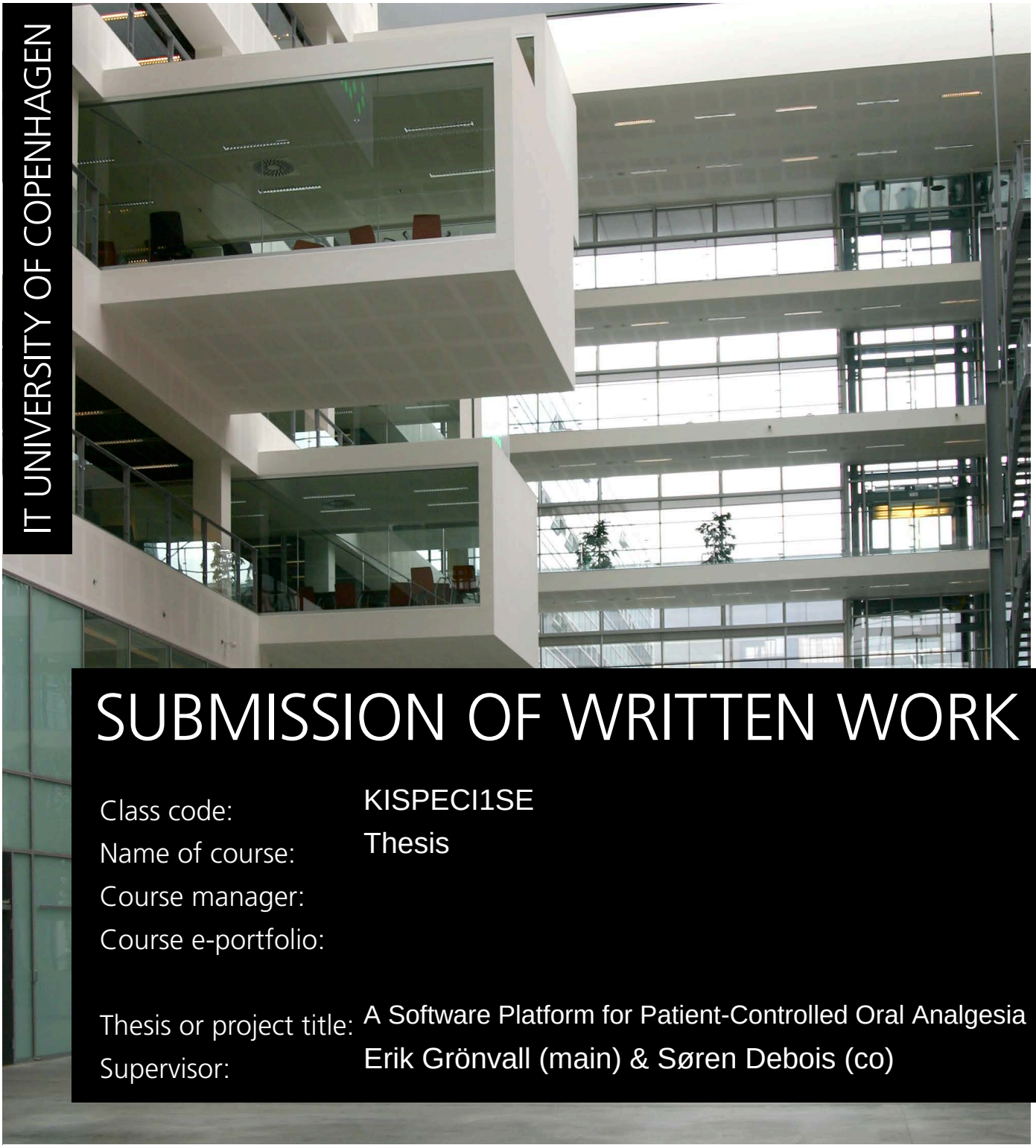
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code: KISPECI1SE

Name of course: Thesis

Course manager:

Course e-portfolio:

Thesis or project title: A Software Platform for Patient-Controlled Oral Analgesia

Supervisor: Erik Grönvall (main) & Søren Debois (co)

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Janus Varmarken	22/05-1989	janv@itu.dk
2.			@itu.dk
3.			@itu.dk
4.			@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk

IT UNIVERSITY OF COPENHAGEN

Master Thesis

A Software Platform for Patient-Controlled Oral Analgesia

Author: Janus Varmarken
Supervisor: Associate Professor Erik Grönvall
Co-Supervisor: Associate Professor Søren Debois

IT University of Copenhagen
Copenhagen, Denmark

A thesis submitted in fulfillment of the requirements for the degree of Master of Science.

June 2017

Dedication

To my dad,

Jens-Erik Varmarken,

who has always been an inspiration to myself and many others.

Preface

Preliminary work on this thesis was carried out as part of the *Thesis Preparation SDT* (TPSDT) course at the IT University of Copenhagen (ITU). The course is divided into two halves. In the first half, the student must define a thesis project and find a supervisor. In the second half, the student starts preliminary work on the thesis, and this work must be documented in a report.

The author chose to spend the second half of the TPSDT course on developing an understanding of the problem domain and reviewing related work. The results of these activities were documented in what would become a draft of the introduction and related work chapters of this thesis. This draft appeared in the report [64] which the author handed in for the TPSDT course. Although both chapters have been heavily revised and augmented since they appeared in the thesis preparation report, some text has been reproduced here. ITU faculty has decided that text from the TPSDT report may be reused in the final thesis, provided that this is explicitly acknowledged and clearly referenced therein.

Abstract

The medical term patient-controlled analgesia (PCA) denotes any method by which the patient controls their own pain relief via self-administration of analgesic medication. In hospitals, PCA is usually practiced using a medication dispenser that allows the patient to self-administer the medication while preventing overdose. This study explores how the patient's personal smartphone can be integrated with the medication dispenser with the purpose of improving PCA treatment. A software prototype, the Painkiller Software Platform, in which the patient's personal smartphone acts as a remote control for the hospital-owned medication dispenser, is proposed. Interviews with patients and nurses and a study of literature on PCA have helped identify where the design has potential for improving PCA treatment. With these findings in mind, the prototype demonstrates how the integration of the patient's smartphone can enhance treatment transparency, automate collection of pain information, and ensure that only the patient can activate the medication dispenser.

Keywords: Patient-Controlled Analgesia, Ubiquitous Computing, Qualitative Interviews, Low-Fidelity Prototypes, Software Prototyping, Risk Analysis.

Acknowledgments

First and foremost, I owe my deepest gratitude to my supervisor, Erik Grönvall, and co-supervisor, Søren Debois. Erik and Søren have both provided invaluable feedback on my work and have been patient with me throughout the entire course of this project. Erik has also served as a personal mentor and provided moral support in times of struggle. For this, I am especially grateful.

I also owe an enormous thank you to my mom, Inge-Merete Varmarken, and my stepmom, Tine Kjøller Varmarken, for all their love, encouragement, and support. Tine is a nurse with many years of experience, and she has helped answer many of my questions regarding working practices in Danish hospitals.

I would like to thank patients and staff at 'Sjællands Universitetshopsital, Køge' for taking time to talk to me about patient-controlled analgesia. I would also like to thank my good friend, Simon Langhoff, for proofreading and commenting on the final draft of this thesis. I also owe my gratitude to my friend and fellow student, Mark Dear, who has assisted me whenever my English skills have proven inadequate.

Tony Beltramelli deserves a big thank you for providing a beautiful LaTeX template. Finally, I would like to thank Philipp Jahoda and Daniel Cohen Jindi for providing a beautiful chart library for Android and iOS, without which visualization of pain and visualization of drug usage would have required a tremendous amount of additional programming.

Contents

List of Figures

List of Tables

List of Listings

1

Introduction

Proper pain treatment is important in order to ensure a swift recovery following major surgical procedures such as total hip replacement. The purpose is to make the patient capable of getting out of bed such that the body can adjust to the change. In the conventional analgesia scheme, the nurse administers the medication. The process generally goes as follows. The patient initiates the process by calling the nurse to their bedside to inform the nurse that they are in pain. The nurse then goes to the medication room to prepare a dose of analgesic medication. When in the medication room, the nurse must first consult the patient's (digital) medical record to make sure that it is safe to administer a dose at the given time. Next, the nurse must manually prepare the dose and make a note in the patient's (digital) medical record, specifying information about the dose such as the drug, the dose size, and the time. Finally, the nurse returns to the patient's bedside to inject the medication (intravenous medication) or to hand out and observe the patient consume the medication (oral medication).

In contrast, patient-controlled analgesia (PCA) is a medical term that refers to any method by which the patient is in charge of their own pain relief via self-administration of analgesic medication. Patients have been self-administering analgesic medication for centuries, and hence PCA is not a new idea, yet it has been a hot topic in medical research for the past decades

due to the development of PCA pumps. PCA pumps are infusion pumps that allow the patient to *parentally* self-administer the medication at the touch of a button. The self-administration is parental in the sense that the PCA pump allows the medical staff to configure the dose size, the minimum time between each dose (referred to as the *lockout period*), the maximum dose per set time (e.g. every four hours), and a continuous background infusion[1]. This prevents the patient from overdosing as the PCA pump will ignore requests for medication if either of these limits have been reached.

In order to be able to decide if a dose of medication can be safely dispensed, the PCA pump must keep track of how much medication the patient has consumed. Drug usage data is hence available and could be presented to the patient for added treatment transparency (i.e. insight into one's own treatment). However, most commercial PCA pumps feature specialized on-pump user interfaces that only the medical staff is allowed to operate. Patients are hence unable to access information about their drug usage, which is ironic as it would be sensible to assume that treatment transparency would be a core component of a *patient-controlled* treatment. This thesis proposes a software platform for PCA that incorporates the patient's personal smartphone. Among other things, the design allows for enhanced treatment transparency through presentation of drug usage information directly on the patient's smartphone.

Today, PCA pumps have become so popular that the term, PCA, often implies that there is a device that ensures parentally controlled delivery of analgesic medication. Medical studies have attempted to pinpoint the key to the popularity and success of PCA pumps. Ballantyne et al. found that patient preference strongly favors PCA over conventional (i.e. nurse-administered) analgesia [3]. According to their results, PCA has a small, yet statistically

[1] A continuous stream of analgesic medication that is automatically dispensed, i.e. the patient does not control it.

significant, positive impact on analgesic efficacy. However, as the effect is small, Ballantyne et al. suggest that the preference may also be rooted in the patients' sense of control of their own level of analgesia. Pain relief is right at the patient's fingertips — there is no need to wait for a nurse to become available, as can be the case with conventional analgesia. A later review by Hudcova et al. arrived at the same conclusions [20]. In addition to these patient-centric benefits, PCA also helps nurses save time, as documented by Jackson [26], since the administration of analgesic medication no longer requires manual labor. Furthermore, Rodríguez et al. have shown that manual preparation of medication is an error-prone task [54]. PCA removes the risk of over- or underdose as dosing is machine-controlled and hence not subject to human error[2].

While PCA pumps have proven very successful in practice, some issues are still left unresolved. The first issue relates to patient safety. Currently, most PCA pumps are activated using a simple click-button. This means that anyone can force a dose upon the patient by picking up the remote control and activating the PCA pump on the patient's behalf. Although the lockout settings *should* prevent overdose from occurring from such behavior, fatal incidents have been reported [8]. The problem has even been assigned its own term: *PCA by proxy* [68]. The second issue relates to the delivery route. Traditionally, PCA devices deliver the medication via intravenous or epidural routes. Viscusi and Schechter identify this as a possible limiting factor for the potential of PCA and conclude that the "development of new technology offering alternative routes for PCA administration is at the forefronts of PCA research" [66].

[2]Programming mistakes during initial setup of the PCA device can cause over- or underdose.

1.1 Objective and Scope

Recent advancements in consumer electronics present new opportunities for securing and personalizing PCA by incorporating the patient's personal device(s). Specifically, this thesis asks the following research question:

> How can the patient's personal smartphone be integrated with the PCA device with the purpose of improving PCA treatment?

The research question entails several subquestions, in particular:

- What potential do end users see in the extra functionality that may be offered by integrating the patient's personal smartphone with the PCA device?
- How can the hospital-owned PCA device and the patient-owned smartphone locate each other and communicate securely without the need for a tedious set-up procedure?
- What are the risks involved in making the PCA device a networked device and employing the Bring Your Own Device (BYOD) scheme in a hospital setting?

A specification of scope is necessary as the research question is quite broad. First, inspired by the agenda set forth by Viscusi and Schechter [66], this thesis will focus on constructing technological aids for PCA delivered through the *oral* route (i.e. the analgesic medication is in tablet form), henceforth referred to as Patient-Controlled Oral Analgesia (PCOA). A clinical study on pain treatment after cesarean delivery by Davis et al. [12] serves as an example of the importance of the oral route in (patient-controlled) pain treatment. The study found that patients, who received manually administered

oxycodone-acetaminophen tablets, experienced better pain relief and fewer side-effects than patients who received intravenous morphine using standard PCA equipment. These findings suggest that the oral route is – under certain circumstances – superior to the intravenous route. In turn, this warrants the development of technology for PCOA as there is potential for reaping the same benefits as intravenous PCA brings to the table. On the other hand, as the focus is on PCOA, the findings produced here do not *necessarily* apply to intravenous PCA.

Second, the thesis will only focus on building the *software components* for technology-assisted PCOA. It is acknowledged that the physical tablet dispenser, henceforth referred to as the PCOA device, is a crucial component. However, designing the physical enclosure and mechanical dispensing mechanism is in larger part a mechanical and/or electrical engineering task than it is a software development task, and possibly deserves an entire thesis of its own.

1.2 Contribution

A software solution for PCOA, the Painkiller Software Platform, that integrates the patient's personal iPhone with the PCOA device is proposed. The platform consists of three software applications: Painkiller Server, which runs on the PCOA device, Painkiller Patient, which runs on the patient's iPhone, and Painkiller Staff, which runs on the medical staff members' iPads. The Painkiller Server is the focal component. It is responsible for controlling the physical behavior of the PCOA device and managing all patient and treatment data. However, it provides no user interface. Instead, it exposes a secured REST service. Painkiller Patient and Painkiller Staff are client applications of this REST service. They are responsible for providing a user

interface for the functionality exposed by Painkiller Server, including rendition of treatment data in a meaningful way. In addition, the two client applications are also responsible for performing (local) user authentication.

Interviews with end users and a literature study have helped identify and confirm a few aspects of PCA treatment that could be improved by integrating the patients smartphone with the PCOA device. The Painkiller Software Platform demonstrates these ideas. First, the platform prevents unauthorized activation of the PCOA device (e.g. unauthorized PCA by proxy) by requiring that the patient authenticates themselves using the fingerprint scanner of their iPhone. Second, Painkiller Patient automatically prompts the patient to submit information about their pain. This is a task that is currently handled manually by the nurse and is often forgotten. Third, the Painkiller Software Platform provides the patient with a greater level of insight into their own treatment as the Painkiller Patient application allows them to access information about their drug usage and pain history on their personal iPhone. Fourth, the medical staff can access the same information from their iPads, which may allow for a more informed evaluation of the efficacy of the analgesic medication.

1.3 Report Outline

The remainder of this thesis is structured as follows. Chapter 2 makes use of a scenario to illustrate how the Painkiller Software Platform will be used in the real world and, through inclusion of screenshots, also serves as a user guide. Chapter 3 presents an overview of related work. Chapter 4 explains what analytical tools the author has put to use to tackle the research question. Chapter 5 presents the findings of a user study that was carried out with the purpose of uncovering users' interest in the envisioned system. Chapter 6

describes the technical implementation of the Painkiller Software Platform. Chapter 7 presents a risk analysis that was carried out in order to understand the risks that the Painkiller Software Platform introduces. Chapter 8 discusses the findings presented in the previous chapters. Finally, the conclusion is presented in ch. 9.

2

Scenario

This chapter presents a scenario that illustrates how end users might interact with the Painkiller Software Platform, and what tasks the system helps its users accomplish. Screenshots of the user interfaces (UI) are included so that the scenario may also serve as a user guide for the Painkiller Software Platform. Benyon et al. distinguish between four different types of scenarios: user stories, conceptual scenarios, concrete scenarios, and use cases [6, pp. 192-199]. The scenario presented here is an example of their definition of a use case as its purpose is to *document* how users interact with the system. The scenario has been developed in collaboration with an orthopedic theatre nurse, Mette Zwergius, to ensure that the described usage of the Painkiller Software Platform conforms with current working practices in Danish hospitals.

2.1 Operating Room

It is Monday morning at Smallville Hospital. Patient Jane Doe is in the operating room. She is about to receive total knee replacement. Surgeon Schmidt informs her that she will now receive anesthesia and that she will be waking up in the recovery ward. He explains that the nurses in the recovery

ward will have a hospital porter bring her and her personal belongings to a nursing ward when she is conscious and able to breathe on her own. Schmidt says that the nursing ward staff will keep her under observation for a couple of days, after which she will be able to go back home.

2.2 Preparation in the Nursing Ward

Meanwhile in the nursing ward, nurse Nielsen is going through the list of today's incoming patients. He is scheduled to receive a patient, Jane Doe, who has undergone total knee replacement. The doctor has prescribed that Jane *must* consume 100 mg of Tramadol[1] every six hours for the first 24 hours in order to numb her pain sufficiently enough for her to be able to get out of bed[2]. Additionally, the doctor has prescribed that Jane *may* be given supplementary PRN[3] Paracetamol when the Tramadol alone is unable to numb Jane's pain[4]. The PRN prescription states that Jane may receive a 1000 mg Paracetamol dose every four hours, yet at most four doses every 24 hours. Nielsen checks Jane's digital medical record and finds that Jane has given her consent to use the Painkiller Software Platform, which allows

[1] An opiod pain medication often used in Danish hospitals for the treatment of post-operative pain.

[2] The use of *mandatory* doses of Tramadol (or a similar drug) at fixed intervals is standard procedure in Danish hospitals for the treatment of post-operative pain following total knee replacement. The purpose is to ensure that the patient achieves some level of analgesia such that they can get out of bed, as this has a positive effect on the recovery process. Present work is not concerned with these mandatory doses, but they are mentioned here to make the scenario mirror reality as closely as possible.

[3] *Pro re nata*. When used in the context of medicine, the phrase means *as needed* or *when necessary*.

[4] It is important to understand the difference between the mandatory medication and the PRN medication. Failure to take the mandatory medication on time is considered non-adherence and pose a threat to the success of the treatment. In contrast, it is favorable if the PRN medication is not taken on time as this type of medication should only be consumed if necessary. Present work targets the PRN medication.

Jane to self-administer her PRN medication. This prevents delays as Jane will hence not have to wait for Nielsen to bring her the medication, when she is in pain. In addition, the system also frees Nielsen from spending time on managing Jane's PRN medication.

2.2.1 Creating a Patient Entry

Nielsen uses his iPad to configure the Painkiller Software Platform such that it is ready for when Jane arrives. As Smallville Hospital has many PCOA devices, Nielsen must first locate the PCOA device next to Jane's assigned bed in the list of available PCOA devices (see fig. 2.1). Each physical PCOA device is labeled with a unique call name which Nielsen uses to select the corresponding virtual representation of the PCOA device in the list in fig. 2.1. This takes him to a list of patients who have been (or currently are) using the selected PCOA device (see fig. 2.2). He taps the button labeled with a "+" in order to create a patient entry for Jane. The action brings up the screen shown in fig. 2.3. Nielsen enters Jane's personal information and taps the "Save" button. He is prompted to verify his fingerprint (see fig. 2.4) in order to ensure that only he can use his iPad for configuring the PCOA device. He places his thumb on the fingerprint scanner of his iPad and is successfully authenticated. Nielsen is then taken back to the list of patients shown in fig. 2.2, but now the list also contains an entry for Jane Doe.

2.2.2 Creating a Prescription

Nielsen selects Jane's name in the list of patients and is presented with a screen that shows an overview of Jane's personal and medical data (see fig. 2.5). From here, Nielsen selects the tab labeled "Medication" which takes him to a screen that shows a list of medications prescribed for Jane (see fig.

Figure 2.1: Painkiller Staff: UI for selecting which PCOA device to configure.

Figure 2.2: Painkiller Staff: UI for selecting which patient to view.

Figure 2.3: Painkiller Staff: UI for creating a new patient entry. The "Save" button remains disabled until all required input has been provided.

