulating inputs from certain devices or services.

4 Following an Agile Lifecycle: Scrum

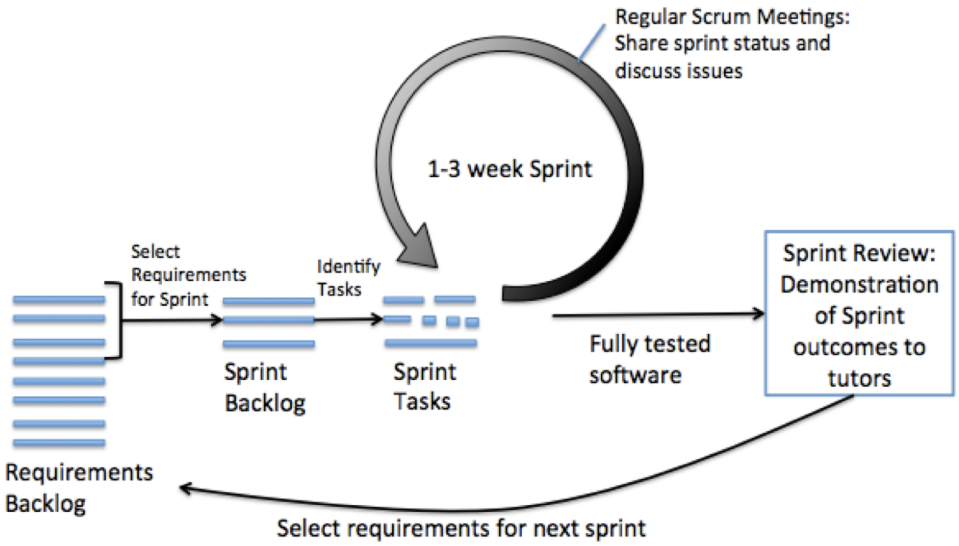

Figure 1: Agile Scrum Development Lifecycle

This assignment requires your group to follow an Agile software development lifecycle involving a minimum of three sprints. Figure 1 illustrates the Scrum development lifecycle. The Scrum lifecycle will also be covered in CM10251 lectures.

You will need to use your requirements specification to prioritise and select requirements for each sprint. Selected requirements will form your sprint backlog, which can then be broken down into tasks for your group members to complete during the sprint. Each sprint should last between 1-3 weeks, and include design, implementation and system verification of your personal informatics software. You should hold regular 'Scrum' meetings during the sprint to share status and discuss issues. At the end of each sprint you should give a short, informal demonstration of your software to the course tutors and discuss the outcomes of the sprint.
Your tutor will give you feedback every week verbally on your risk management and adherence to Agile principles as you carry out your scrum project.

4.1 Monitoring Risks

When risk monitoring is discussed in your meeting, you must be sure to note that fact, including any decisions you make to change the likelihood or severity of any risk on your list, or the addition

or removal or risks from your list of risks. You must include copies of your meeting records as an appendix to your deliverable.

4.2 Coordinating Group Work

You may wish to use technologies that support group work (e.g. Google Docs, Trello, SVN, git/github, pastebin, Facebook groups, Slack, ...), and you are at liberty to do so, but realise that course tutors may not be able to provide assistance for all of the technologies that you choose. It is essential for groups to maintain records or logs of their work so that they can keep track of their software process and report on its management.

You may decide to allocate group tasks to specific individuals or to deal with them as subgroups. You should draw on your experience of group work in Semester 1 to inform your decision. Whatever you decide, make sure you decide as soon as you can and make sure that you achieve consensus within your group about the way the group is going to work.

You are advised to communicate and coordinate regularly online as well as holding face-to-face 'Scrum' meetings. All members of the group should participate, and the group should keep minutes of its meetings. i.e. The date, time (and location of the meeting if applicable), Who is present at the meeting, Issues discussed at the meeting, Who has agreed to do what by the next meeting ("actions"), The date, time (and location if applicable) of the next meeting

As will be detailed in the report specification (Section 6), group members must clearly identify the areas of the report to which they have contributed. Some sections of the report will be contributed to by single individuals, some by multiple group members. All members of the group must participate in the assignment. No group has the right to exclude the work of one or more members from the submitted work of the group.