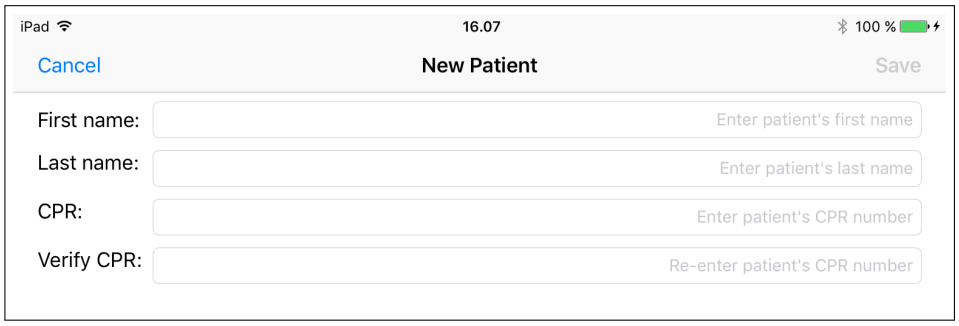
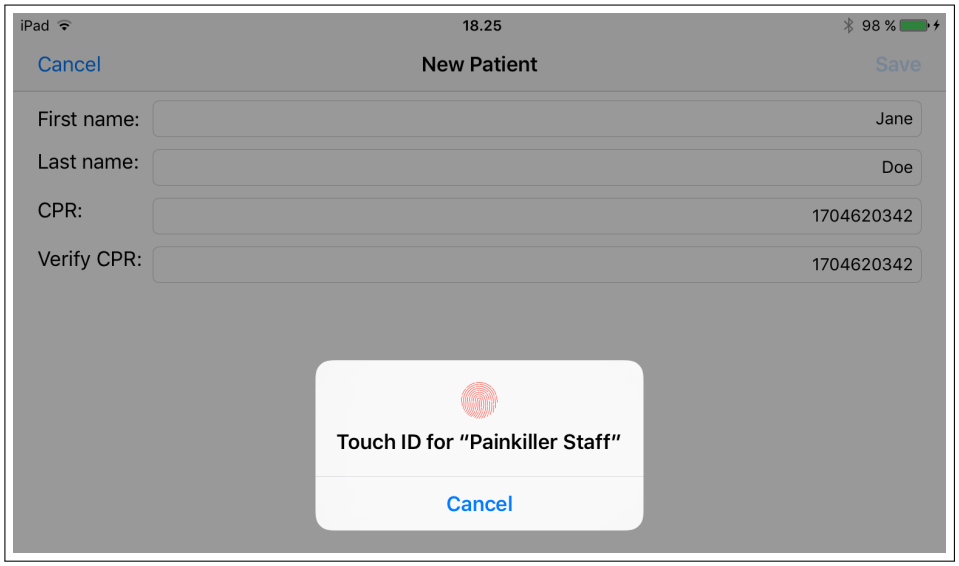


Figure 2.4: Painkiller Staff: fingerprint authentication prompt.



2.6[5]). Similarly to when he created the patient entry for Jane, Nielsen now taps the button labeled with a "+" to indicate that he wants to set up a new prescription for Jane, and he is taken to the screen shown in fig. 2.7.

Nielsen taps the row labelled "Drug" which brings up a screen with a list of the drugs that are available in the hospital's inventory (see fig. 2.8). He selects "Paracetamol, 500 mg tablet" and returns to the previous screen (fig. 2.7) by tapping the back button labeled "New Prescription". As mentioned earlier, the doctor has prescribed that each dose should contain 1000 mg of Paracetamol, and Nielsen therefore uses the step-buttons in the row labeled "Tablets per dose" to adjust the tablet count per dose to be two. Next, Nielsen specifies the lockout settings (see fig. 2.7). The lockout settings are used by the PCOA device whenever it must decide if a request for a dose should be granted. He configures the *lockout period*, i.e. the minimum time that must elapse between two successive doses, to be four hours. With a

[5]Note that Nielsen will see an empty list, as there are no prescriptions for Jane yet. The screenshot contains a prescription to visualize how it will appear in the list.

Figure 2.5: Painkiller Staff: UI that presents an overview of a patient's personal and medical data. Additionally, this UI also provides the functionality that allows the nurse to initiate the pairing procedure, which lets the patient connect their personal iPhone to the PCOA device.

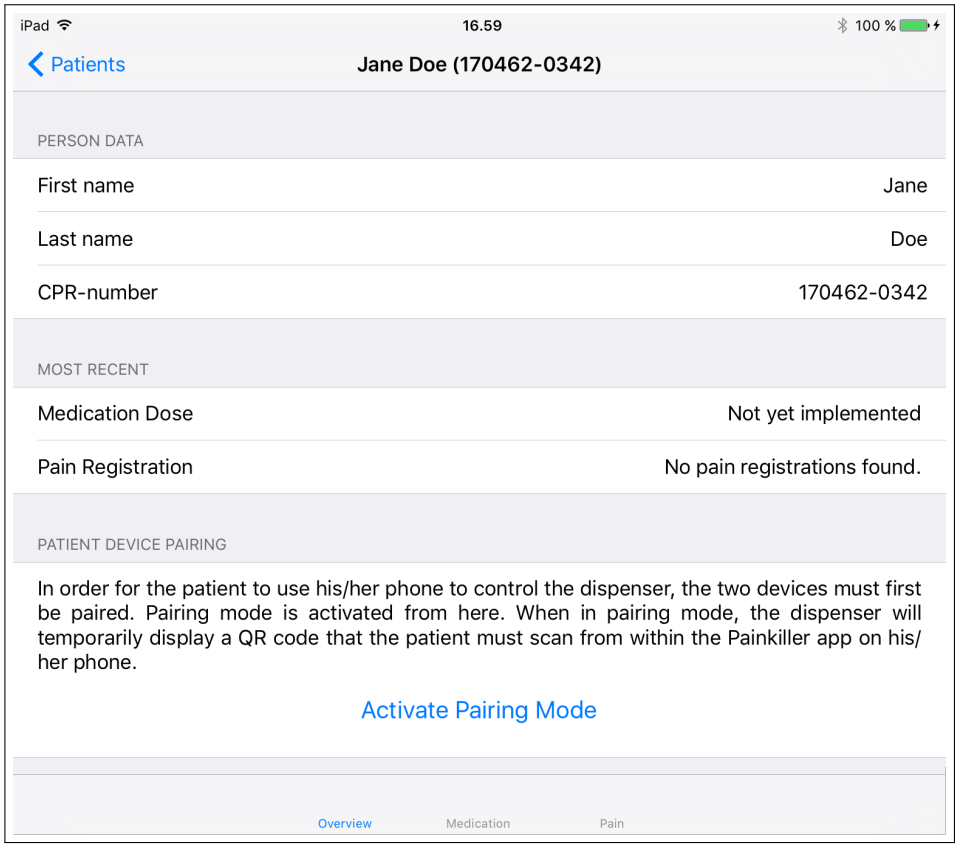


Figure 2.6: Painkiller Staff: UI for managing a patient's list of prescriptions.

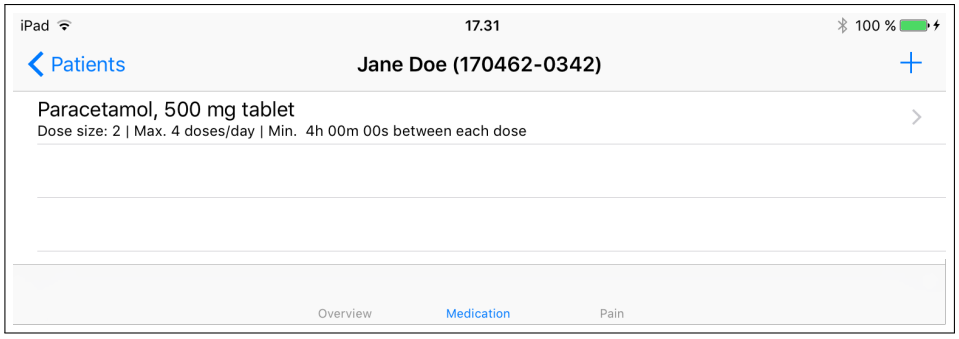


Figure 2.7: Painkiller Staff: UI for creating a new prescription.

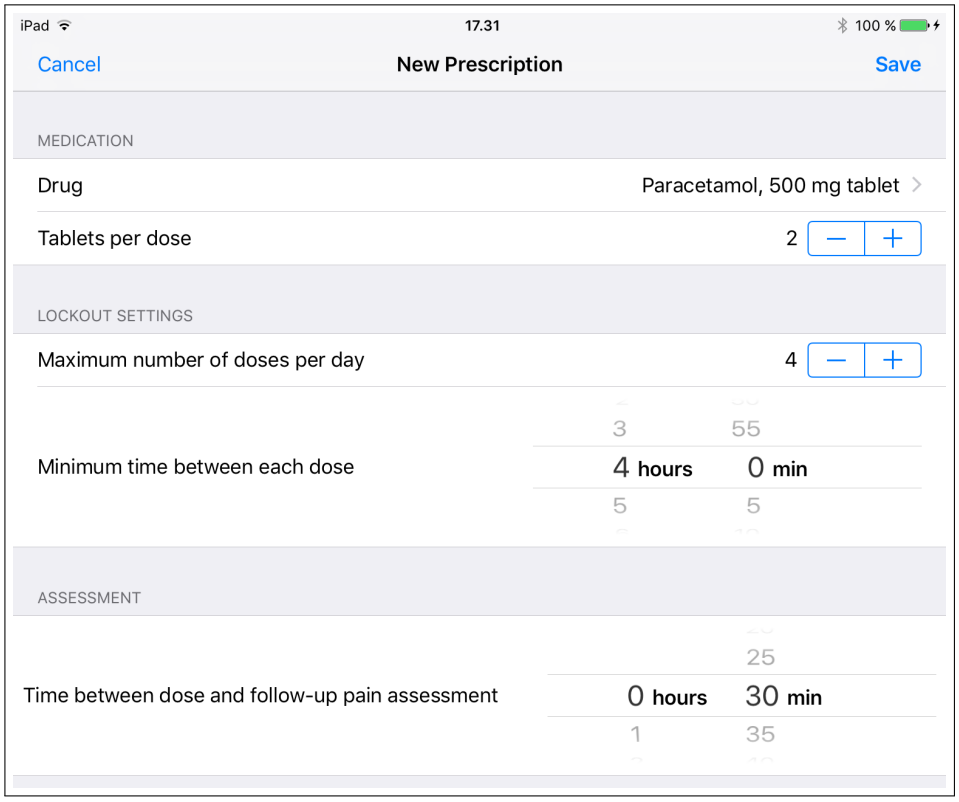
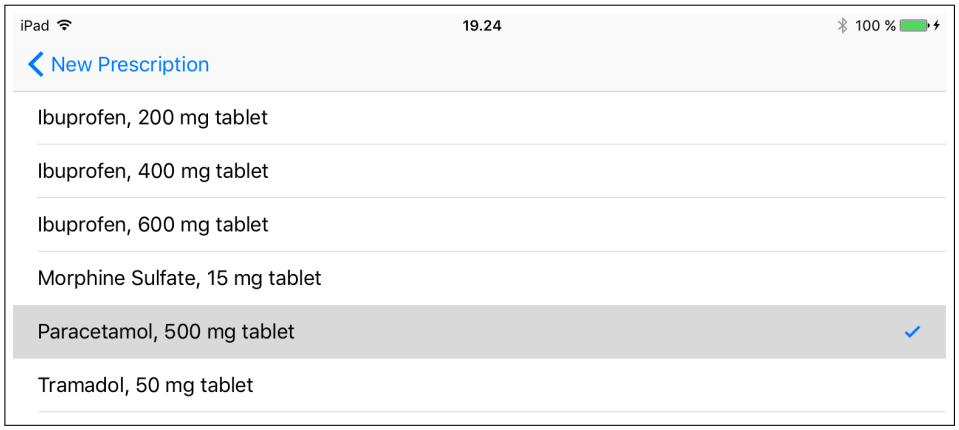


Figure 2.8: Painkiller Staff: UI for selecting the medication of a prescription.



lockout time of four hours, Jane can potentially consume six doses daily. However, the doctor has prescribed that Jane may only consume four doses daily, so Nielsen specifies a *24 hour limit*, i.e. the maximum number of doses the patient may dispense every 24 hours, of four. Finally, nurse Nielsen sets the *reassessment time* to be 30 minutes (fig. 2.7 bottom). The reassessment time is the time that should elapse between the consumption of a dose and a follow-up evaluation of its effect. Nielsen then taps the "Save" button and is prompted to verify his identity by scanning his fingerprint (using a UI similar to the one shown earlier in fig. 2.4, omitted for brevity), and the new prescription data is sent to the PCOA device once the fingerprint has been successfully verified.

The final step remaining before the configuration is complete is to physically load the PCOA device with Paracetamol tablets. The PCOA device is portable, so Nielsen simply brings it with him to the medication room, unlocks its medication tray, fills the tray with Paracetamol tablets, and locks the tray again. The PCOA device is now preloaded for the patient's arrival, but is left in the medication room for now for security purposes.

2.3 Enrolling the Patient

The operation goes well, and Jane arrives in Nielsen's nursing ward at noon. Nielsen stops by the medication room to pick up the PCOA device and then goes to welcome Jane to the ward. He shows Jane the PCOA device, and explains that it is an electronic system that can help Jane safely manage her PRN analgesic medication on her own, so that she does not have to wait for a nurse when she is in pain. Nielsen reminds Jane that she gave her consent to use the system, and confirmed that she owned a compatible iPhone, when she attended her medical pre-examination a few weeks earlier. He asks Jane

Figure 2.9: Painkiller Server: UI that facilitates device pairing.

to reconfirm that she is still interested in using the system, and Jane agrees.

Nielsen explains to Jane that the system is controlled directly from Jane's personal iPhone, and that it will therefore have to be connected to the hospital's Wi-Fi. Whenever a patient is admitted, the hospital issues a set of Wi-Fi credentials for the patient. Nielsen has brought Jane's credentials with him on a piece of paper. While Jane logs onto the Wi-Fi and downloads the Painkiller Patient application from the Apple App Store, Nielsen unlocks his iPad and navigates to the screen that presents an overview of Jane's personal and medical data (fig. 2.5). He taps the button labeled "Activate Pairing Mode" and is prompted to verify his fingerprint. Once the fingerprint has been verified, a request is sent to the PCOA device, informing it that it should prepare for Jane to connect. The PCOA device responds by displaying a QR code and a countdown timer on its small display (see fig. 2.9). The QR code embeds information that allows Jane's iPhone to locate the PCOA device on the network and prove its identity to the PCOA device (this will be explained in detail in sect. 6.3.2).

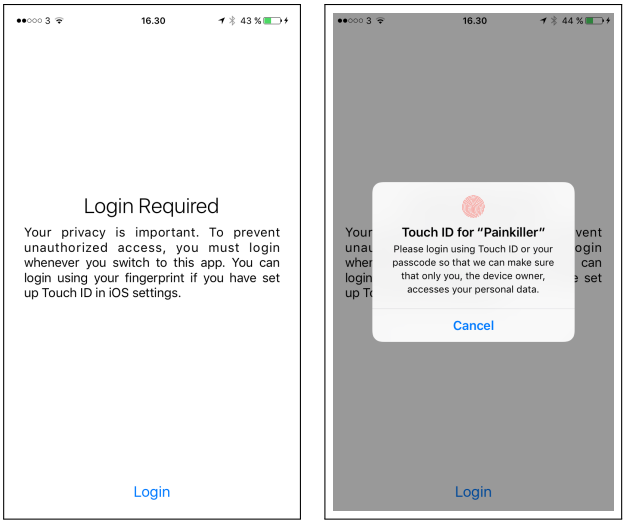
Nielsen instructs Jane to launch the Painkiller Patient application. When Jane opens the application, she is prompted to login using her fingerprint (see fig. 2.10)[6]. Once Jane has successfully authenticated herself, she is

[6]This UI is presented whenever Painkiller Patient is launched or switched to. The pur-

taken to a new screen that shows information about the PCOA device she is using (see fig. 2.11a), but no data is displayed as Jane's iPhone still needs to be connected to the PCOA device. Nielsen instructs Jane to tap the "Edit" button, and a QR code reader appears (see fig. 2.11b). Nielsen directs Jane to scan the QR code displayed by the PCOA device (fig. 2.9), and a prompt appears on Jane's iPhone asking her to confirm her identity (see fig. 2.11c). The purpose of this prompt is to allow for detection of human errors occurring during setup (e.g. if the nurse has mistakenly selected the wrong patient). Once Jane confirms her identity, the Painkiller Patient application on Jane's iPhone proves its identity to the PCOA device by sending the answer (extracted from the QR code) to a challenge set forth by the Painkiller Server application running on the PCOA device (technical details are provided in sect. 6.3.2). Painkiller Server responds with a token that Painkiller Patient can use to prove its user's identity to Painkiller Server in all subsequent requests.

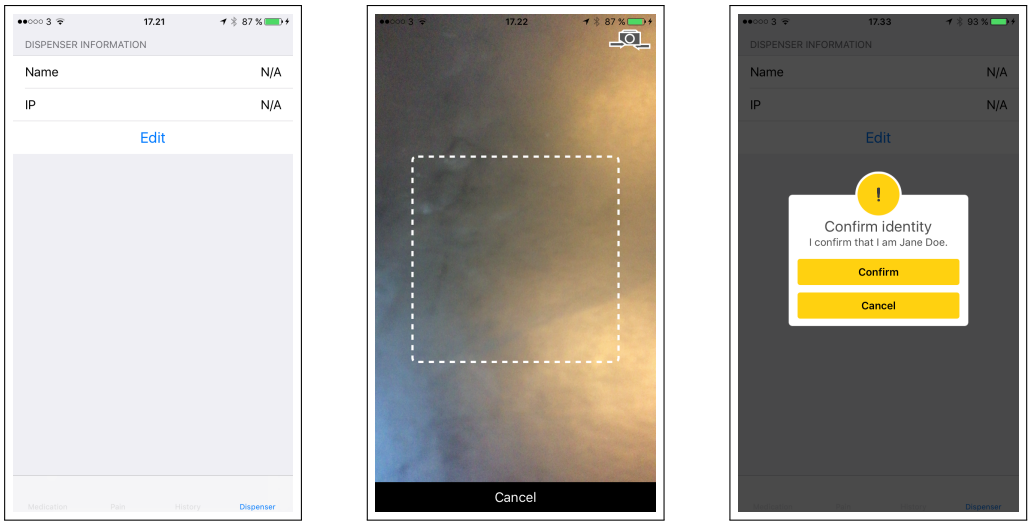
The three disabled tabs at the bottom of fig. 2.11a become enabled just as the pairing procedure completes. Nielsen tells Jane to select the tab labeled "Medication" (see fig. 2.12a) and explains that Jane will be using this tab whenever she wants to dispense a dose. He adds that Jane must first select *which* medication to dispense as the system can be used for handling multiple different medications. As per Nielsen's guidance, Jane taps the row labeled "Medication" which brings up the list of medications prescribed for Jane (see fig. 2.12b). Jane selects the Paracetamol prescription that Nielsen added as part of preparing the system earlier that day. Nielsen lets Jane know that she is now set to dispense her medication on her own. He informs Jane that she will be prompted to assess her pain when she dispenses a dose, and yet again 30 minutes after consuming the dose. Before leaving Jane, Nielsen briefly explains the pain scale and that the purpose of the pain assessments is to let the medical staff evaluate the effect of the pain treatment.

pose is to restrict access to personal information (e.g. in case the iPhone is left unlocked).



(a) Login screen. (b) Fingerprint prompt.

Figure 2.10: Painkiller Patient: UI for logging in using fingerprint when the application is launched or moves from the background to the foreground.



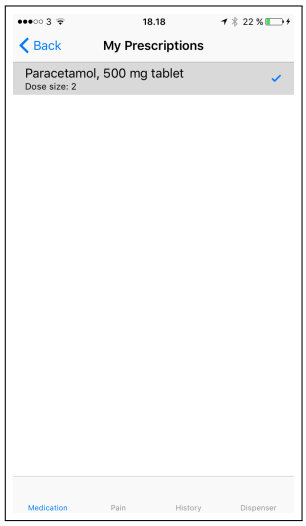
(a) Setup is initiated by editing the current PCOA device.

(b) Patient scans QR code displayed on the PCOA device.

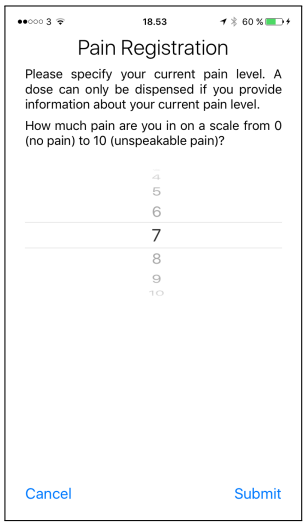
(c) To prevent mistakes, the patient is prompted to confirm their identity.

Figure 2.11: Painkiller Patient: UI for connecting to the PCOA device.

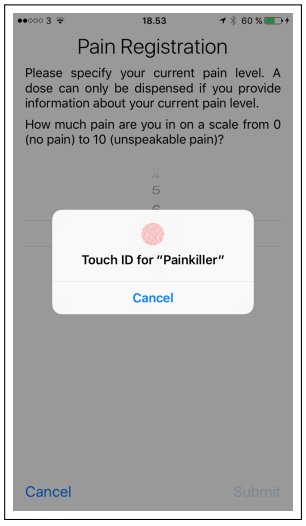
(a) The Medication tab from which the patient initiates a request for a dose.



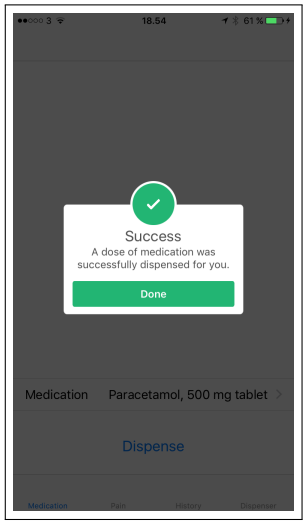
(b) Medication selection. The system can assist the patient with managing multiple analgesic medications.



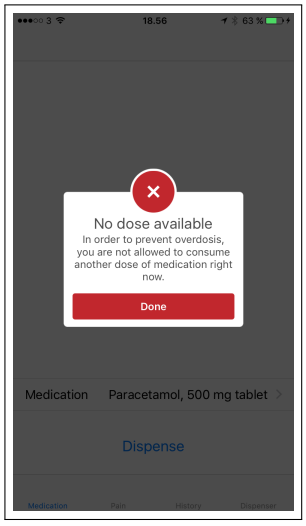
(c) Pain registration prompt. The patient must provide a pain assessment when requesting a dose.



(d) The patient is prompted to authenticate herself using her fingerprint when requesting a dose.



(e) A request to dispense a dose was approved.



(f) A request to dispense a dose was denied because of an ongoing lockout period or the 24-hour limit being reached.

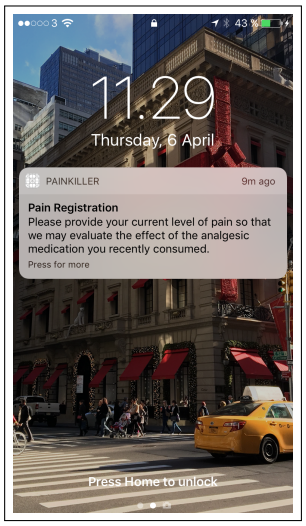
Figure 2.12: Painkiller Patient: UI for dispensing a dose.

2.4 Dispensing a Dose and Assessing Pain

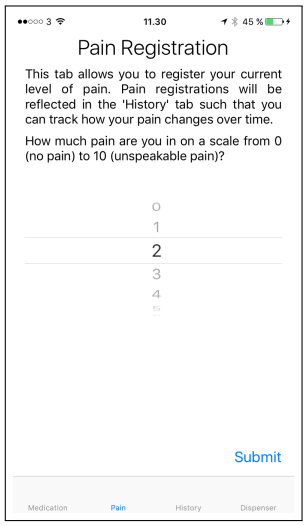
About an hour later, Jane starts to feel pain as the sedation from the surgery starts to wear off. She picks up her iPhone and opens the Painkiller Patient application. After logging in (fig. 2.10), she navigates to the Medication tab (fig. 2.12a) and requests a dose of Paracetamol by tapping the button labeled "Dispense". A prompt asking Jane to assess her pain appears (see fig. 2.12c). Jane is in quite severe pain, so she selects a pain level of 7 out of 10[7]. She taps the "Submit" button and is prompted to authenticate herself using her fingerprint (see fig. 2.12d) to ensure that no one else but Jane herself can dispense a dose on her behalf. Upon successful authentication, a request for a dose is sent to the Painkiller Server application on the PCOA device. After establishing the authenticity of the request, the Painkiller Server verifies if a dose is available. As no doses have been dispensed yet, the request can be safely granted, so the physical dispensing mechanism of the PCOA device is triggered, and an acknowledgement response is sent back to Jane's iPhone. Jane is presented with a dialog that informs her that her dose was dispensed (see fig. 2.12e). She picks up the tablets from the dispensing tray of the PCOA device and swallows them with a sip of water.

30 minutes later, Jane's iPhone notifies her that it is time to reassess her pain (see fig. 2.13a). However, Jane has fallen asleep in the meantime. When Jane wakes up two hours later, she picks up her iPhone and taps the notification, which opens the Painkiller Patient application. After logging in (fig. 2.10), Jane is presented with the "Pain" tab (see fig. 2.13b). Jane feels much better than earlier, so she selects a pain level of 2 out of 10 and taps the "Submit" button. She is prompted to authenticate herself by scanning her fingerprint (the UI is similar to the one in fig 2.12d). Upon successful

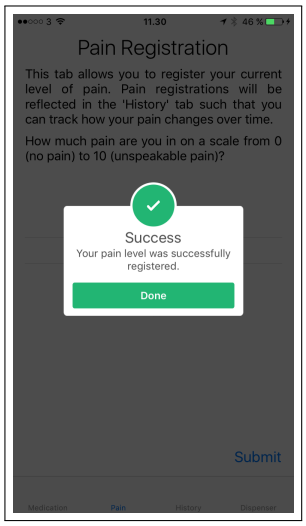
[7]The Numeric Rating Scale (NRS-11) is used. NRS-11 is an 11-point scale that is widely used for patient self-reported pain. The scale ranges from 0 (no pain) to 10 (unbearable pain).



(a) The patient is notified that it is time to reassess her pain level.



(b) The "Pain" tab from which the patient submits her current pain level.



(c) Pain level submission success indicator.

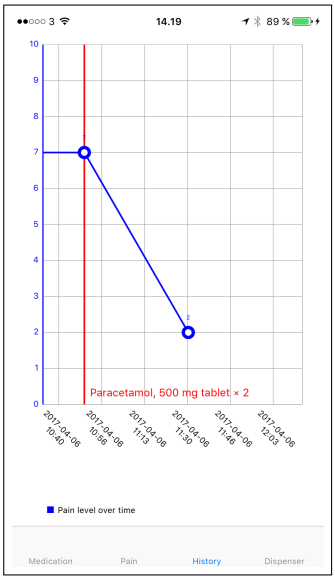
Figure 2.13: Painkiller Patient: UI for pain reassessment. UIs for required fingerprint-based authentication have been omitted as they are similar to those shown earlier.

authentication, the pain level is sent to the PCOA device, and Jane is presented with a status dialog (see fig. 2.13c). The Painkiller Software Platform handles Jane's delayed reaction to the pain reassessment prompt gracefully by timestamping the pain level registration at the time of submission rather than at the time of the notification.

Jane starts exploring the Painkiller Patient application. She discovers that she can access information about how much medication she has consumed and how her pain level has changed over time (see fig. 2.14). This pleases Jane as she believes that it is important to consume as little analgesic medication as possible and therefore it is important for her to keep track of her usage.

Another 30 minutes pass, and Jane gets out of bed to visit the restroom.

Figure 2.14: Painkiller Patient: drug usage and pain level over time.



The physical activity makes Jane well aware of her new knee – it is very painful. When Jane returns to her bed, she decides to use the Painkiller Patient application to request another dose of Paracetamol to numb the pain. However, when doing so, Jane is presented with the screen shown in fig. 2.12f. At this point in time, only three hours have elapsed since Jane's last dose of Paracetamol, and she is denied another dose because nurse Nielsen configured the lockout time to be four hours.

2.5 Evaluating the Pain Treatment

The following morning, doctor Dawson, who is the doctor responsible for post operative pain treatment, is in his office. Dawson is going through a list of patients, who received surgery yesterday, to see how they are doing. He is currently looking at Jane's drug use and pain history on his iPad (see

Figure 2.15: Painkiller Staff: patient's drug usage and pain level over time.

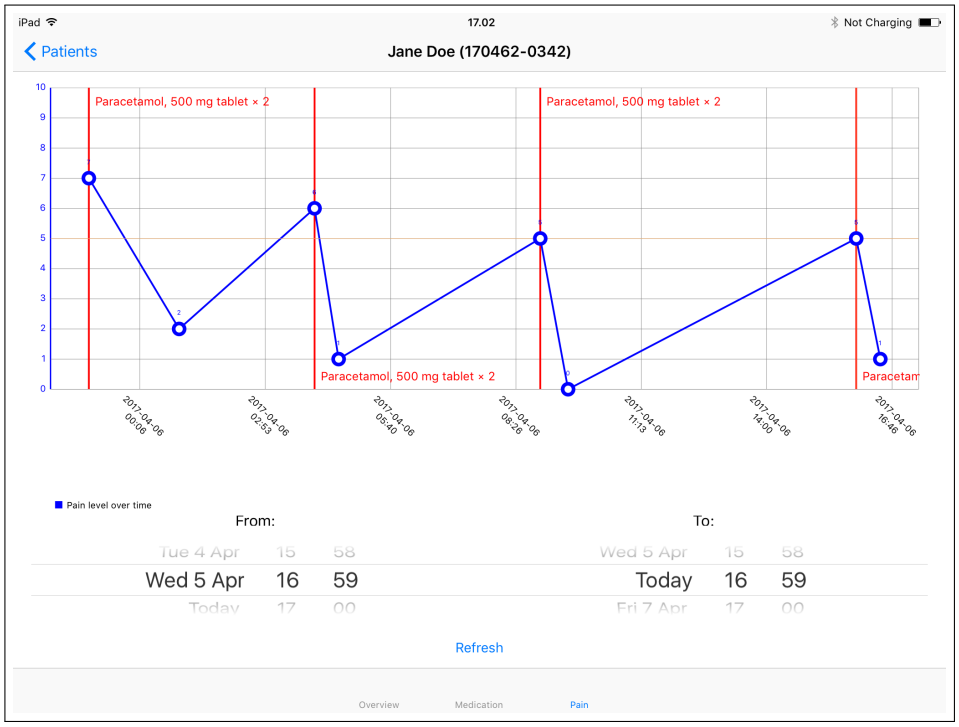


fig. 2.15). Through experience, Dawson knows that the surgical procedure performed on Jane often causes patients to be in a lot of pain within the first 24 hours. Looking at the graph, he confirms that Jane does indeed experience moderate to severe pain. However, he also sees that whenever Jane has used the PCOA device to dispense a dose of PRN Paracetamol, her pain significantly lowers shortly after. Dawson is hence pleased with the effects of the drugs prescribed for Jane and deems that there is no need to alter the treatment.

3

Related Work

The idea of using a smartphone as a remote control for an appliance, in this case a medication dispenser, is not new per se. Section 3.1 provides a brief historical overview of systems that have explored this idea. Note, however, that this should not be considered an exhaustive list.

An important contribution of present thesis is the demonstration of the practicality of using the patient's private smartphone as an authentication tool, while keeping the necessary set-up procedure simple. For this reason, sect. 3.2 highlights other inventions that bridge identity in the physical world with identity in the digital world, and argues why these alternatives are less favorable.

A vast amount of research has investigated how IT can be used to build smart medication dispensers for tablets. However, most of this work targets home care and does not necessarily fit in a hospital setting. Section 3.3 highlights a few of these systems and explain why they cannot serve as a PCA device for oral medication in a hospital. Section 3.4 gives an overview of existing PCOA solutions, and highlights shortcomings of these systems.

3.1 Wireless Activation of Appliances

Pioneering work in the area of remote control of appliances using handheld devices was carried out as part of the Pebbles project at the Carnegie Mellon University and presented by Myers in [42]. In particular, the team behind the Pebbles project, henceforth referred to as the Pebbles group, introduced the concept of a *Personal Universal Controller* (PUC) [46, 43]. A PUC was a handheld device (e.g. a PDA) that acted as a remote control for multiple home and office appliances. The Pebbles group's hypothesis was that the increase in the number and/or complexity of functions offered by appliances in turn had made their user interfaces harder to use. They argued that the solution was to move the user interfaces onto a handheld device as it possessed more processing power and better input-output capabilities, which could be used to enhance the usability. A significant contribution of the Pebbles group's work, described in detail in papers by Nichols et al. [46, 45], was a user interface generator that automatically generated PDA/smartphone user interfaces for appliances from high-level, abstract specifications of the functions of the appliances.

The idea of a using a handheld computer as a universal remote control for multiple home appliances was later revisited by Lee et al. [36], but implemented differently as they made use of visual tags for appliance recognition in an augmented reality.

Other early work in the area has explored how a user can control a display from a palmtop computer. Noteworthy mentions are Rekimoto's M-Draw [52], SharedNotes by Greenberg et al. [16], and U S West Homes by Robertson et al. [53]. In M-Draw, users can draw and type on their personal palmtop computers and then choose to paste their creations onto an interactive whiteboard for collaboration with other users. SharedNotes allows users to post and modify notes on a public display using their palmtop computers.

U S West Homes is a digital real estate catalog in which the user browses home listings by controlling an interactive TV from a PDA.

Sjölund et al. explored so-called *Distributed User Interfaces* (DUI) [58]. Unlike the PUC, in which the *entire* user interface was moved to the smartphone, Sjölund et al. sought to improve the user experience and make the most of the available screen space by moving *parts* of the user interface to proximate devices, hence letting the user use the devices in concert. They demonstrated their idea by building a smartphone application that would act as a remote control for the Windows Media Player running on a PC. The user interface of the remote control offered only the basic functions such as play, stop, and volume control, while the user interface of the PC application offered only the more advanced functions.

Since these early systems, smartphones have been used as remote controllers for a plethora of appliances such as robots [19, 47, 2], video games on remote displays [63, 38, 29], HVAC [33], and lighting [34, 33].

3.2 Bridging Physical and Digital Identities

Patient wristbands with bar codes is the de facto standard for patient identification in the Danish Healthcare System [60]. Unfortunately, bar codes have their limitations as an identification tool as they do not read well when wrinkled, wet, or torn [18, 32]. Moreover, a bar code is insufficient as a "key" that prevents unauthorized use of a medication dispenser as it is extremely easy for an adversary to copy it: all it takes is visual access – for example, one can use a smartphone to take a picture of the bar code and then later trick the bar code scanner by presenting it with the picture displayed on the smartphone's screen.

Cangialosi et al. predict that the adoption of radio-frequency identification (RFID) technology in hospitals can have a profound impact on patient safety, if implemented correctly [7]. Just like their bar code-based siblings, RFID wristbands can be used for patient identification, for example to ensure that a dose of medication is given to the right patient. Although the price of a passive RFID tag is as low as a few cents and therefore perhaps negligible, use of RFID technology as an authentication and authorization mechanism still requires that the medication dispenser is equipped with additional hardware, namely an RFID reader. As modern smartphones come pre equipped with fingerprint scanners, which can serve as an authentication tool as demonstrated by present thesis, the extra cost of an RFID reader is unwarranted.

Matsushita et al. present a system in which a ubiquitous computer is personalized by the user's touch [39]. The user wears a *wearable key* (a digital transmitter) around their wrist, and the computer is equipped with a *ubiquitous keyhole* (a digital receiver). Both of these items are equipped with electrodes that touch the user's skin when they interact with the devices. The user's body is used as conductive liquid that facilitates communication between the wearable key and the ubiquitous keyhole when the user touches both items at the same time. User authentication is achieved by using the established link for transferring a secret known to both the wearable key and the ubiquitous computer. While this form of authentication is very user friendly as it is quick and natural, it is not immediately clear if it is suitable for use in hospitals as the electrodes and the RF signal going through the patient's body might interfere with other medical equipment that rely on electrodes attached to the patient's body (e.g. electrocardiography equipment).

Santos-Pereira et al. propose a system, OFELIA, that grants a patient control over other people's (e.g. healthcare personnel and family members)

access to their Electronic Health Record [57]. In OFELIA, the patient's personal smartphone is the patient's interface to the system and hence acts as the object that bridges identity in the physical world with identity in the digital world. The patient enrolls in the system by carrying out a registration procedure at one of the hospital's desktop computers. The procedure involves scanning of a QR code and manual entry of a one-time password. This is similar to the procedure required for enrolling a patient's iPhone in the Painkiller Software Platform (described in detail in sect. 6.3.2). However, the procedure in Painkiller Software Platform is simpler from a user's point of view as it does not require manual data entry. Furthermore, while OFELIA offers an interesting and seemingly secure approach to authorization management and boasts a simplistic enrollment procedure, it requires a comprehensive infrastructure consisting of several servers and a desktop computer.

3.3 Smart Medication Dispensers

McCall et al. propose a system that helps a patient administer multiple medications [40]. The system is comprised of a plate fitted with a rotational spoke that divides the plate into sections. Each individual medication container is placed in its own section. When it is time for the patient to take a given medication, the spoke rotates the medication container onto a scale. By calculating the difference in weight of the medication container prior to and after the patient has taken the medication, the system can verify if the patient took exactly the number of tablets corresponding to the prescribed dose size. This device is insufficient as a PCOA device as it grants the patient access to the entire stash of medication as opposed to just a single dose. Needless to say, this is dangerous when dealing with strong analgesic medication.

An earlier system by Tsai et al. is similar to the one presented by McCall et al. in that it helps the patient administer multiple medications, but its design is different [61]. In this system, each medication container is placed in its own socket. The system alerts the patient when it is time to take a medication by sounding an alarm, and the patient pushes a simple click-button to start the dispensing procedure. The system indicates the target medication container by lighting a small ring around its socket, and a small display informs the patient how many tablets to take. The procedure proceeds by the patient manually moving the tablets from the medication container to a verification box that performs visual inspection (it is equipped with a camera) to verify that the patient took out the right amount of tablets. Once the tablet count has been verified, the procedure concludes by the tablets being dispensed to a removable cup. The system has a locking mechanism that prevents the user from removing a medication container from its socket until the time when the medication is to be consumed. However, the locking mechanism only ensures that the patient picks up the correct medication container. It does not prevent access to the medication as the medication container is taller than its socket (imagine a mole poking out of its hole), and its lid can therefore easily be removed. Since the system allows access to the entire stash of medication and does not require the patient to authenticate themselves, it is, alas, unsuitable as a PCOA device.

Pak and Park propose a smart medication dispenser that is allegedly appropriate for both home care and hospital use [49]. The authors claim novelty for two design features of the dispenser: it can be used simultaneously by multiple (up to six) patients, and comes with a facility for remote management by medical staff. Unfortunately, the dispenser is activated using a simple push-button, and there is hence nothing that prevents one patient from (intentionally or unintentionally) operating the dispenser on behalf of another patient.

Crema et al. present an architecture for a smartphone operated medication dispenser that they claim could be used in both home care and hospitals [10]. The medication dispenser is controlled using an Android application, which runs on the smartphone and communicates with the medication dispenser using NFC. They declare that their use of a smartphone, as a user-friendly human-machine interface and patient identification tool, is novel. However, the identification claim is weak as it relies entirely on the assumption that the smartphone is a personal device. There is no mechanism in place that ensures that whoever uses the smartphone to activate the medication dispenser is in fact the patient and not someone else. Furthermore, the proposed system does not offer functionality that allows caregivers to follow the patient's treatment. The Painkiller Software Platform addresses the patient identification issue by performing fingerprint verification whenever an action is carried out, and also provides an application for the caregiver that offers access to treatment data.

Bardram describes a (vision for a) tablet container for hospital use that employs RFID technology to sense the proximity of the correct patient and also uses a fingerprint scanner (built into the tablet container) to verify the patient's identity [4]. Bardram envisions a design that entails departments, one for each dose of medication, which are programmatically controlled to only reveal the proper dose at the proper time. Admittedly, Bardram was first to envision the use of fingerprint matching for patient identification in the medication-taking process, but his proposed design is more costly than the one presented here, as it requires additional hardware (in the form of a fingerprint scanner) on the medication dispenser.

Finally, it should be noted that all systems mentioned in this section are, by design, unsuitable as PCOA devices as they target medication that *must* be taken according to a prescribed schedule. Consequently, these systems consider deviation from the medication schedule to be abnormal behavior

and respond by notifying the patient and/or medical staff about the issue. "Schedules" for the intake of analgesic medication are of opposite nature as medication is taken on an as-needed basis, although with limits on how often a dose can be consumed. Deviation from the "schedule", that is to say *not* reaching the medication limits, is actually desired as doctors generally agree that patients should only consume the minimal amount of medication that is necessary in order to achieve analgesia.

3.4 Existing PCOA Solutions

Kastanias et al. present a low-tech take on PCOA [30]. In their approach, a patient is allowed to keep a single dose of oral, immediate-release opioid analgesic at their bedside in a child-resistant vial. After taking a dose, the patient calls a nurse who refills the vial. While this scheme brings all the patient-centric benefits of PCOA, it does not lighten the nurse's work. Each dose must be prepared and delivered manually. Kastanias et al. mention that this sometimes caused delays, which defeats the purpose of PCOA. Moreover, the nurse must also manually keep and consult a record of doses consumed by the patient. This reintroduces the risk of under- and overdose due to human error, which (intravenous) PCA equipment prevents. Finally, as the vial does not enforce patient authentication, other patients (e.g. drug addicts) may steal the dose.