Each group is required to provide a single-page Group Contribution Form (GCF) with every group member's name, signature and agreed assessment of their percentage contribution to the deliverable. It will directly inform the allocation of marks to each individual group member. In the case that the contributions are not evenly distributed, the document should include a brief paragraph that explains any assessment of an exceptionally high or low contribution.

If no agreement can be reached on the contributions of individual members, the group can hand in more than one GCF. Members will then be required to produce evidence in a hearing with the unit leaders to support their case. In such an eventuality, the Director of Teaching may also be involved.

The best strategy for every group member is to ensure the group as a whole functions well: that they are inclusive, treat members with respect and consistency, and report honestly on the status of work to be completed and any concerns they may have.

5 Conducting a Scientific Experiment with your System

One learning outcome of this unit is to explain how the scientific method relates to the empirical testing of software. In this coursework, you will examine one or more observable aspects of your software system as a means of testing a hypothesis. That means contrasting at least two versions of your software system, based on an independent variable of your choosing (e.g. two independently coded implementations of a particular subsystem; two alternative user interfaces; two alternative sorting algorithms). Your contrast will be accomplished by comparing observations you make of each version and performing statistical analysis using Student's t-test (this will be covered in lectures).

For the purpose of this coursework, empirical testing requires you to quantize your observations. That means defining metrics to represent an observable attribute that you believe is important for the effectiveness of your system as a whole and that is feasible for you to collect. Metrics are often a compromise based on the availability of data in a software system's environment of operation. Not all environments make the same data available. Comparing metrics should allow you to make

an unbiased decision on the relative merits of one version versus another version of your system. You must state your measurement concept as well as the metrics in your analysis, with a critical discussion of both the adequacy of the metrics and suitability of the measurement concept for your purposes.

Here are three examples:

Example 1: Testing usability of user interfaces. You might assess user efficiency (a measurement concept) in carrying out specified tasks with your system (for example, obtaining a summary of data for a particular day) by counting the number of operations they perform (a metric) or the number of seconds taken (an alternative metric for the same measurement concept) with interface version one vs version two.

Example 2: Comparing algorithms for performing operations on PI data. You might contrast the operational performance (a measurement concept) of two recognised sorting algorithms for the data you have collected using your software system. You could measure the number of milliseconds taken (a metric) to return sorted results for different datasets and compare these results between implementations of the two algorithms.

Example 3: Power demand. You might assess the energy usage (a measurement concept) of your software system in processing a known volume of data over a fixed period of time by counting the total CPU time allocated to it (a metric) or percentage of battery drained (an alternative metric).

You are expected to make defensible claims about the quality of your software system or process, supported by the evidence produced by your empirical testing. That means you must make a clear argument for the meaning of the measurements you have decided upon in relation to the viewpoints you have identified. You must be clear about the explanatory frame you believe stakeholders will bring to their understanding of the value of your work.

6 Final Report Specification + Demo Video

To demonstrate your working software you are required to produce a 3-minute video that captures your software in operation (you must show your working software for at least 60 seconds). This should be uploaded to Moodle along with the final report.

There is one written deliverable for this group coursework, which requires submission of a single report. You are strongly advised to have completed your work well in advance of the deliverable deadline, so that the group as a whole is in a good position to finalise their work in time to make the deadline. It is important that the Group Contribution Form properly reflects the final state of your project work. Your attention is drawn to the penalty for late submission of coursework, as set out in the Student Handbook and the associated rules and regulations.

Your final group report will document the complete analysis, design, implementation and testing of your Personal Informatics system, as well as your project management activities as they have implemented your software process.

An electronic version (**pdf format, do not zip**) must be uploaded on moodle by the deadline.

Please ensure that all fonts used are legible and appropriate. Attention should be paid to report presentation: all pages must be numbered, all diagrams must be clearly labelled, and suitable headings and sub-headings used. Fonts in figures should be no smaller than the font in their captions. Number your sections but do not go deeper than four levels of hierarchy (e.g. Section 1.2.3.4).