Rosati et al. present an evaluation of PCOA device for pain management in oncology inpatients [55]. In this system, the patient activates the medication dispenser by scanning an RFID armband. The armband uniquely identifies the patient to the device, ensuring that the patient has exclusive access to their personal medication. The medication is packed in a dose tray that resides beneath a security cover, which exposes a single dose at a time.

When the patient scans the RFID armband, the device rotates the dose tray which exposes the next dose. 95% of patients experienced better pain control using the device, and a commercial edition of the device was later released [9]. The commercial edition was evaluated by Stubbs and Monsivais who confirmed that patient satisfaction was increased due to better pain control [59]. However, the study also revealed that the device had usability problems (such as lengthy programming times) and a severe technical issue in form of an unreliable power cord, which occasionally rendered the device useless. Furthermore, the solution does not offer anything in terms of treatment transparency, i.e. the patient cannot access live information about their drug usage.

Zalviso is a PCOA device by AcelRx Pharmaceuticals [1] that has been shown to be a non-inferior alternative to intravenous PCA morphine sulfate for the management of post-operative pain [41]. The device is a hand-held dispenser that lets the patient self-administer sufentanil tablets for sublingual use. The patient uses an RFID thumb tag to identify herself to, and activate, the dispenser. Zalviso boasts being less invasive than intravenous PCA equipment which tethers the patient to the bed. However, the RFID thumb tag may still be a nuisance (e.g. when the patient performs fine motor tasks such as smartphone operation), and it is not a hygienic choice. Moreover, Zalviso only supports one specific medication and hence does not serve as a general PCOA device for arbitrary tablets.

4

Method

This chapter describes the set of analytical tools that have been put to use to tackle the research question.

4.1 User Study

A small user study was carried out at the Danish hospital, 'Sjællands Universitetshopsital, Køge'. The hospital was chosen because it was conducting a trial on the use of PCA pumps for the treatment of post-operative pain. This gave the author direct access to patients and medical staff members who had firsthand experience with PCA. It is acknowledged that the medication delivery route in the trial was different from the delivery route in the envisioned system (intravenous vs. oral), but the author argues that this is not an issue as the purpose of the user study was not to understand the differences between oral and intravenous PCA, but rather to develop an understanding of the stakeholders' views on technology-assisted PCA, including what role data transparency plays.

The user study was conducted during the early stages of the project such that its findings could guide development efforts. Semi-structured, qualitative interviews were used. These are powerful during idea generation since

interviewees can talk more freely and express their own ideas, while the interviewer makes sure that the important topics are addressed. Two patients and two nurses were interviewed. The patients were mainly asked questions about how they perceived the PCA treatment as well as the importance of being responsible for, and gaining insight into, their own treatment. The interviews with the medical staff revolved around workflow processes and how drug use and pain data were used for guiding pain treatment. The interview schemas used in the interviews are made available in appendices A and B, and the user study is presented in chapter 5.

4.2 Hand-Drawn Mock-Ups

One of the primary ideas that motivated this work was that the design of the Painkiller Software Platform would make it possible to provide patients and medical staff with easily accessible treatment data. Low-fidelity prototypes were used during early design in order to develop an understanding of how users would want the treatment data presented, as per the advice given by Rudd et al. [56]:

> Use low-fidelity prototypes to investigate early concepts about what function the product might have and how it might be presented to the user [...] A low-fidelity prototype gives them [users] some idea of what is possible, providing a starting point for discussion and a target for criticism.

Low-fidelity prototypes are usually made from paper, cardboard, and other office supplies. They are therefore fast to produce and can be put to use before writing any code. Hence, one does not run the risk of having to rewrite a lot of code and thereby also prevents Lauesen's third law of usability from

coming into play [35]: "The more effort developers have spent making a prototype, the less they are willing to replace it."

The author created hand-drawn mock-ups that sketched how drug usage could be visualized in the Painkiller Software Platform. These were shown to end users as part of the interviews described in sect. 4.1. The purpose was to gain an understanding of what kind of visualization would be informative and easily comprehensible. The hand-drawn mock-ups and the loose structure of the interviews proved beneficial as interviewees had suggestions for alterations to the presented visualizations, and these could then easily be incorporated into the final functional prototype (see sect. 4.3).

4.3 Software Prototyping

A functional prototype has been developed with the purpose of exploring and demonstrating the idea of personalizing the PCA experience by making the patient's personal smartphone part of the system. While it may be argued that this could also have been demonstrated using paper mock-ups, a functional prototype provides better basis for evaluating the personalization aspects as it allows users to try out the system on their own personal devices. For example, users will get to experience the physicality of the authentication mechanism (scanning fingerprint on phone and seeing the PCOA device react).

As explained in sect. 4.4, the Painkiller Software Platform is a safety-critical system and hence a lot of attention must be devoted to ensuring that the technical implementation is secure. While a risk analysis (see sect. 4.4) is a great tool for uncovering risks and identifying countermeasures, building an actual implementation, instead of just imagining one, may aid in uncovering low-level security issues that are not immediately apparent.

Finally, the implementation of a functional prototype helps ensure the real world feasibility of the envisioned solution. For example, the prototype can be tested on the chosen hardware to verify if the system can run on resource constrained devices with acceptable response time.

4.4 Risk Analysis

At its core, the Painkiller Software Platform is an IT system that helps a patient self-administer her medication. Its operation hence has a direct impact on the patient's physical wellbeing: if the system malfunctions and dispenses too much medication, the patient may overdose. The Painkiller Software Platform can therefore be characterized as a safety-critical system.

A risk analysis has been carried out in order to identify the safety-related consequences of making the PCOA device a networked device and employing the BYOD scheme in a hospital setting. The outcome of the risk analysis is a list of risks that can direct further initiatives to improve the safety of the Painkiller Software Platform. The risk analysis is presented in ch. 7.

Risk analysis is not a method that is specific to the domain of software development. For example, one could also conduct a risk analysis to understand the risks involved in pursuing a new business opportunity. The risk analysis performed as part of present work follows the recipe given by Basin et al. [5] as it specifically targets IT systems. A summary of this particular framework for risk analysis follows.

First, one defines the IT system under analysis. Depending on the scope of the risk analysis and the stage of the project, this can be anything from a detailed technical description of a functional system to an abstract description of an envisioned system. Important related human-driven processes should

also be included in the description. Second, one identifies the stakeholders, i.e. the set of people who have interest in the system. Third, one identifies the set of assets. An asset is anything that is part of the system and is of value to some stakeholder. For each asset, a state space is defined, and the value of the asset in each of its states is defined for each stakeholder. Fourth, one identifies the set of vulnerabilities. A vulnerability is anything that can cause a change to the state of an asset. The impact of a vulnerability is defined by the change in the value of the asset. Any (planned) countermeasures that may reduce the likelihood or impact of a successful exploit of a vulnerability should also be listed as part of this step. Fifth, one identifies threats by considering who (the threat source) might exploit a vulnerability in what way (the threat action). Finally, one arrives at a final measure for risk by assigning an impact and a likelihood to each threat. The impact is usually directly derived from the impact assigned to the vulnerability that the threat targets. The likelihood is often based on a heuristic evaluation of the capabilities and motivation of the threat source. The resulting risks form a basis for further decision making. For example, one may choose to accept the risk or develop new countermeasures that can reduce its likelihood and/or impact.

5

User Study

This chapter presents the findings from the user study that was carried out at Sjællands Universitetshopsital, Køge. As explained in sect. 4.1, the study made use of semi-structured, qualitative interviews as the purpose of the study was to generate ideas for useful functionality and to develop an understanding of the potential of the Painkiller Software Platform. Section 5.1 presents the findings from interviewing patients. This is followed by the findings from interviews of nurses in sect. 5.2. Conclusions are presented in sect. 5.3.

All interviews were conducted in Danish. As present report is in English, quotations from the interviews have been translated by the author. When doing so, the author has attempted to keep the wording as close to the original responses as possible, but slight modifications have been necessary as direct word-for-word translations would not produce meaningful English sentences. The interviews have not been transcribed. Instead, audio recordings of the interviews accompany this report in its digital form. Table 5.1 presents the format used when referring to the audio files.

Interviewee identifier	Reference format	File name
Patient01	[Patient01 HH:mm:ss-HH:mm:ss]	`patient01.m4a`
Patient02	[Patient02 HH:mm:ss-HH:mm:ss]	`patient02.m4a`
N/A	[Nurse01 HH:mm:ss-HH:mm:ss]	`nurse01.m4a`

Table 5.1: Format used when referring to interview audio files.

5.1 Patients

The patients were selected based on a single criteria, namely that they had firsthand experience with technology-assisted PCA. At the day of the author's visit to Sjællands Universitetshopsital, Køge, two patients in the ward fit this criteria, and they both agreed to be interviewed. Both patients happened to be adult males, one in his forties and one in his fifties. The patients had each used a PCA pump with intravenous analgesics (morphine) for a period of 24 hours after having received total hip replacement. The interviews were conducted shortly after the PCA pumps were taken out of use.

5.1.1 Attitude Towards Technology-Assisted PCA

The patients were asked how they felt about using technology to help them administer their analgesic medication without involving the nurse. Both patients responded positively. Specifically, Patient01 said "I think that's brilliant" [Patient01 00:03:20-00:03:30] and even went on to suggest an idea for the iPhone application, namely that it could include functionality that would allow him to configure automatic doses at specified times [Patient01 00:03:30-00:04:00]. Patient02 responded by saying "I think that is a really, really good thing. I assume that it makes it significantly easier for the medical staff, and I like that I, as a patient, can take my medication immediately when I feel the need for it." [Patient02 00:01:52-00:02:11].

There is obviously less human interaction between the nurse and the patient when a PCA device is used in place of the conventional scheme in which the nurse administers the analgesic medication. For this reason, Patient01[1] was asked if the reduction in interaction with the nurse made the treatment feel impersonal. Patient01 acknowledged that this was true, but he did not consider it problematic, as he knew that the nurses would always have to prioritize their time [Patient01 04:00-06:37]. In fact, he said "I think technology is good. If it can be beneficial, then it should be put to use." [Patient01 00:06:28-00:06:35].

5.1.2 Experiences with the PCA Pump

The patients were questioned about their experiences with the PCA pump. The purpose of this was to gain an understanding of the real-world use context of a PCA device, but also to identify advantages and shortcomings of the PCA pump, such that these could be considered when designing the Painkiller Software Platform.

Despite of facing some problems with nausea, both patients were in general very satisfied with the PCA pump. For example, when asked about whether he had a good or a bad experience using the PCA pump for pain relief, Patient02 responded "It has been super" [Patient02 00:11:15-00:11:26]. Moreover, both patients felt completely safe using the PCA pump. Both patients attributed the problems with nausea to the drug being used, and it was hence not considered a problem intrinsically associated with PCA.

Patient02 described that one of the positive things about using the PCA pump was the ability to take a preemptive dose just before doing something that he knew would be painful. In particular, he said [Patient02 00:12:52-

[1]Unfortunately the author forgot to ask Patient02 this question.

00:13:05]:

> So I see it as a big advantage [...] if you have to do something, for example the first time I had to get out of bed and walk to the restroom, I thought to myself that it would probably be painful [...], so I took one of these [referring to a dose of medication taken using the PCA pump] immediately before doing so.

He also praised the portability of the PCA pump. The ability to bring it with him when he had to go somewhere, for example when he went to take a shower or when he had his x-rays taken, gave him peace of mind as he knew he could achieve immediate pain relief, instead of having to wait until he got back to his bed [Patient02 00:14:13-00:17:01].

Both patients expressed that the handheld, wired remote control was in the way when they were in bed. Patient01 said "There are too many wires for this day and age that we are currently in." [Patient01 00:10:03-00:10:11], and Patient02 said "Then you would avoid having an annoying cord laying around" [Patient02 00:18:25-00:18:28].

5.1.3 Treatment Transparency

One of the primary ideas that motivated this project was that the integration of the patient's smartphone would allow for added treatment transparency as patients could be given access to their treatment data on their personal smartphones. For this reason, the interviews sought to uncover if this was actually something that was of any interest to the patients.

Both patients considered it important to have insight into, and responsibility for, their own treatment and drug use. In particular, when asked

if he considered it important to have insight into own treatment, Patient02 responded "In theory, yes. I would say so. I would like to know what I put in my body." [Patient02 00:06:02-00:06:12]. Additionally, both patients expressed an interest in consuming as little analgesic medication as possible.

The patients were asked if the PCA pumps that they had been using allowed them to access live information about their drug usage, which turned out not to be the case. However, Patient01 did mention that the nurse would, at some point, present him with some post-treatment data. Patient02 explained that the nurse had told him that the PCA pump would play a beep-sound if a dose was granted, when he activated it by clicking the button on its remote control, but that there was no indication that would allow him to know in advance if a dose was available or not [Patient02 00:20:16-00:21:18]. He also made a comment in which he clearly expressed his desire for more information [Patient02 00:03:55-00:04:50]:

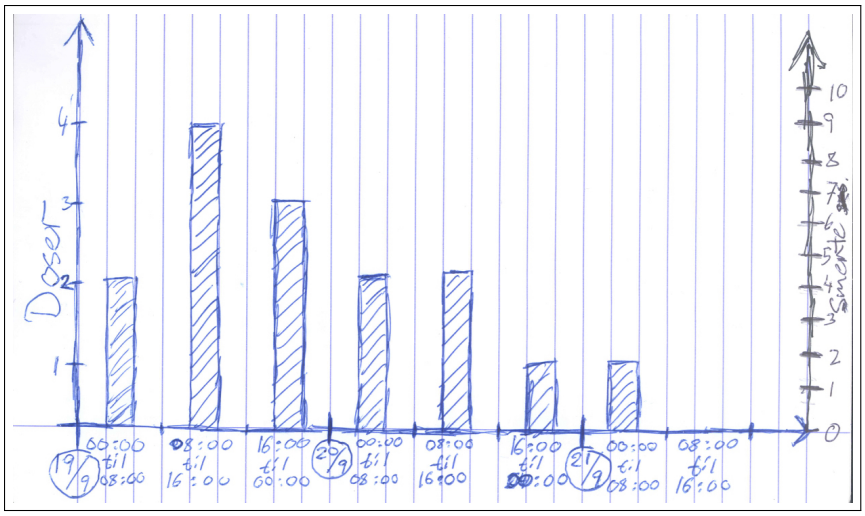
> Here [referring to his use of the PCA pump], I have just pushed a button, but I have no idea how much or how little [medication] I have taken [...] I think it would be nice to be able to track how much [medication] I have taken. [...] And then something I have been missing is that, if the nurse administers the medication, she will let you know that "you had a dose six hours ago, so another two hours must pass [before the next dose]". That information is not available to me here. I have no idea, or at least cannot remember, how many times I have pushed [the button on the remote control of the PCA pump] [...] I would like a log telling me "you have taken a dose at 9 AM, then again at 11 AM, and then already again at 12 PM".

Patient01 explained that he had come to learn that it was optimal for him, in terms of analgesia, if he used the PCA pump roughly once every hour.

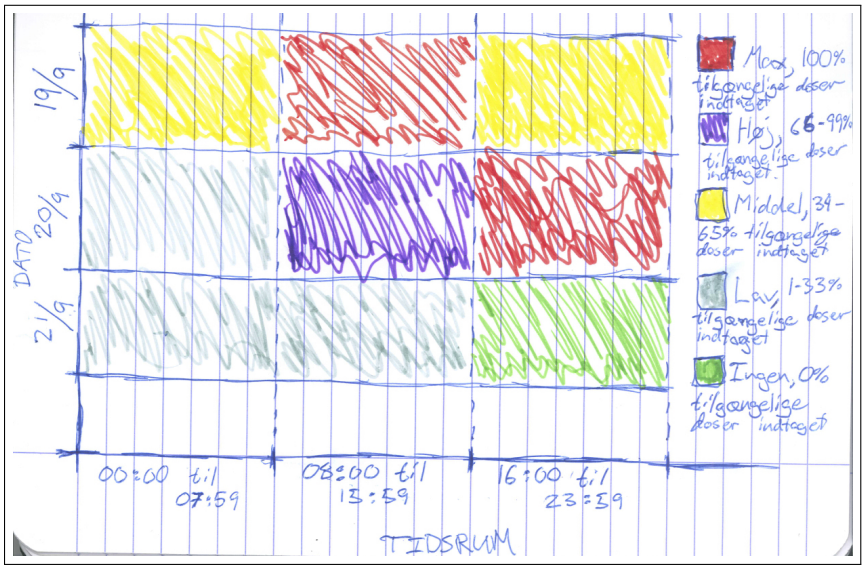
However, as he had no concrete data for his last use of the PCA pump, he did so based on rough estimates. He therefore suggested that it would be beneficial if the system would be able to recommend when to take a dose based on an analysis of his drug usage and pain level [Patient01 00:12:30-00:13:43].

When asked about how he would want to access information about his drug usage, Patient01 responded that he would prefer using his phone because it would be more convenient than having to look at the display of the PCA pump [Patient01 00:14:36-00:15:22]. Patient02 pointed to an info screen hanging from the ceiling at the foot end of his bed, and suggested that it could display information about all medications, including analgesics, that he had been taking [Patient02 00:05:22-00:06:02]. However, later in the interview, he said that it would be "[...] even better if you had an app that you could log onto [...]" [Patient02 00:17:03-00:17:28].

The patients were shown sketches of how their drug usage could be visualized in the smartphone application (see fig. 5.1). As described in sect. 4.2, the purpose was to gain an understanding of what kind of visualization would be informative and easily comprehensible before committing too many resources to implementing one that could potentially later prove inadequate. Both patients had a hard time understanding the color-coded diagram shown in fig. 5.1b. In contrast, the simple bar chart in fig. 5.1a was easily understood, but the information it conveyed was not granular enough. Specifically, Patient02 suggested that it would be more useful to show the actual time and size of each individual dose, rather than displaying the count of doses consumed within a given time interval [Patient02 00:22:06-00:26:02].



(a) Drug usage visualized using a bar chart.



(b) Drug usage visualized using color-codes.

Figure 5.1: Hand drawn sketches of drug usage visualizations.

5.1.4 Security

The patients were asked if they were concerned with the fact that there was nothing preventing a malicious third party from activating the PCA pump on their behalf. Patient01 did not consider this an actual, real-world threat. Patient02 was not concerned with the possibility of such a scenario playing out, but still liked the idea of using the fingerprint scanner to prevent it. He even said that, while it was not something he would attribute a lot of value to, an authentication mechanism would probably be a necessity as just a single incident of the described scenario would cause major headlines in tabloid newspapers and probably result in a ban of the system [Patient02 00:39:57-00:42:12].

5.1.5 Other Observations

Patient02 suggested that a touchscreen-based solution (i.e. tablet or smartphone) would be beneficial as it would allow for easy administration of multiple medications. When presenting this idea, he also added that this would not be as practical on the PCA pump that he had been using as that would imply learning that one click meant medication A, while two clicks meant medication B etc., as the PCA pump only featured a simple push-button [Patient02 00:17:50-00:19:41].

Patient02 presented an interesting social dilemma that he thought the PCA pump helped prevent. He explained that he thought that some patients may opt to postpone asking for more analgesics until the nurse comes to check on them by herself as they do not want to cause inconvenience by calling the nurse to their bedsides [Patient02 00:11:56-00:12:17].

5.2 Nurses

The nurses were selected based on the criteria that they had experience with treating patients' pain using technology-assisted PCA. The purpose of this requirement was to ensure that the interviewees were able to compare technology-assisted PCA with conventional analgesia. Danish nurses have busy workdays with little time for meetings, and arranging interviews therefore proved more difficult than initially anticipated (e.g. one nurse had to reschedule twice due to shortage of staff). Ultimately, the author managed to meet with two nurses in two different nursing wards at Sjællands Universitetshopsital, Køge. Unfortunately, only one of these meetings proved fruitful due to a mishap in communication. It turned out that one of the two nurses only had experience with epidural infusion pumps, which provide a constant infusion of analgesics and hence do not offer patients any kind of control as in PCA. Alas, this interview was discarded, and the following subsections report only on the findings from the interview with the remaining nurse.

5.2.1 Attitude Towards Technology-Assisted PCA

The nurse was generally very pleased with the PCA pumps. For example, she said "I think that the cool thing about these [PCA] pumps [...] is that the patient pushes [the button] and gets help right away." [Nurse01 00:08:03-00:08:09]. She contrasted this by confirming the hypothesis that nurse administered PRN analgesic medication often entails delays, because the nurse gets distracted by other things in-between first consulting the patient about the issue and subsequently returning with the medication [Nurse01 00:10:55-00:11:07]:

> [As part of explaining what happens after leaving the medication

> room] Then I *should* [emphasis added to convey interviewee's tone] return to the patient [with the medication], but the problem is that, when you are a well-known face because you have been working here for many years, you often get interrupted [by other employees on the way back].