The expected length of the main body of the report is 30 pages; do not exceed this limit as additional pages will not count toward the assessment. Additional material (such as questionnaires, interview transcripts, details of meetings, etc.) may be included in an appendix. The cover

page, abstract, table of contents, GCF, appendix and references do not count toward the page limit. All sources of information should be correctly referenced and a full list of these references provided at the end of the report.

A guide to the format and salient elements of the report is given below. We have included a guide as to how you might allocate your total "budget" of up to 30 pages between sections of the report. The actual distribution of page count is up to you.

Cover Page

This should include the unit code (CM10251), title of your report, group name and number, and the date. You should also include a table of authors (the members of your group) on this page, e.g.

Group Member	Username	Degree Course
M. Ahmed	msa123	BSc Computer Science (yr. 1)
T. Black	tdbc567	BSc Computer Science (yr. 1)
R. Blanc	rsb324	BSc Computer Science (yr. 1)

Title and Abstract

A single page. The first line should be the report title. On the second line should be a comma-separated list of authors. These lines are followed by an "abstract" of the document - about 200 words summarising the problem, key features of your solution, what your experiment has shown and implications for future work.

Table of Contents

A table containing the title of each section and subsection of the report, with page numbers. Numbering should start from the title page.

Importantly, you should also indicate who were the authors of each section or subsection. In most cases, the authors will be a subgroup who worked together to prepare it. In some cases, the authors will be individual members of your group who were separately delegated to prepare that section.

Introduction (2 page)

A brief introduction that includes:

- A description of the problem area (Personal Informatics) supported by references to good quality contemporary literature.
- A clear statement of the main aim of your software system as a potential solution in the problem area.

Agile Software Process Planning and Management (2 pages)

This section should include evidence of your Agile approach, including how you planned and tracked your sprints, managed the development of your requirements, and how Scrum meetings addressed project risks. This should include an account of how early plans evolved to account for the results of your knowledge generation activities.

Software Requirements Specification (7 pages)

This section should include:

- An account of how you established your detailed system requirements (building on the initial requirements).

- A description of the specific domain of your personal informatics system.
- A description of the various types of user in the problem domain, their viewpoints and their tasks (including ethical concerns).
- Use cases and associated scenarios.
- A set of functional and non-functional requirements for your system, organised into sensible groups and following the guidance we have given you in lectures (e.g. noting relationships and priorities).
- A description of how you managed conflicts between requirements.

Design (5 pages)

This section has been allocated additional pages due to its reliance on figures/diagrams. It should include:

- The high-level architectural box-and-line model of your system with accompanying description.
- A set of UML models describing the low-level design of your system (class models, sequence diagrams, etc.)
- A justification of your chosen design. This should show how your design meets both functional and non-functional requirements by making explicit reference them to illustrate the rationale behind your design.

Make sure you include explanatory text with your design models: the meaning of a diagram is very rarely self-evident because it depends on an explanatory frame for interpretation. So you must make sure to express in words what you wish the reader to understand by them.

Software Testing (Verification and Experiment) (10 pages)

This section should include:

- Testing plans, indicating how you planned to perform verification of your PI system. These plans should reflect your requirements and design work, and be completed before any implementation (coding) work begins.
- Evidence of testing and debugging (you may include unit tests, additional test runs etc. as an appendix).
- Evidence of conducting an experiment with your system, including your hypotheses, procedure, measurement concepts/metrics and results.
- A justification of the approaches you used to examine an important quality of your system.

Reflection and Conclusion (4 pages)

This section should include:

- A critique of your work: requirements specification, high and low- designs, verification work and what was learnt from your experiment.
  - what worked well and why you believe this
  - what might be improved and why you feel it would be better with these changes or additions
- Comparative reflection on the group's software process, including:
  - Evidence of Agility in your software process compared to your semester 1 projects

- How did your group's implementation of a professional code of conduct differ compared to semester 1 (especially regarding responsibilities to self development and to colleagues)?
- How did your group organisation compare with that of your first semester groups?
- What are the main lessons learned across the two SE projects you completed in the first year?