The nurse also described that she was under the impression that the added sense of control that the PCA pump gave the patients had a positive effect on their drug usage [Nurse01 00:21:00-00:21:18]:

> I see some benefits in that they [the patients] feel that they possess the right of self-determination. [...] And then perhaps they also consume less [medication] than they would have if they had to ask for more.

When presented with the idea of integrating the patient's personal smartphone with the PCA device, the nurse predicted that this would definitely be hit a among her patients [Nurse01 00:29:14-00:29:32]:

> And again, if they are allowed to use an app, they will get to feel that they have co-determination [...] They would simply love it. They would eat it up. Especially the type of patients that I treat as they are [mentally and physically] well enough [to operate such a system].

5.2.2 Procedure for PRN Analgesic Medication

The nurse confirmed that, if done by the book, the procedure for handing out PRN analgesic medication in Danish hospitals is as follows [Nurse01 00:10:38-00:15:39]. First, the patient calls the nurse to their bedside and informs the

nurse that they are in pain. The nurse decides if the patient should be given a dose of PRN analgesic medication, and if so goes to the medication room to prepare the dose. When doing so, the nurse also updates the patient's digital medical record with information stating that the dose was prepared for, but not yet consumed by, the patient. The nurse then returns to the patient with the medication. However, before the medication is given to the patient, the nurse must first ask the patient for their pain level and add this information to the digital medical record. Finally, once the nurse has observed the patient consume the medication, the nurse must update the information in the patient's digital medical record such that it now states that the dose has been consumed by the patient. Then nurse then leaves the patient, but must return 30 minutes later to question the patient about their pain level and add this information to the patient's digital medical record.

However, the interview also revealed that the prescribed procedure is rarely followed down to the last detail. The nurse described that she would "cheat" by logging the consumption of the medication already in the medication room in order to avoid having to waste time logging into two different computers [Nurse01 00:13:18-00:14:29]. It also became clear that the follow-up pain scores were often forgotten. In particular, the nurse said [Nurse01 00:15:39-00:15:49]:

> [...] I never manage to do it half an hour after [medicating the patient], because if the patient does not call for me, the problem must have been solved. So I never remember that I have to do it half an hour after [medicating the patient].

5.2.3 Intravenous vs. Oral Medication

The loose structure of the interview proved beneficial as the nurse unveiled some information about the use of intravenous and oral medication in Danish hospitals that the author had not planned to ask about. The nurse explained that the intravenous medication used in the PCA pump is beneficial because the patient feels its effect very quickly, but unfortunately the duration of the effect is also short, and the patient's pain level may therefore fluctuate a lot [Nurse01 00:08:41-00:08:55]. In contrast, tablet medication work longer and can therefore help maintain a more constant pain level. The nurse also described that the patient would feel tethered to the bed when using the PCA pump – as opposed to tablet medication – because of the tube attached to their body [Nurse01 00:19:06-00:20:08]. Furthermore, the nurse explained that the Danish hospital culture actually favors tablets over intravenous medication [Nurse01 00:47:36-00:48:45].

It became clear from the conversation that the nurse already practiced PCOA, albeit without any technological aids. She explained that she would leave some extra fast-acting tablets[2] at the patient's bedside so that the patient would have them available when needed [Nurse01 00:09:05-00:09:33]. She made it clear that she would obviously only do so when she was confident that the patients were capable of safely managing the medication on their own. However, there is no log of when the patient actually decides to take the medication, and hence the nurse can never be completely sure of when it is safe to hand out the next dose when using this manual approach to PCOA. When asked why the hospital did not employ technological aids for PCOA, the nurse responded that she was not aware of the existence of such technology [Nurse01 00:46:18-00:46:30], but that it would definitely be useful [Nurse01 00:48:45-00:49:18].

[2]It was not clear if the nurse referred to tablets or capsules. Nevertheless, both kinds are administered through the oral route.

(a) UI displaying hourly totals of doses requested and doses granted.

(b) Bar chart displaying doses granted throughout the last nine hours.

Figure 5.2: The PCA pump used at Sjællands Universitetshopsital, Køge.

5.2.4 Treatment Data

The nurse was asked what treatment data the PCA pump made available to her in order to uncover if this kind of functionality was already available in commercial products, and how it was (or could be) used for improving the pain treatment.

The nurse explained that the PCA pump only provided totals for the number of doses requested (i.e. the number of times the patient had attempted to activate the PCA pump) and the number of doses granted throughout the entire treatment period [Nurse01 00:26:10-00:26:52]. However, this statement turned out to be inaccurate, as a medical student later showed the author the PCA pump and, as part of doing so, discovered that the pump could also display hourly totals (see fig. 5.2).

The nurse reacted positively when she was presented with the idea that the PCA device could be programmed to provide more granular treatment data [Nurse01 00:26:30-00:28:20]. In particular, she saw an opportunity in looking for similarities in treatment data from multiple patients as such similarities

could possibly indicate specific periods after surgery in which the patient's pain would escalate.

When asked about how she wanted to access the drug usage information, the nurse initially responded that she already had confusingly many monitors, so she would prefer to access the information directly on the PCA device [Nurse01 00:30:12-00:33:30]. She later came to realize that – if the data was to be used as guidance *during* treatment – it would require that she would interact with the PCA device more often than what she currently did (she would normally only operate it when setting it up and when putting it away at the end of the treatment). However, she did not consider such a change to her workflow an issue [Nurse01 00:33:21-00:33:28]: "It would require a new workflow [...] But we are about to become a university hospital, so we might as well just get started."

The nurse was also asked for her opinion on the sketches of how drug usage could be visualized in the system (fig. 5.1). It was not clear if the nurse preferred one visualization over the other, but she mentioned that she liked the use of color-codes in fig. 5.1b because color-codes were already widely used in other aspects of her daily work [Nurse01 00:33:35-00:35:55].

5.2.5 Stability

The nurse made a comment in which she clearly stated her opinion on technological aids that malfunction or are hard to use [Nurse01 00:42:06-00:42:22]:

> It *has* to work [flawlessly] otherwise I will throw it out the window [...] I am no IT [technician]. I become infuriated when things [technological aids] don't work.

5.3 Conclusion

While only a few patients and nurses were interviewed, there seem to be consensus among the two parties that technology-assisted PCA is beneficial. However, patients think that PCA technology should be modernized to match the technological standards of the 21st century. In particular, patients welcome the idea of controlling the PCA device from their personal smartphones as it is more convenient – for example because it gets rid of annoying wires.

Patients consider insight into, and a sense of responsibility for, own treatment an important aspect. They think that treatment data should be made available to them directly on their personal smartphones as it would be the most convenient way to access the information. In addition, nurses see opportunities in the availability of more granular treatment data, especially when aggregated across multiple patients. Specifically, such data could possibly help identity common post-operative pain patterns; e.g. pain escalating at a certain time after surgery.

The initial hand-drawn sketches of medication consumption charts were found to be inadequate as they did not provide a sufficient level of detail. It was suggested that the charts should display the precise time and size of each dose rather than displaying the number of doses consumed within a given timeframe. This suggestion was incorporated into the final design of the drug usage visualization found in the functional prototype of the Painkiller Software Platform.

The procedure for nurse administered PRN analgesic medication is cumbersome. Important steps are often omitted as a result of forgetfulness and/or conscious circumventions in the name of productivity. In particular, follow-up pain assessments are often forgotten. The Painkiller Software

Platform may help remedy exactly this problem as patients are automatically prompted to reassess their pain.

As shown in ch. 3, the availability of technological aids for PCOA is limited, and this was reflected in the user study as nurses were not aware of the existence of such equipment. This is an unfortunate situation since Danish hospital culture favors tablet medication over intravenous medication, and since Danish nurses already practice PCOA despite the lack of technological aids. Nurses therefore think that PCOA equipment would be a helpful tool to have at their disposal.

In summary, while limited in size, the user study has shown that the Painkiller Software Platform has potential for becoming a useful tool in Danish hospitals because it can help increase the safety of PCOA, but also because it can help reduce the workload that is imposed on nurses when PCOA is manually practiced. In addition, the system can supply a greater level of treatment transparency, which is something that is in high demand among patients.

6

Implementation

This chapter delves into the technical details of the implementation of the Painkiller Software Platform. Section 6.1 provides a high level view of the system by describing the hardware setup and the software architecture. Painkiller Server, the application that runs on the PCOA device, defines the data model of the system and is responsible for enforcing the business logic. It is therefore described in detail in sect. 6.2. Section 6.3 explains the security aspects of the implementation, including how user authentication and user authorization has been implemented and how the procedure that pairs a patient's iPhone with the PCOA device works.

6.1 System Architecture

6.1.1 Hardware

The hardware setup required to run the Painkiller Software Platform consists of three computing devices: a Raspberry Pi 2 Model B, an iPhone 5S or later, and an iPad Mini 3 or later. Additionally, a wireless network infrastructure is required as the three devices must be able to communicate over the local network. A firewall shields the network from inbound Internet traffic. Both

the iPhone and the iPad communicate with the Raspberry Pi. There is no direct communication between the iPad and the iPhone. Figure 6.1 visualizes the hardware setup and the communication links.

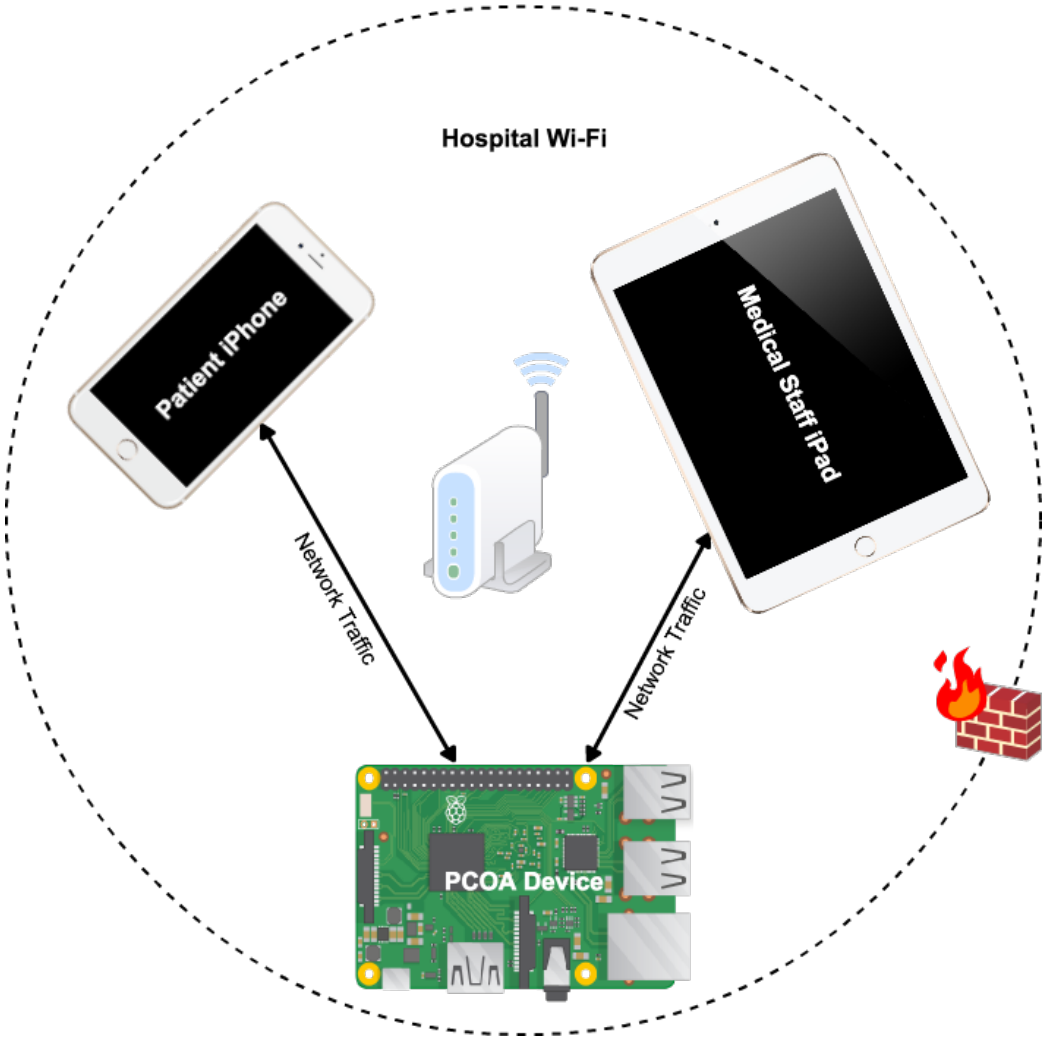
The Raspberry Pi is the microcontroller that is built into the PCOA device. Its operating system is Raspbian OS, and it is fitted with a small display and a Wi-Fi adapter. It is connected to the hospital's wireless network and is assigned a static IP. The static IP is a necessity as the procedure that connects the patient's iPhone to the PCOA device (see sect. 6.3.2) entails an assumption, namely that the IP of the PCOA device does not change after the procedure has concluded. The Raspberry Pi was chosen because it complies with the requirements for a PCOA device identified in the user study in sect. 5.1. The user study revealed that the PCOA device must be portable and unobtrusive. The Raspberry Pi matches this specification as it is small (about the size of a credit-card) and consumes little power, so it may run temporarily on a battery.

The iPhone is the patient's private device. It runs iOS version 10 or later and is connected to the hospital's wireless network. It may be assigned either a static or a dynamic IP. The Painkiller Software Platform depends on the availability of a fingerprint scanner (Touch ID), and hence the iPhone must be a model 5S or later.

The iPad is the medical staff member's device. Each staff member has his/her own iPad. The iPads run iOS version 10 or later and are connected to the hospital's wireless network. They may be assigned either a static or a dynamic IP. The Painkiller Software Platform depends on the availability of a fingerprint scanner (Touch ID), and hence the iPads must be a model Mini 3 or later.

While the iPhone and the iPad offer no special functionality that cannot be found in Android (or Windows) counterparts, they were chosen for the

Figure 6.1: Hardware setup of the Painkiller Software Platform. Image attributions: `www.freeiconspng.com` (iPhone and iPad) and `www.raspberrypi.org` (Raspberry Pi).



prototype implementation as they provide a better basis for an evaluation of the system in which users use their own personal devices as opposed to provided devices.

This conclusion was drawn based on the following data. First and foremost, iPhones dominate the Danish smartphone market as evidenced by a survey conducted by Danske Medier Research in 2015 [11], so the likelihood that a potential test subject will own an iPhone is high. Second, a potential test subject's smartphone must be equipped with a fingerprint scanner as the system is designed around authenticating the user by virtue of their fingerprint. Touch ID, Apple's name for its fingerprint scanner, first appeared on the iPhone 5S, which was released in 2013, and has been present on all later models. In contrast, Apple's main competitor in the smartphone market, Samsung, did not introduce a *reliable* fingerprint scanner in their smartphones until the Galaxy S6 which was released in 2015[1]. This suggests that the availability of fingerprint scanners in smartphones currently in the wild will be greater for iPhones than for their Android counterparts.

A secondary driving force behind choosing iOS devices over Android devices was the author's personal desire to learn how to program in Swift[2] and how to use the iOS SDK.

[1]The Galaxy S5, released in 2014, was equipped with a fingerprint scanner, but the scanner was criticized for being unreliable and clunky and was hence not of much use.

[2]Swift [23] is an Object-Oriented programming language that was originally a proprietary technology of Apple Inc., but has recently been turned into an open-source project. Swift is a cross-platform language, and it includes features often found in modern programming languages such as support for functional programming patterns and a concise syntax.

6.1.2 Software

The Painkiller Software Platform consists of three software applications: Painkiller Server, which runs on the Raspberry Pi, Painkiller Patient, which runs on the patient's iPhone, and Painkiller Staff, which runs on the medical staff member's iPad. Figure 6.2 depicts the modules of each application and how the three applications interact.

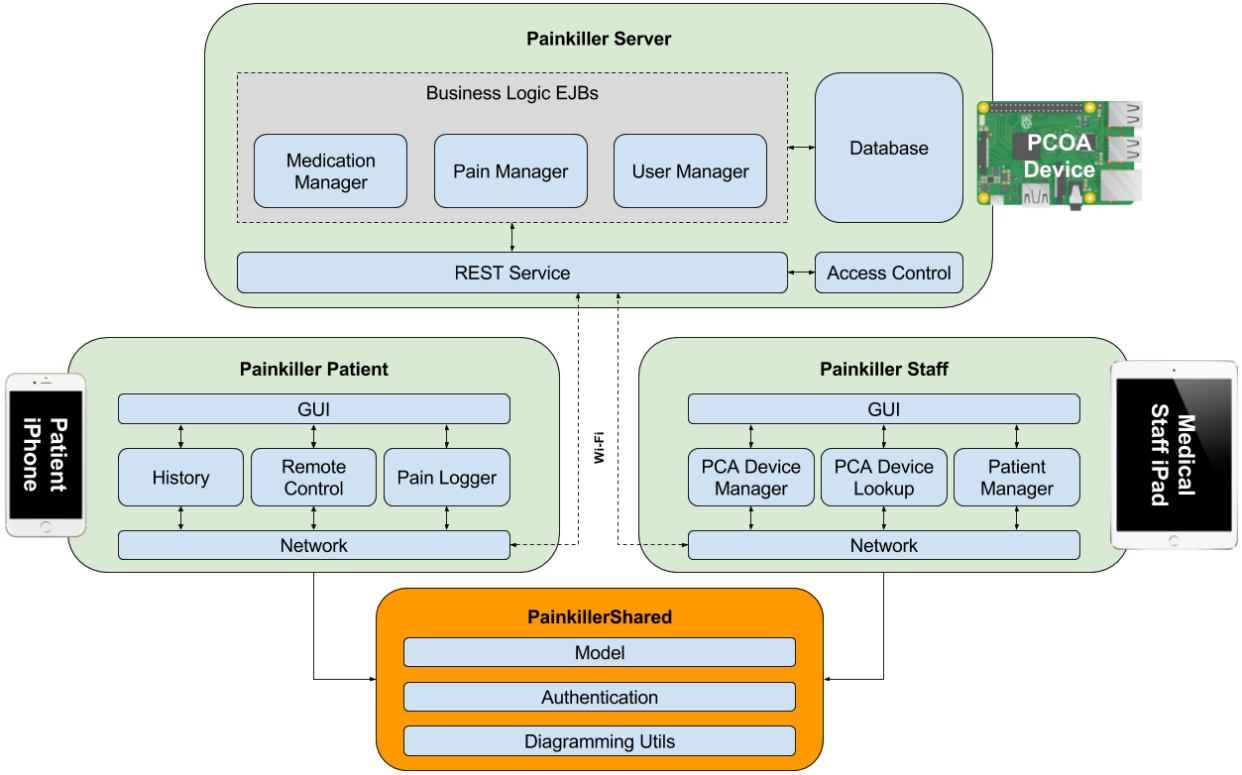
6.1.2.1 Painkiller Server

Painkiller Server controls the physical behavior of the PCOA device. It is a RESTful Web Service written in Java and deployed in a GlassFish container (Open Source Edition, version 4.1). Its Access Control module is a token-based solution that restricts access to the REST endpoints to a set of authorized users. It is explained in detail in sect. 6.3.1. For now, simply note that it is completely independent of the backing database. The business logic is implemented in a set of Enterprise Java Beans (EJB). The Medication Manager is responsible for deciding if a dose of medication should be dispensed upon request. The decision logic is explained in detail in sect. 6.2.5. The Pain Manager is responsible for managing a patient's pain registrations, and the User Manager is in charge of managing the set of users that can operate the PCOA device.

6.1.2.2 Painkiller Patient

Painkiller Patient is a native iOS application, written in Swift, which runs on the patient's iPhone. It assumes the role of the patient's interface for the Painkiller Software Platform. Its History module is responsible for obtaining and presenting the patient's historic drug usage and pain level data

Figure 6.2: Architecture of the Painkiller Software Platform. Green boxes are executable applications. Orange boxes are libraries/frameworks. Hardware is included in the diagram in order to visualize where each applications runs. Image attributions: `www.freeiconspng.com` (iPhone and iPad) and `www.raspberrypi.org` (Raspberry Pi).



in a meaningful manner. The Remote Control module is the functionality that activates the dispensing mechanism of the PCOA device by invoking a protected REST endpoint of Painkiller Server. The Pain Logger module is responsible for prompting the user for their pain level at the right times and sending the pain level data to Painkiller Server.