References

A list of references to articles from which you have gathered information as part of your research and have cited in the body of your report. We recommend that you consult the University guide to citing references, which is available in the library. Please use the Harvard (Bath) citation and referencing style: http://www.bath.ac.uk/library/infoskills/referencing-plagiarism/harvard-bath-style.html

Group Contribution Form

The Group Contribution Form (GCF) is a one-page document with every group members name, signature and agreed assessment of their percentage contribution to the project as a whole. It must contain the names of all group members and their signatures with agreed relative contributions to the project over the coursework period.

Group Member	**Contribution** %
Akil Azaki	20
Bernadette Brigham	20
Cecil Snape	10
Doreen Davies	25
Elinor Mau	25
TOTAL	100

If percentage contributions differ by more than 5 points from a simple equal share, you could include a brief statement to explain this. For example: "Cecil did not contribute anything to the group until March, causing significant disruption to the organisation and delegation of project work (e.g. unanswered group emails, stalled work). However, he began to help out conscientiously mid-March, especially to the evaluation, and also contributed some good material to our report."

Appendices

You should include:

- Your one-page GCF, signed by all group members
- Group meeting minutes.
- Transcripts of interviews and/or other evidence of primary research you have conducted.
- Any unit tests or additional test runs you might wish to include with your report.

7 Marking Criteria and Feedback

Note that you will be receiving verbal feedback every week from your tutor in your scheduled tutorial and/or lab session. You will receive marks and written feedback on your final deliverable in accordance with the following criteria:

7.1 Agile Process Planning and Management Activities (30 marks)

Evidence of your decision making and management, including how you planned and tracked your sprints, managed the development of your requirements, and how Scrum meetings addressed project risks.
A critique of your work, including requirements specification, high and low- designs, verification work, knowledge generation and identification of viewpoints.
Comparative reflection on the group's Agile software process, code of conduct, contrasting with experiences in semester one, including possible improvements for future projects.

7.2 Background Research and Requirements Specification (20 marks)

A clearly written account of your analysis of the problem to be solved, including the evidence you have used to support your claims about its key aspects.
A well-organised set of requirements for your system that conform to the principles taught in semester one, including the context of your personal informatics system.
Discussion of ethical concerns, given the various types of user in the problem domain and their viewpoints. This should make explicit reference to a recognised professional code of conduct (e.g. ACM or BCS).
Detailed Use Cases and scenarios in text and graphical form.

7.3 Agile Design and Development (20 marks)

High-level architectural box-and-line model of your system and UML models describing the low-level design of your system (class models, sequence diagrams, etc.), all with explanatory text to justify your chosen design.
Evidence of the rationale for your design, showing connections to functional and non-functional requirements.

7.4 Verification and Experimental Testing (30 marks)

An account of your test-driven verification strategy.
Evidence of testing and debugging, e.g. unit test results.
Evidence of software in operation.
Discussion of experiment with regard to scientific method – especially formulation of hypothesis.
Evidence of contrasting observation with appropriate statistical analysis.

8 Plagiarism and University Regulations

- Late submissions will follow University penalty regulations.
- Use **pdf format** for your report submission because all reports will be checked using the Turnitin plagiarism detection service – do **not** zip your submission.
- Plagiarism is a serious offence. You must therefore make sure that you understand what plagiarism is and how you can avoid it. Please check the following websites for more information:
  http://www.bath.ac.uk/library/help/infoguides/plagiarism.html
  and http://www.bath.ac.uk/students/support/academic/academic-integrity/

References

[1] Fritz, T., Huang, E. M., Murphy, G. C., and Zimmermann, T. Persuasive technology in the real world: a study of long-term use of activity sensing devices for fitness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), ACM, pp. 487–496.

[2] He, H. A., Greenberg, S., and Huang, E. M. One size does not fit all: Applying the transtheoretical model to energy feedback technology design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2010), CHI '10, ACM, pp. 927–936.

[3] Li, I., Dey, A., and Forlizzi, J. A stage-based model of personal informatics systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2010), ACM, pp. 557–566.

[4] Li, I., Medynskiy, Y., Froehlich, J., and Larsen, J. Personal informatics in practice: improving quality of life through data. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems* (2012), ACM, pp. 2799–2802.

[5] Tollmar, K., Bentley, F., Viedma, C., and Libertyville, I. Mobile health mashups. *Proc. PervasiveHealth 12* (2012).