6.1.2.3 Painkiller Staff

Written in Swift, Painkiller Staff is a native iOS application which runs on the medical staff member's iPad. It serves as the medical staff member's interface for the Painkiller Software Platform. Painkiller Staff targets both nurses and doctors which is why the term medical staff member was used. However, in the following, nurse and medical staff member will be used interchangeably since use of the former often enhances readability.

To facilitate deployment of multiple PCOA devices in a single hospital, while still allowing the nurse to manage all of them from a single iPad, a mechanism that allows the nurse to choose a specific PCOA device from a list of available PCOA devices must be in place. This functionality is provided by the PCA Device Lookup module. It is assumed that the hospital's IT department runs a central server that the PCA Device Lookup module can query to get a list of the hospital's PCOA devices. Each list entry should contain a call name for the PCOA device and its IP address.

The PCA Device Manager module provides functionality that allows the medical staff member to control the set of users associated with a specific PCOA device. This includes functionality for creating new users and for initiating the pairing procedure (see sect. 6.3.2) that lets a patient connect their personal iPhone to the PCOA device.

The Patient Manager module contains the functionality that allows the

medical staff member to configure a patient's prescriptions and view the patient's drug usage and pain history.

6.1.2.4 PainkillerShared

Painkiller Patient and Painkiller Staff share a great deal of code as they are both iOS clients of the same server application. In order to avoid code duplication, this shared functionality has been extracted to a Cocoa Touch Framework, PainkillerShared, which is also written in Swift.

The Model module defines a Swift representation of the data model of Painkiller Server (see sect. 6.2.2), including code for marshalling and unmarshalling data exchanged with Painkiller Server. The Diagramming Utils module of PainkillerShared contains tools for generating drug use and pain history diagrams. Finally, the Authentication module provides functionality for verifying the user's identity by scanning her fingerprint. It is used by Painkiller Patient and Painkiller Staff to ensure that no one but the patient and the nurse can use their respective devices for operating the PCOA device.

6.2 Painkiller Server: A Detailed Look

The Painkiller Software Platform adopts a server-centered design: all data is stored on the PCOA device, and Painkiller Server is responsible for enforcing the business logic (e.g. deciding if a dose of medication can be dispensed). Painkiller Patient and Painkiller Staff can be thought of as thin clients as they serve as user interfaces for operating the PCOA device by invoking operations of Painkiller Server. Given its role as the central component and the brain of the Painkiller Software Platform, Painkiller Server is also the most interesting from a technical point of view and will therefore be discussed in more detail

in this section.

6.2.1 A Flexible Design

Painkiller Server uses Java EE technologies to abstract away low-level details. This increases development speed[3] and confidence in the correctness of the implementation as these building blocks have been thoroughly tested. Specifically, the code for exposing the REST endpoints is written using the JAX-RS API version 2.0 [48], and database interaction is built using the Java Persistence API (JPA) version 2.1 [17].

However, these APIs are merely *specifications*. A specification is a formal document that specifies *what* the API must provide and do, but not *how* it should do it. Hence, a specification does not provide the actual functionality (i.e. executable code) – that part is the responsibility of the vendor who provides an *implementation* of the API – but it does provide the user of the API with a set of guarantees that must hold for *all* implementations of the API. The implementation, on the other hand, is the executable code that provides the functionality defined by the specification.

The Painkiller Server prototype uses Jersey[4] as its JAX-RS implementation and EclipseLink[5] as its JPA implementation. These two implementations offer extra functionality beyond what is specified in the respective specifications, but care has been taken *not* to use this extra functionality.

[3]The author had to learn how to use the technologies from scratch, so there was a learning curve to overcome. However, the seasoned Java EE developer should experience a significant gain in terms of development speed.

[4]Jersey is a framework for developing RESTful Web Services in Java. It is developed and maintained by the Oracle Corporation and serves as the JAX-RS reference implementation.

[5]EclipseLink is a framework that simplifies the process of writing Java code that interacts with data services, including databases. It is developed and maintained by the Eclipse Foundation, and it includes an implementation of the JPA specification.

In other words: the source code only references classes, interfaces, and annotations defined by the two specifications. This makes Painkiller Server implementation agnostic. Jersey can be replaced by any other JAX-RS implementation, e.g. RESTEasy from JBoss, and EclipseLink can be replaced by any other JPA implementation, e.g. Hibernate, without changing the source code.

This kind of pluggability is important from a security standpoint as vulnerabilities lower down the software stack propagates to Painkiller Server as a whole. If, for example, a critical security issue is discovered in Jersey, the problem can easily be remedied by (temporarily) replacing Jersey with RESTEasy. This is revisited in sect. 7.3.

6.2.2 Data Model

The data model of Painkiller Server has been designed using the Java Persistence API (JPA). JPA is an Object/Relational Mapping (ORM) framework for Java. JPA allows the data model and database queries to be defined purely in code. No SQL has been written. As a result, Painkiller Server is fully database agnostic. The set of possible relational databases is only constrained by what SQL dialects the chosen JPA provider supports, and, as explained in sect. 6.2.1, the JPA provider can easily be replaced in case it does not support the chosen database. The following subsections briefly describes the defined JPA entities and their relationships.

6.2.2.1 User

The `User` entity models the individuals who use the Painkiller Software Platform. It contains basic personal information such as name and CPR number

and a role field that identifies if the user is a patient or a medical staff member. It also contains a collection of `Prescription` instances. This collection is the model's representation of the medication that has been prescribed to the user.

6.2.2.2 Prescription

The `Prescription` entity contains all information relating to a prescription for analgesic medication. This includes the name of the drug and the dose size, i.e. the number of tablets in each dose. It also contains information about how long time must pass between each dose (the lockout period) and the maximum number of doses the patient may consume every 24 hours (referred to as the daily dose cap). Each `Prescription` also carries a collection of `Dose`s. This collection models the doses that have been dispensed as part of the prescription. Finally, there is also a reference back to a `User` entity, which is the patient to whom the prescription was issued.

6.2.2.3 Dose

The `Dose` entity is simplistic. Its only data members are a timestamp, which indicates when the dose was dispensed, and reference back to the `Prescription` instance under which the `Dose` was granted. A `Dose` is meaningless on its own as it contains no information about the medication – it only exists in the context of a `Prescription`.

6.2.2.4 PainRecord

The `PainRecord` entity is Painkiller Server's notion of a pain registration submitted by a patient. It consists of an NRS-11[6] pain level, a timestamp that ties the pain level to an instant in time, and a reference to a `User` entity, which represents the patient who reported the pain.

6.2.3 URI Design

The URI design of Painkiller Server makes use of a hierarchical structure that reflects the design of the data model described in sect. 6.2.2. For example, to retrieve a specific user's list of prescriptions, one first locates the user by appending `/users/{userId}` (where `{userId}` indicates an integer value) to the base URI and then drills down to its associated prescriptions by appending `/prescriptions` to this URI. Assuming a base URI of `http://painkiller.dk`, the complete URI for the list of prescriptions of the user with id 42 then becomes:

`http://painkiller.dk/users/42/prescriptions`

Furthermore, Painkiller Server obeys the REST architectural constraints [14] as HTTP methods are used in accordance with their definitions [13]: `GET` is used for resource retrieval, and `POST` is used for resource creation. The full set of REST Endpoints exposed by Painkiller Server is shown in table 6.1.

[6]The Numeric Rating Scale (NRS-11) is an 11-point scale that is widely used for patient self-reported pain. The scale ranges from 0 (no pain) to 10 (unbearable pain).

URI pattern	Method	Operation
/users	GET	Return all users.
/users	POST	Create a new user.
/users?role={role}	GET	Return all users in a specific role.
/users/{uId}/painrecords	GET	Return a user's pain records.
/users/{uId}/painrecords	POST	Add a new pain record for user uId.
/users/{uId}/painrecords/newest	GET	Return a user's most recent pain record.
/users/{uId}/painrecords?from={d1}&to={d2}	GET	Return those of a user's pain records which were registered between dates d1 and d2.
/users/{uId}/prescriptions	GET	Return a user's prescriptions.
/users/{uId}/prescriptions	POST	Add a new prescription to a user's collection of prescriptions.
/users/{uId}/prescriptions/{pId}	GET	Return a specific prescription from a user's collection of prescriptions.
/users/{uId}/prescriptions/{pId}/doses	GET	Return all doses dispensed as part of a specific prescription.
/users/{uId}/prescriptions/{pId}/doses	POST	Dispense a dose.
/users/{uId}/prescriptions/{pId}/doses/{dId}	GET	Return a specific dose.
/pairing/await/users/{uId}	POST	Put PCOA device in pairing mode (let patient connect).
/pairing	POST	Perform patient login.

Table 6.1: REST Endpoints of Painkiller Server. URI fragments enclosed in curly braces indicate variable parts. uId is short for user id, pId for prescription id, and dId for dose id. Integer values are expected for these three variables.

6.2.4 Extensibility

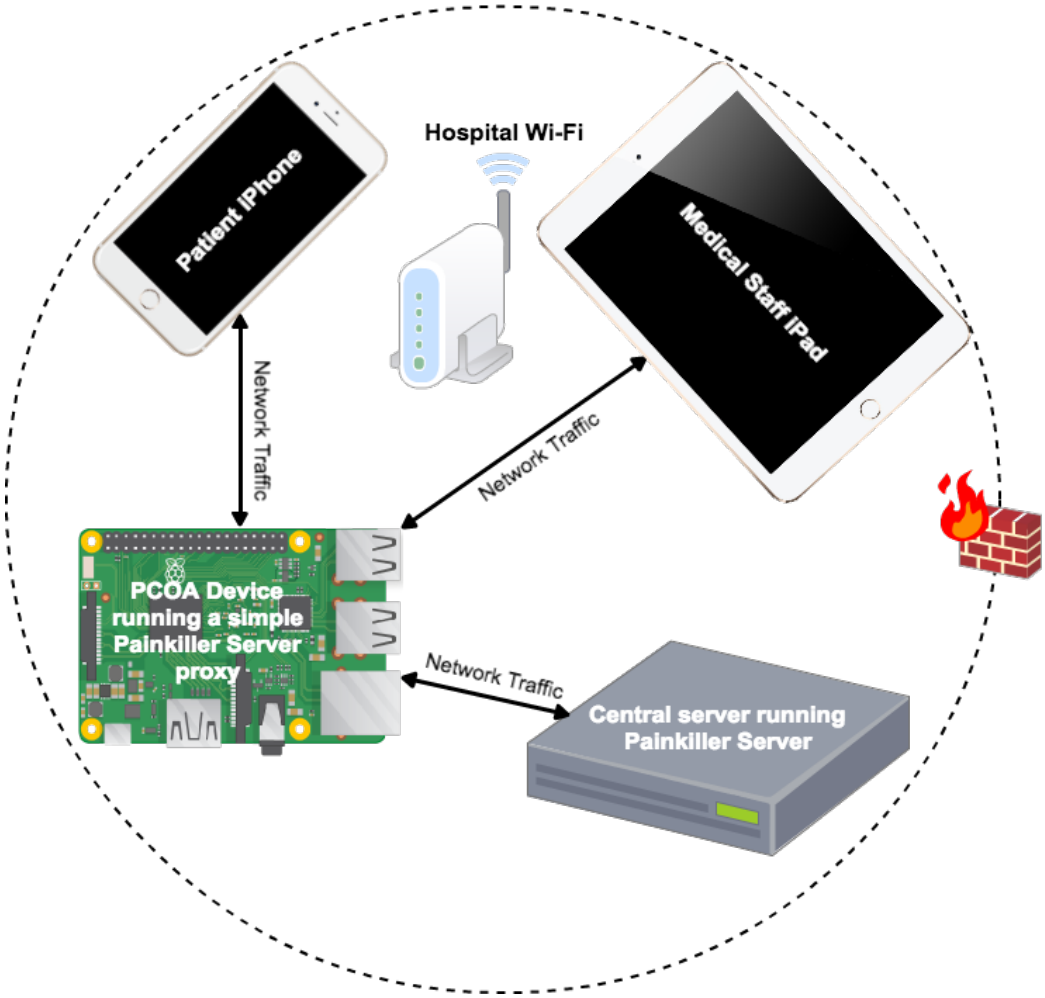
The current Painkiller Software Platform prototype assumes that it is used by a single patient at a time (i.e. one patient per PCOA device). However, the hierarchical data model (sect. 6.2.2) and URI design (sect. 6.2.3) of Painkiller Server actually support multiple patients at a time. The Painkiller Software Platform could therefore relatively easily be modified to support a different hardware setup. For example, in order to improve data security and/or redundancy, one could centralize all patient data and dispensing decision logic on a single powerful server instead of maintaining a local database on each PCOA device. With small modifications, Painkiller Server could run on the central server. In this setup, each PCOA device would become a simple proxy responsible for relaying information between the client devices and the central server and for triggering the physical dispensing mechanism based on decisions received from the central Painkiller Server. The example is visualized in fig. 6.3.

The hierarchical data model and URI design have also made it possible to include functionality that allows a patient to manage multiple medications concurrently (as shown in fig. 2.12b) – something that was suggested by a patient in the user study (see sect. 5.1.5).

6.2.5 Dispensing Algorithm

The single most important piece of logic in Painkiller Server is the algorithm responsible for dispensing a dose of medication. Before a dose can be dispensed, the system must first verify that a dose of medication is available. This part of the code is shown in listing 6.1 and is relatively straight forward. First, the full set of doses dispensed as part of the specified prescription is filtered, creating a list, named `recentDoses`, consisting of the doses that were

Figure 6.3: Example of alternative hardware setup of the Painkiller Software Platform. Image attributions: `www.freeiconspng.com` (iPhone and iPad) and `www.raspberrypi.org` (Raspberry Pi).

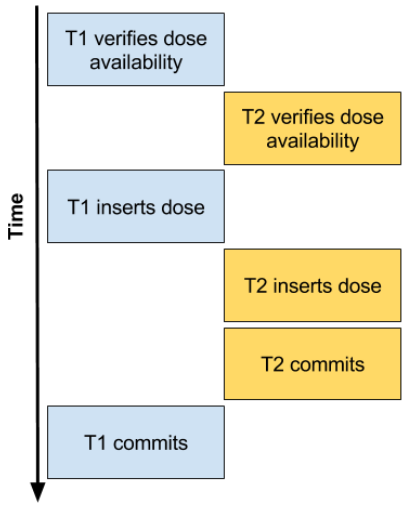


dispensed within the last 24 hours. The number of doses in `recentDoses` is then compared against the daily dose cap of the prescription to check if the patient has already used up her daily medication quota. The code then proceeds to find the most recent dose and decides if a dose is available based on whether or not a duration equal to the lockout period has passed since the dose was dispensed.

Listing 6.1: Dispensing a dose: checking if a dose is available.

```
@Stateless
public class DoseService { // ... some fields and other functionality omitted
   @TransactionAttribute(value = TransactionAttributeType.MANDATORY)
   private boolean isDoseAvailable(Prescription p) {
       ArrayList<Dose> recentDoses = new ArrayList<>(); // Doses for the last 24 hours.
       for(Dose dose : p.getDoses()) {
           if (isInDurationBeforeNow(Duration.ofHours(24), dose.getDate())) {
               recentDoses.add(dose);
           }
       }
       // Daily dose cap reached?
       if (recentDoses.size() >= p.getDailyDoseCap()) { return false; }
       // A dose is available if there are no recent doses.
       if (recentDoses.size() == 0) { return true; }
       Dose mostRecentDose = recentDoses.get(0);
       for(Dose dose : recentDoses) { // Search for the most recent dose...
           if(dose.getDate().after(mostRecentDose.getDate())) {
               mostRecentDose = dose;
           }
       }
       // Check if enough time has passed since the most recent dose was dispensed.
       return !isInDurationBeforeNow(Duration.ofMillis(p.getLockoutIntervalMillis()),
            mostRecentDose.getDate());
   }
   // Checks if a date lies within a specified duration before the current instant in time.
   private static boolean isInDurationBeforeNow(Duration duration, Date date) {
       Instant now = Instant.now();
       Instant input = date.toInstant();
       Instant nowMinusDuration = now.minus(duration);
       return !input.isBefore(nowMinusDuration) && input.isBefore(now);
   }
}
```

Figure 6.4: Example of unlucky ordering of operations of two concurrently executing dispense dose requests wrapped in EJB transactions.



The logic for deciding if a dose can be dispensed is only the first step towards dispensing a dose. It must be combined with the insertion of a new `Dose` instance into the database such that subsequent requests will take the new dose into account. As the full operation requires two separate database queries to be executed sequentially, care must be taken to avoid the check-then-act pitfall (also known as the non-repeatable read anomaly) illustrated in figure 6.4.

The code that performs the database insertion is shown in listing 6.2. EJB transactions alone are insufficient as they are unable to detect the change to the `Prescription` entity as its corresponding database row is not changed when a new `Dose` is added to it. This stems from the fact that the `Dose` is the owning side of the relationship between itself and its `Prescription`. In the database, the row that represents the `Dose` will have a foreign key pointing to the `Prescription`, but the row that represents the `Prescription` will have no pointers to any of its `Dose`s. The problem is remedied by guarding the `Prescription` with an optimistic write lock that forces an update to the

`Prescription`'s row when a new `Dose` is inserted, as explained by Keith and Schincariol [31]:

> The write lock [...] pledges to increment the version field in the transaction regardless of whether a user updated the entity or not. [...] the common case for using `OPTIMISTIC_FORCE_INCREMENT` is to guarantee consistency across entity relationship changes (often they are one-to-many relationships with target foreign keys) when in the object model the entity relationship pointers change, but in the data model no columns in the entity table change.

Listing 6.2: Dispensing a dose: database insertion.

```
@Stateless
public class DoseService { // ...some functionality omitted
    @PersistenceContext(unitName = "PrescriptionUnit")
    private EntityManager entityMgr;
    @TransactionAttribute(value = TransactionAttributeType.REQUIRED)
    public Dose addDose(long prescriptionId) {
        Prescription p = entityMgr.find(Prescription.class, prescriptionId);
        if (p == null) { // Invalid prescription id
            throw new IllegalArgumentException(/* error message omitted */);
        }
        entityMgr.lock(p, LockModeType.OPTIMISTIC_FORCE_INCREMENT);
        Dose d = null;
        if (isDoseAvailable(p)) { // A dose is available, create it and insert it.
            d = new Dose();
            d.setDate(new Date());
            d.setPrescription(p);
            p.getDoses().add(d);
        }
        try { // Flush changes to database and return new dose.
            entityMgr.flush();
            return d;
        } catch (OptimisticLockException ole) { // Rethrow application-managed exception.
            throw new ChangeCollisionException();
        }
    }
}
```

6.3 Security

6.3.1 Authentication and Authorization

In order to ensure that it is only the patient who can trigger the dispensing mechanism, Painkiller Server must implement a mechanism for authenticating the user. However, the patient is not the only user. The medical staff members must also have exclusive access to the configuration operations exposed by Painkiller Server. As users have different rights, Painkiller Server must therefore also implement a mechanism for authorization.

6.3.1.1 Token Content

User authentication and authorization is achieved through use of JSON Web Tokens (JWT), which is an open standard [28] for transmitting tamper-proof information (encoded in a JSON object) between two parties. A JWT consists of a header, a payload (referred to as a set of *claims*), and a JSON Web Signature (JWS) [27]. The header contains metadata such as the token type (optional) and the signature algorithm. The claims in the payload section are key-value pairs that carry the information that the two parties want to exchange. The JWS section is a digital signature created by providing the header section, the payload section, and a secret key[7] as input to the signature algorithm specified in the header. As the signature is based on the content of the token, modification of the content will be detected when the verifying entity reconstructs the signature and compares it with the signature bundled in the JWT.

Painkiller Server expects all incoming requests to operations that require

[7] JWTs are flexible: they can be symmetrically signed using a shared secret key, but can also be signed using a public/private key pair in the form of an X.509 certificate.

authentication and authorization to carry a JWT issued by Painkiller Server itself in the `Authorization` field of the HTTP header. Patients obtain a JWT through a pairing procedure which will be described in sect. 6.3.2. It is assumed that the hospital's IT department will generate and install the medical staff members' JWTs when setting up and handing out the iPads.

An example of a JWT issued by Painkiller Server is shown in listing 6.3. The payload section contains two claims, namely `sub` (short for subject) and `role`. The value of the `sub` claim is an identifier for the user to whom the JWT was issued. The value of the `role` claim specifies the role of the user to whom the JWT was issued. There are two possible user roles: `PATIENT` and `MEDICAL_STAFF`. The value of the `alg` claim in the header section specifies that the signature is generated and verified using the HMAC SHA-512 algorithm.

Listing 6.3: Example of a JWT issued by Painkiller Server (formatting added for clarity). The first block is the header section. The second block is the payload section. The final part is the digital signature.

```
{
    "alg":"HS512"
}
{
    "sub":"4",
    "role":"MEDICAL_STAFF"
}
< ... signature blob ... >
```

The JWTs issued by Painkiller Server are self-contained. They carry all the information (user identifier and user role) that is required when determining if a request to a restricted resource should be allowed. Painkiller Server can therefore avoid querying its database for said information. The use of JWTs also ensures that Painkiller Server adheres to the RESTful design

System property	Example usage	Description
`keystorepath`	`-Dkeystorepath=/etc/ps/ks.jks`	The path to the keystore.
`keystorepassword`	`-Dkeystorepassword=unicorn42`	The password for the keystore.
`friendlyname`	`-Dfriendlyname=Dispenser911`	A call name for the PCOA device. Helps users distinguish it from other PCOA devices.
`java.awt.headless`	`-Djava.awt.headless=false`	Specifies that Painkiller Server should be allowed to present a user interface. Must be `false`.

Table 6.2: Required system properties of Painkiller Server.

principles as it does *not* need to retain copies of the issued tokens for token verification (as is the case in a basic token-based authentication scheme). A token is verified using only its signature and the secret key of Painkiller Server. Refraining from persisting copies of tokens on the server is also beneficial from a security point of view: if the database is compromised, the adversary does not gain access to valid tokens that could be used to act on behalf of the individuals to whom they were issued.

6.3.1.2 Secret Keys

Painkiller Server uses 512 bit secret keys to generate and verify the signatures present in the JWTs. The keys are stored in a password protected Java KeyStore. In order for Painkiller Server to be able to load the keys from the keystore file, a set of system properties must be set when Painkiller Server launches. These are listed in table 6.2.[8]

Painkiller Server expects that the keystore contains two distinct keys, and that these are identified by two predefined aliases. One key, identified by

[8]While only the first two system properties are relevant to the discussion here, all required system properties are listed for the sake of completeness.

the alias `signaturekey_staff`, is used for verifying JWTs issued to medical staff members. A second key, identified by the alias `signaturekey_patient`, is used for signing and verifying the JWTs that are issued to patients. The use of different keys allows Painkiller Server to invalidate all patients' JWTs without affecting the validity of the medical staff members' JWTs and vice versa. This will be returned to in sect. 6.3.2.

6.3.1.3 Access Configuration and Token Verification

Painkiller Server defines an annotation, called `AccessControlRequired` (see listing 6.4), that is applied to all JAX-RS resource methods (REST endpoints) that require authentication and, optionally, authorization. The annotation defines an element of type `Role[]`. When `AccessControlRequired` is applied to a JAX-RS resource method, the value of the element is the set of user roles who are allowed to invoke the REST endpoint. If the element is omitted, the REST endpoint only requires authentication, i.e. all user roles are allowed to invoke it. An example of how the annotation is used is given in listing 6.5. It should be clear from the example that the annotation makes it very simple to configure access rights whenever a new endpoint is introduced.

Listing 6.4: Annotation type used for specifying that a REST endpoint requires authentication and, optionally, authorization.

```
/**
 * When applied to a resource (or one of it's method), this annotation indicates that
 * execution of the resource (or the resource method) can only be performed by
 * authenticated users. Authorization is also supported by supplying a {@link Role} for
 * the value of this annotation. E.g., if {@code @AccessControlRequired({Role.PATIENT})}
 * is applied to a resource method, the method can only be executed by an authenticated
 * user whose role is {@code Role.PATIENT}. If no role is supplied, authenticated users in
 * all roles may execute the method. Method annotations override class annotations.
 */
@NameBinding
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE, ElementType.METHOD})
public @interface AccessControlRequired {
    Role[] value() default {};
}
```

Listing 6.5: Using the `AccessControlRequired` annotation to confine access to resource methods (REST endpoints).

```
@Stateless
public class PrescriptionResource {
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    // Both patients and medical staff may invoke this operation.
    @AccessControlRequired(value = {Role.PATIENT, Role.MEDICAL_STAFF})
    public List<Prescription> getPrescriptionsForPatient(/* omitted */) { /* omitted */ }

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    // Only medical staff may invoke this operation.
    @AccessControlRequired(value = {Role.MEDICAL_STAFF})
    public Response addPrescription(/* omitted */) { /* omitted */ }
}
```

The `AccessControlRequired` annotation is functionless on its own. It is only a marker that is used to inform the JAX-RS runtime that two custom `ContainerRequestFilters` [51] implementations should be invoked *before* a

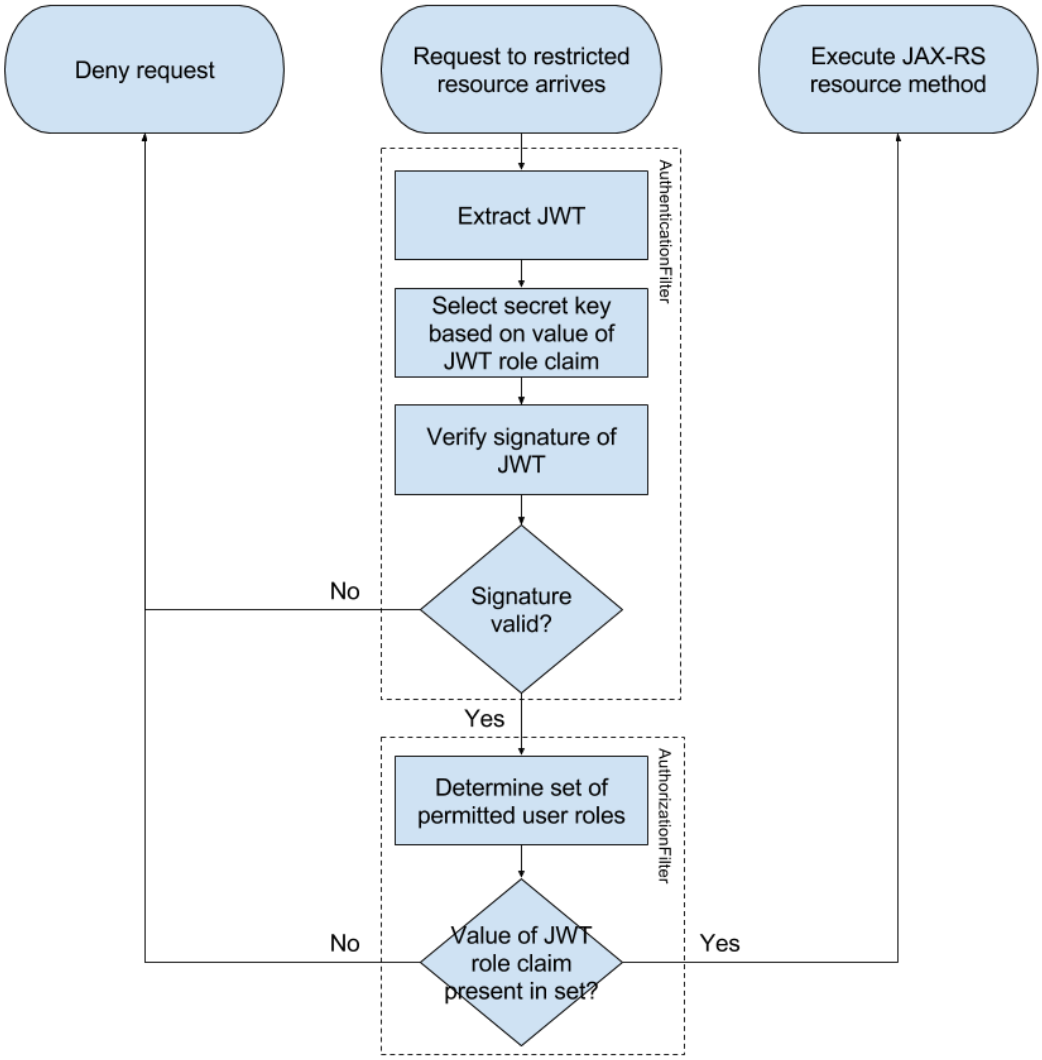
JAX-RS resource method is executed. The actual verification of the JWT happens in these two filters. The two filters are invoked sequentially. The first filter, called `AuthenticationFilter`, is responsible for extracting the JWT from the HTTP header and verifying its signature. As the secret key used for signature generation and verification is different depending on the JWT owner's role (see sect. 6.3.1.2), the filter must first select the proper secret key based on the value of the JWT's `role` claim before it proceeds to verify the signature. If signature verification fails, the request is immediately denied, and execution never reaches the JAX-RS resource method. If signature verification succeeds, the second filter, called `AuthorizationFilter`, is invoked. This filter uses Java reflection to obtain the set of roles provided to the `AccessControlRequired` annotation of the target JAX-RS resource method. It then compares the value of the `role` claim of the JWT to this set. If the value is present in the set, control proceeds to the JAX-RS resource method. If not, the request is immediately denied. The procedure is summarized by the flowchart in fig. 6.5.

6.3.1.4 Token Storage

The JWT is analogous to a physical key: anyone who obtains a copy can use it to gain access to the protected REST endpoints of Painkiller Server. Furthermore, the JWT uniquely identifies an individual and can hence serve as a tool for determining accountability for harmful behavior (e.g. misconfiguration of the PCOA device). It is therefore important that the JWT is inaccessible to third party code on the client device (the patient's iPhone and the medical staff member's iPad).

The JWT is stored in the iOS Keychain [22] which secures the JWT by encrypting it before storing it in the file system. This keeps the JWT safe even if the kernel is compromised as encryption and decryption happens in

Figure 6.5: Flowchart illustrating how Painkiller Server performs authentication and authorization checks.



the Secure Enclave [21], which is Apple's name for a coprocessor present in the Apple A7 or later A-series processors.

The Keychain API call that stores the JWT in the Keychain is invoked with the `kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly` and the `kSecAccessControlUserPresence` flags. These flags tell the Keychain API that access to the JWT requires fingerprint or, as a fallback, password authentication. As the JWT must be bundled in all requests to protected endpoints of Painkiller Server, the user will be prompted to authenticate using fingerprint whenever Painkiller Patient or Painkiller Staff prepares such a request since the application must first extract the JWT from the Keychain.

The previous paragraphs revealed an important point about how fingerprint authentication works in the Painkiller Software Platform, namely that it only serves as a form of *local* authentication that controls access to secret information (the JWT) that can in turn be used to perform authentication on the remote device (the PCOA device). In other words: *no* biometric data is transferred over the network. Biometric data is extremely sensitive information as it can be used for identity theft. It is hence favorable from a security and privacy point of view that such data remains on the local device as the attack surface grows when data is transferred over the network.

6.3.2 Pairing Procedure

When the PCOA device is assigned to a new patient, trust must be established between the patient's iPhone and the PCOA device. This will be referred to as the *pairing procedure*. The pairing procedure entails several challenges:

1. Only medical staff members should be allowed to initiate the pairing

procedure in order to prevent a third party from gaining control of the PCOA device by pairing his/her own device with the PCOA device.

2. Unlike the medical staff members' iPads, the patient's iPhone is a third party device and is hence not under the hospital's IT department's control. It therefore cannot be assumed that the iPhone knows the network address of the PCOA device in advance. The pairing procedure must therefore include a mechanism that lets the patient's iPhone locate the PCOA device on the network.

3. A valid JWT that uniquely identifies the patient must be generated and securely transferred to the patient's iPhone.

4. If the PCOA device was previously used by a former patient, the bond between that patient's iPhone and the PCOA device must be invalidated.

The pairing procedure is illustrated in fig. 6.6. The nurse initiates the pairing procedure from his/her iPad. This causes Painkiller Staff to send a request to a REST endpoint of Painkiller Server, telling Painkiller Server that a specific patient (identified by a user id embedded in the URI) is about to connect. In order to address challenge 1 from the list above, the REST endpoint is protected by the authentication and authorization mechanism described in sect. 6.3.1 – only users in the `MEDICAL_STAFF` role are allowed to invoke it. The request also causes Painkiller Server to overwrite the secret key used when generating and verifying signatures of JWTs issued to patients (sect. 6.3.1.2). This action solves challenge 4 from the above list as it invalidates the JWT that was issued to the previous patient since Painkiller Server will now attempt to verify its signature with a different key than the one that was used to construct the signature.

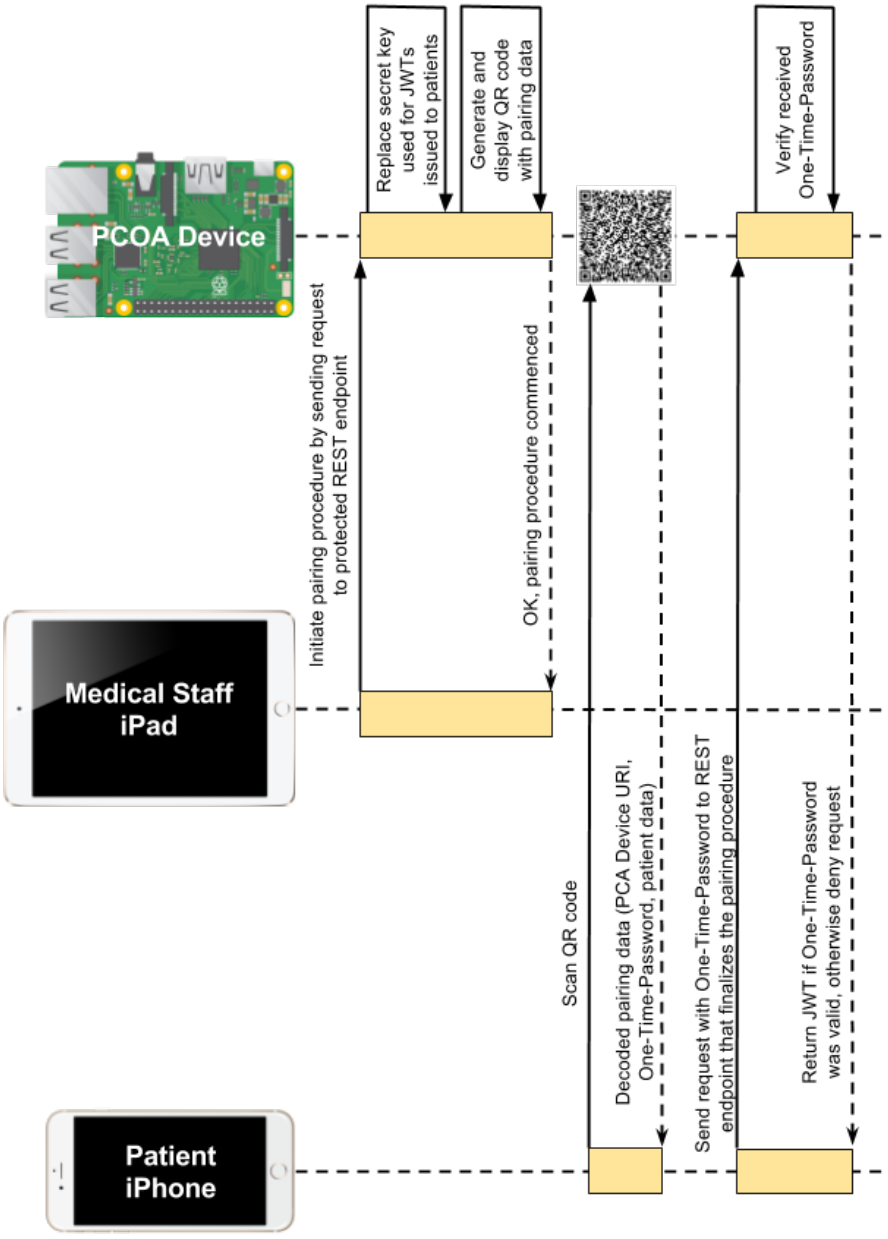


Figure 6.6: Interaction diagram illustrating the pairing procedure. Image attributions: `www.freeiconspng.com` (iPhone and iPad) and `www.raspberrypi.org` (Raspberry Pi).

Listing 6.6: Example of JSON embedded in the QR code that is displayed during the pairing procedure (formatting added for clarity).

```
{
  "dispenser": {
    "friendlyName": "Dispenser911",
    "url": "http://10.26.23.46:8080/pcoadeviceserver/webapi/"
  },
  "otp": "lrc2hpnpkill6mm670n2emsnl",
  "owner": {
    "cpr": "310182-1337",
    "firstName": "Boris",
    "id": 1,
    "lastName": "Grishenko",
    "role": "PATIENT"
  }
}
```

Next, Painkiller Server generates and displays a QR code that contains the information necessary for Painkiller Patient to be able to locate the PCOA device and finalize the pairing procedure. The information is structured as JSON, and an example is given in listing 6.6. The pairing procedure proceeds by the patient scanning the QR code and Painkiller Patient parsing its decoded content. The `dispenser` object in the JSON from listing 6.6 contains the base URI of the Painkiller Server instance running on the PCOA device, and challenge 2 from the list above has thereby been addressed as Painkiller Patient now knows where to direct its requests.

The `owner` object in the JSON from listing 6.6 is the patient that Painkiller Server expects to connect. Painkiller Patient uses it to ask the patient to confirm his/her identity before finalizing the pairing procedure. The point of this is to allow for the detection of human error early (see sect. 7.3). If the nurse mistakenly selects the wrong patient, and the pairing succeeds, the

patient will be dispensing medication prescribed to a different patient than herself.

The value pointed to by the `otp` key in the JSON from listing 6.6 is a one-time-password (OTP) generated using a cryptographically strong random number generator. Once the patient has confirmed his/her identity, Painkiller Patient sends the OTP back to a REST endpoint of Painkiller Server that finalizes the pairing procedure. Painkiller Server checks if the OTP matches the OTP it generated as part of the pairing data displayed in the QR code and if the OTP is still considered valid (in the prototype implementation, Painkiller Server considers an OTP to be expired five minutes after its issuance). If the OTP verification succeeds, Painkiller Server generates and sends back the patient's JWT, which concludes the pairing procedure.

7

Risk Analysis

The risk analysis presented here follows the recipe given by Basin et al. [5], but with a few deviations. First, present risk analysis does not include a section describing the system as this is the topic of chapter 6. Second, for the sake of brevity, asset state valuations are *not* defined on a per-stakeholder basis, but rather merged into a unified valuation in which the value of an asset state is based on how well the Painkiller Software Platform is able to achieve its overall goal for that asset state. The author argues that this simplification is sound as the stakeholders of the system share the same interests, namely to allow the patient to achieve analgesia without involving the nurse and to obtain insight into the patient's treatment.

Each asset state is assigned a value from the set **{High, Medium, Low, Undesired}**, where the values are arranged in descending order. States assigned the Undesired value represent a critical asset state where it is impossible, or, at the very least, extremely costly or cumbersome to regain the asset's value.

The straight forward way of arriving at an impact of a vulnerability, when dealing with asset state values that stem from an ordinal scale, is to define a matrix that specifies the impact for all possible combinations of source and destination asset values. Unfortunately, this strategy does not work here

Impact	**Description**
High	The event may: (1) lead to bodily harm; and/or (2) put the system out of use either permanently or for an extended period, in such a way that working order is only restored at high cost; and/or (3) lead to exposure of confidential information.
Medium	The event may put the system out of use: (1) for an extended period, in such a way that working order is restored only at low to moderate cost; or (2) for a short period, in such a way that working order is restored only at high cost.
Low	The event may put the system out of use for a short period in such a way that working order is restored at low to moderate cost.

Table 7.1: Definition of impact.

as the duration of the degradation of an asset's value and the cost involved in returning the asset to its source state are of importance. To provide an example of why this is so, consider a situation where the patient's iPhone shuts down because it runs out of power contrasted with a situation in which the patient renders their iPhone unusable for good by spilling a glass of water on it. In both cases, the patient will not be able to dispense medication and one could therefore argue that the impact is maximal as the system will not perform its single most important task. However, the first situation can be remedied in a matter of seconds or minutes whereas the second situation renders the system useless for the remainder of the patient's stay. It would therefore be very misleading to assign the same impact to both situations. As a consequence, present risk analysis adopts a heuristic approach to impact assessment. Once again an ordinal scale is used, this time consisting of the set **{High, Medium, Low}** where the values are arranged in descending order. Table 7.1 attempts to let the reader in on the reasoning used in the heuristic approach by providing a definition of each impact value.

The same set of values, **{High, Medium, Low}**, is used for specifying the likelihood of an event. The likelihood scale is presented in table 7.2 and is a combination of the likelihood scale given in [5] and a simplified version of the

Likelihood	Description
High	(1) The threat source is highly motivated and sufficiently capable of exploiting a given vulnerability, and there are no or only ineffective countermeasures in place; and/or (2) there is a strong possibility the event will occur as there is a history of frequent occurrences of similar events.
Medium	(1) The threat source is motivated and capable of exploiting a given vulnerability, but countermeasures that may impede a successful exploit of the vulnerability are in place; and/or (2) the event might occur at some time as there is a history of casual occurrences of similar events.
Low	(1) The threat source lacks motivation or capabilities to exploit a given vulnerability; and/or (2) countermeasures that prevent, or significantly impede, a successful exploit are in place; and/or (3) the event is not expected, but there's a slight possibility it may occur at some time.

Table 7.2: Definition of likelihood.

Likelihood	Impact		
	Low	Medium	High
High	Low	Medium	High
Medium	Low	Medium	Medium
Low	Low	Low	Low

Table 7.3: Risk-level matrix. Reproduced from [5].

likelihood scale given in [62]. The reasoning behind combining the two scales is that present risk analysis considers both motivated/intentional events and unmotivated/unintentional events (e.g. accidents). The scale presented in [5] does not take unmotivated events into account. In contrast, the scale presented in [62] does not consider the motivation of the threat source.

Finally, in order to be able to infer the *risk-level* for a given impact and likelihood, a risk-level matrix must be present. This risk analysis adopts the risk-level matrix presented by Basin et al. in [5] as it fits the impact and likelihood scales presented here. The matrix is reproduced in table 7.3.

In order to confine the scope of the risk analysis, it is assumed that the hospital's employees are *not* malevolent. An example of a malevolent employee could be an IT employee who abuses their administrative rights to gain access to personal information. It is also assumed that the implementation is correct. In other words, system malfunction that stem from programming mistakes will not be considered. Furthermore, malfunctioning of the mechanical dispensing mechanism (e.g. medication jam) will also not be considered as this thesis does not suggest a specific mechanical design.

7.1 Stakeholders

Patients: These are the patients who receive pain treatment. Patients are motivated by the ability to receive immediate pain relief as well as the ability to keep track of their drug usage.

Medical staff: These are the nurses and doctors responsible for treating the patients. Nurses are motivated by the ability to dedicate their time to other tasks than medication administration. Doctors are motivated by the information that the Painkiller Software Platform provides as it allows for a more informed evaluation of the pain treatment.

7.2 Assets

7.2.1 Physical Assets

PCOA device: This is the physical device that contains and dispenses the medication. In this prototype implementation, the medication storage and

State no.	State description	State value
(1)	Functional: the PCOA device dispenses tablets upon request.	High.
(2)	Nonfunctional: the PCOA device does not dispense tablets upon request.	Low
(3)	Missing: the location of the PCOA device is unknown.	Low

Table 7.4: The state space of the PCOA device.

dispensing mechanism remains imaginary as the focus of this thesis is on the software components. The PCOA device can be in one of the following states: (1) functional, (2) nonfunctional, or (3) missing. The states and their values are summarized in table 7.4.

Medication: This asset is the medication tablets that are stored in, and dispensed from, the PCOA device. The state space of each individual tablet is shown in table 7.5. The states (1), (3), (5), and (8) are all described as *intentional* as they are the possible states a medication tablet can be in when the system is working as intended. It follows that these states are of the highest possible value. The remaining states are all undesirable states. State (2) is assigned a higher value than the other undesirable states as the medication is still locked away which prevents undesired events such as overdose and theft. State (8) is assigned the lowest value as this state is dangerous to the patient's health (the patient has overdosed).

Medical Staff iPads: The iPads are used by the medical staff to configure the PCOA device and access its patient data (e.g. drug usage and pain information). The state space of an iPad is shown in table 7.6.

Patient iPhone: The patient's iPhone is the patient's tool for operating the PCOA device. Its state space is shown in table 7.6.

Hospital Network: The hospital network infrastructure facilitates com-

State no.	**State description**	**State value**
(1)	Intentionally locked away in the PCOA device.	High.
(2)	Unintentionally locked away in the PCOA device.	Medium.
(3)	Intentionally dispensed to the tablet tray of the PCOA device.	High.
(4)	Unintentionally dispensed to the tablet tray of the PCOA device.	Low.
(5)	Intentionally in the patient's possession.	High.
(6)	Unintentionally in the patient's possession.	Low.
(7)	Intentionally consumed by the patient.	High.
(8)	Unintentionally consumed by the patient.	Undesired.
(9)	In a third party's possession.	Low.

Table 7.5: The state space of the medication.

State no.	**State description**	**State value**
(1)	Functional: the iPad/iPhone is fully functional and the user is aware of its location.	High.
(2)	Nonfunctional: the iPad/iPhone is not functional (e.g. it cannot be powered on or is unresponsive), but the user is aware of its location.	Low
(3)	Missing: the user is unaware of the location of the iPad/iPhone.	Low

Table 7.6: The state space of the iPads and the iPhone.

State no.	State description	State value
(1)	Operational: the patient's iPhone and the medical staff's iPads can communicate with the PCOA device.	High
(2)	Down: the patient's iPhone and the medical staff's iPads can *not* communicate with the PCOA device.	Low

Table 7.7: The state space of the hospital network.

State no.	State description	State value
(1)	Confidential: the information is only accessible to the patient and hospital personnel bound by professional secrecy.	High
(2)	Compromised: the information is accessible to one or more people who are not part of the group of people specified in state (1).	Undesired.

Table 7.8: The state space of the patient data.

munication between the PCOA device and the patient's iPhone and between the PCOA device and the medical staff members' iPads. Its state space and associated valuation is shown in table 7.7.

7.2.2 Logical Assets

Patient Data: This includes all patient related information such as name, CPR-number, drug usage, and pain scores. All information is confidential as it can have a negative impact on the patient if obtained by a third party (e.g. identity theft). The state space of the information is defined in terms of who has access to the information. The asset decreases in value if anyone else but the patient and the group of staff members (doctors, nurses, and IT-employees), who are bound by professional secrecy, gains access to the information. The state space and associated valuation is shown in table 7.8.

State no.	State description	State value
(1)	Confidential: the login token owner is the only person who has access to the login token.	High
(2)	Compromised: the login token can be accessed by someone else but its owner.	Undesired.

Table 7.9: The state space of a login token.

Login Token: This is the information that is required for operating the PCOA device. It is used for authentication and authorization when a user attempts to perform an operation. A token uniquely identifies an individual and can hence be used as a tool for determining accountability, for example in the event of misconfiguration of the PCOA device. This in turn means that a token loses its value as soon as it is exposed to anyone but the target individual. The state space and valuation of a login token is summarized in table 7.9.

Cryptographic Keys. These are the secret keys used for signing and encrypting communication between the PCOA device and the client devices, i.e. the staff members' iPads and the patient's iPhone. They are stored in a Java Keystore on the PCOA device. Access to the keystore is protected by password (i.e. its content is encrypted using the password). The asset decreases in value if anyone else but the hospital's IT employees, who are responsible for PCOA device maintenance, gains access to any of the cryptographic keys. The state space and valuation is summarized in table 7.10.

7.3 Vulnerabilities

This section describes the vulnerabilities that have been identified and which countermeasures are in place or should be considered. As this thesis focuses on the software components, countermeasures relating to the physical design

State no.	State description	State value
(1)	Confidential: the cryptographic keys are only accessible to the IT professionals in charge of maintaining the PCOA device.	High
(2)	Compromised: the cryptographic keys are accessible to one or more people outside the group of people described in state (1).	Undesired.

Table 7.10: The state space of the cryptographic keys used for signing and encrypting communication between the PCOA device and the client devices.

of the PCOA device can only be presented as proposals and not final plans.

Electricity. The PCOA device, the staff members' iPads, and the patient's iPhone all depend on the availability of electricity. A countermeasure is already in place as hospitals have backup generators that will supply power in case of power outage. However, the user study in sect. 5.1 revealed that it is important that the PCOA device is portable. This in turn means that it cannot be connected to power continuously. The PCOA device should therefore be equipped with a battery. Wireless charging should be considered such that battery recharge will be ensured by the patient simply placing the PCOA device on its designated spot when returning to bed (the hypothesis being that it is less likely that the patient will always recharge the PCOA device if they have to to fiddle with a power cable in order to do so).

The possible asset state transitions resulting from the manifestation of the electricity vulnerability are shown in table 7.11. The impact of the vulnerability is low across the board as its manifestation will only temporarily change the state of the affected asset: when power is restored, the asset will return to its source state. Moreover, the manifestation of the vulnerability does not endanger the patient's health – it only temporarily prevents the patient from achieving analgesia and accessing their drug use data.

Water. Electronic devices are vulnerable to water damage. Of all the

Asset	Source state	Destination state	Impact
PCOA device	(1)	(2)	Low
Medical Staff iPad	(1)	(2)	Low
Patient iPhone	(1)	(2)	Low

Table 7.11: Asset state transitions resulting from the manifestation of the electricity vulnerability.

Asset	Source state	Destination state	Impact
PCOA device	(1)	(2)	High
Medical Staff iPad	(1)	(2)	High
Patient iPhone	(1)	(2)	High

Table 7.12: Asset state transitions resulting from the manifestation of the water vulnerability.

electronic devices, the PCOA device is most likely to get wet as some patients like to bring it with them when they shower (sect. 5.1). A water resistant design, for example inspired by the ones seen in recent smartphones, should hence be considered. The possible asset state transitions resulting from the manifestation of the water vulnerability are shown in table 7.12. The impact of the vulnerability is high across the board as it renders the asset unusable for good.

Network issues. The network may become unavailable due to events such as maintenance, access point failure, or accidental damage. As a countermeasure, the hospital may consider to install extra access points such that any location is covered by at least two access points, but this may obviously be too expensive for some hospitals. The possible asset state transitions resulting from the manifestation of the network issues vulnerability are shown in table 7.13. Notice that the PCOA device is indirectly affected as its operation relies on network connectivity. The impact is low as the state change is not permanent: when network connectivity is restored, the PCOA device will return to its source state.

Asset	Source state	Destination state	Impact
Hospital Network	(1)	(2)	Medium
PCOA device	(1)	(2)	Low

Table 7.13: Asset state transitions resulting from the manifestation of the network issues vulnerability.

Human Error. When preparing the Painkiller Software Platform for use, the nurse must enter information about the prescription into the system. This includes specifying the type of drug, the number of tablets to dispense for each dose, how long time must elapse between two successive doses, and how many doses the patient is allowed to consume every 24 hours. Needless to say, this is subject to human error: if the nurse makes a mistake and inputs an incorrect value during setup, the system will assume that the value is correct and operate accordingly. For example, the nurse might mistakenly input a tablet count that is greater than the intended value, which may in turn make the patient overdose as the system will dispense more medication than what was prescribed by the doctor. Additionally, the PCOA device must be (re-)filled with tablets manually. This procedure also leaves room for human error as the nurse may mistakenly load the PCOA device with the wrong medication, causing a mismatch between what the PCOA device *thinks* it dispenses, and what it *actually* dispenses.

The asset state transitions that may occur as a result of the manifestation of the human error vulnerability are listed in table 7.14. The transitions with source state (1) originate from human mistakes during system configuration, while the transitions with source state (2) originate from human mistakes during the tablet (re-)filling task.

Eavesdropping. Patient data and login tokens are transmitted between the PCOA device, the staff members' iPads, and the patient's iPhone and are therefore vulnerable to eavesdropping while in transit. However, a counter-

Asset	Source state	Destination state	Impact
Medication	(1)	(4)	Medium
Medication	(1)	(6)	Medium
Medication	(1)	(8)	High
Medication	(2)	(4)	Medium
Medication	(2)	(6)	Medium
Medication	(2)	(8)	High

Table 7.14: Asset state transitions resulting from the manifestation of the human error vulnerability.

Asset	Source state	Destination state	Impact
Patient Data	(1)	(2)	High
Login Token	(1)	(2)	High

Table 7.15: Asset state transitions resulting from the manifestation of the eavesdropping vulnerability.

measure is in place: the data is transferred locally over the hospital's wireless network instead of over the Internet. An adversary in proximity of the hospital may use radio equipment to eavesdrop on the communication without being connected to the network. This is mitigated by the hospital network's use of WPA 2 Enterprise security. Given the countermeasures in place, an adversary, who wishes to exploit the eavesdropping vulnerability, must either be part of the local network, or skilled enough to break the firewall guarding the border between the hospital's local network and the Internet. The possible asset state transitions resulting from the manifestation of the eavesdropping vulnerability are shown in table 7.15.

Physical access. Physical access to the medication and/or the devices that participate in the Painkiller Software Platform can be harmful to both physical and logical assets. Physical assets, e.g. the PCOA device or the patient's iPhone, may get damaged or go missing from physical access. This can occur from intentional as well as unintentional actions of the individual who has physical access to the system. Theft is an example of a an intentional

threat action that exploits the physical access vulnerability.

Fortunately, there are some obvious – and one readily available – countermeasures which can help prevent, or recover from, damage to physical assets. As the electronic devices are connected to the hospital's Wi-Fi, a device that has been lost, but is still powered on and present at the hospital site, may be relocated by a network administrator by determining what access point the device is connected to. Additionally, Apple's *Find My iPhone, iPad, and Mac* [24] may also be used to relocate a missing iPhone or iPad. In order to better secure the medication, the PCOA device should be designed so that the medication is locked away and will require substantial force to access. Moreover, the PCOA device should only contain a limited amount of tablets such that the profit in breaking in is small.

When it comes to logical assets, physical access to IT systems can allow for administrative operations, access to the file system, and/or the insertion and execution of malicious code. For the Painkiller Software Platform, this may lead to the exposure of patient data, login tokens, and cryptographic keys. If a login token is stolen, the thief can impersonate the individual to whom the token was originally issued. The exposure of a cryptographic key is even worse as it may be used to produce valid tokens, which the adversary can then use to invoke any of the operations of the PCOA device.

As a countermeasure, the hospital's IT policy should state that staff members' iPads must have fingerprint/password authentication enabled in order to prevent an adversary from unlocking the device. Additionally, the exterior of the PCOA device should shield any I/O ports, and it should not be equipped with any input devices. The keystore that contains the cryptographic keys uses a password to encrypt its contents so that the keys remain safe even if the keystore file is stolen. The login tokens are protected in a similar fashion as they are stored in the iOS Keychain. Patient login tokens

Asset	Source state	Destination state	Impact
PCOA device	(1)	(2)	High
PCOA device	(1)	(3)	High
Patient's iPhone	(1)	(2)	High
Patient's iPhone	(1)	(3)	High
Medical staff iPad	(1)	(2)	High
Medical staff iPad	(1)	(3)	High
Medication tablet	(1), (3), or (5)	(9)	High
Patient Data	(1)	(2)	High
Login Token	(1)	(2)	High
Cryptographic Keys	(1)	(2)	High

Table 7.16: Asset state transitions resulting from the manifestation of the physical access vulnerability.

are invalidated when the PCOA device is assigned to the next patient which makes the timeframe, in which a compromised login token can be used, small.

The countermeasures mentioned up until now do not offer any protection against the event in which an adversary gains physical to access to an unlocked device. This would let the adversary act as if he were the owner of said device. In the case of the patient's iPhone, this means that the adversary will be able to dispense medication and view patient data. In the case of a medical staff member's iPad, the adversary may reconfigure the PCOA device and access patient data. This attack is mitigated by requiring the user to authenticate using fingerprint/password whenever a mutating operation is performed. The possible asset state transitions resulting from the manifestation of the physical access vulnerability are shown in table 7.16.

Software Vulnerabilities. Any of the three software applications that make up the Painkiller Software Platform may contain security related bugs that allow for hostile takeover of the system. Additionally, the Painkiller Software Platform is built on top of a set of external software components and is therefore subject to published and unpublished vulnerabilities present

in these underlying components. These external software components include the operating systems of all the devices participating in the Painkiller Software Platform, external software libraries used in the implementation of the Painkiller Software Platform, and the container (i.e. the application server) for the Painkiller Server application (the software controlling the PCOA device). For example, Painkiller Server is a REST service built using the Jersey JAX-RS implementation. As such, any security vulnerability present in the Jersey implementation propagates to Painkiller Server. Exploitation of software vulnerabilities may lead to the exposure of confidential information (patient data, login tokens, cryptographic keys). In more extreme cases, the adversary may gain full control of a device and render it nonfunctional for its user. The asset state transitions are summarized in table 7.17.

Some countermeasures are in place. First, the part of the code for Painkiller Server that uses Java EE APIs (e.g. JAX-RS and JPA) is written using only classes and interfaces available in the respective specifications. In other words: it does *not* rely on classes specific to a given vendor's implementation of said APIs. As a result, if the implementation in use proves vulnerable, it can be replaced with a different vendor's implementation without any modification to the source code. Second, the hospital's IT policy should require that the software on the PCOA device and the medical staff members' iPads is updated regularly in order to prevent against exploitation of recent published vulnerabilities. In contrast, the patient's iPhone is a private device, and hence there is no policy for how often its software is updated. It should therefore be considered the weakest point for this particular vulnerability. Even worse, the set of software applications running on the patient's iPhone is arbitrary and may include spyware which can potentially gain access to some of the confidential information exchanged between the PCOA device and the patient's iPhone.

Asset	Source state	Destination state	Impact
Patient Data	(1)	(2)	High
Login Token	(1)	(2)	High
Cryptographic Keys	(1)	(2)	High
PCOA device	(1)	(2)	High
Medical Staff iPad	(1)	(2)	High
Patient's iPhone	(1)	(2)	High

Table 7.17: Asset state transitions resulting from the exploitation of a vulnerability in an underlying software component.

7.4 Threat Sources

Staff. As stated in the beginning of this chapter, it is assumed that no staff member is interested in causing the patient intentional harm. However, the Painkiller Software Platform relies on staff members to configure and maintain the system. Staff members may make mistakes when performing these tasks and may hence end up causing unintentional harm.

Patient. The PCOA device and the patient's iPhone are in the hands of the patient. While the patient may not have a motive to damage said devices, there is still potential for damage occurring from careless behavior.

Thieves. The PCOA device, the staff members' iPads, and the patient's iPhone are all easy targets for thieves as they are portable devices. In addition, the iPads and the iPhone are expensive consumer electronics which can easily be converted to cash. Motivation is primarily monetary. However, a thief, who is also a drug addict, may also be motivated by his addiction: the PCOA device may contain morphine which can serve as a replacement drug for a desperate drug addict in need of a fix.

Terrorists. When IT systems control physical behavior, manifestation of system vulnerabilities can cause physical havoc and human casualties. Such

systems are obvious targets for terrorists as these people are driven by a desire to cause destruction and chaos. Terrorists may work independently, but they may also employ skilled hackers to help them achieve their goals. The Painkiller Software Platform falls into this category of systems as it is responsible for ensuring that the right amount of medication is dispensed. If a system vulnerability allows for an increase in the amount of medication being dispensed, the patient may overdose, which can cause severe bodily harm or even death.

Skilled hackers. Skilled hackers may employ both published and unpublished exploits to penetrate the software of the PCOA device and/or eavesdrop on communication between the PCOA device and the iPads and the iPhone. Motivation is mostly monetary. For example, hackers can sell stolen personal information on a black market. Secondary motivation factors are challenge and glory.

Script kiddies. These are similar to the skilled hackers, but they have less capabilities. Script kiddies mostly rely on published vulnerabilities and tools available online, but they may also write simple code to automate tasks (e.g. code that makes repeated requests to the system with the purpose of making the system unresponsive to its users (denial of service)). They are primarily motivated by challenge, glory, and destruction.

7.5 Risk

To arrive at a concluding measure for risk, one now considers what vulnerabilities each individual threat source might exercise. This is done on a per-asset basis. The threats are presented in table form and are accompanied by text that explains the reasoning behind the assigned likelihood and impact values. The resulting risk-level comes from plugging the likelihood

and impact values into the risk-level matrix presented in the beginning of this chapter (see table 7.3). This section does not discuss how to respond to the identified risks. That part is deferred to sect. 7.6.

7.5.1 PCOA Device

The threats targeting the PCOA device are presented in table 7.18. As for any system with a physical component, accidental damage may occur, but should be rare, assuming a water resistant design. The impact, however, is high as it renders the system useless for the remainder of the patient's stay. One could argue that the impact could be lowered as it would be reasonable to assume that a backup PCOA device is available. Unfortunately, this is not the case as drug use data is stored locally on the PCOA device, and migrating the patient to a new PCOA device would therefore mean that any ongoing lockout timers would be lost, which in turn means that the patient could overdose.

In order to be portable, the envisioned PCOA device must rely on battery power, and hence there is high likelihood that it will run out of power at some point. The impact is low, even though the system is put out of use, as the situation can easily be remedied by recharging the device.

The likelihood of theft is obviously a local matter, but in general it should be safe to say that the number of incidents compared to the amount of people visiting a hospital on a daily basis is relatively low. For example, the Copenhagen Police Department reports that there were 151 incidents of theft from hospitals and nursing homes in the Copenhagen area in 2012 [50]. The impact is high since the event puts the system out of use and, like stated previously, the PCOA device cannot be replaced by a backup device.

Digital attacks can also move the PCOA device from its functional state

to its nonfunctional state. A script kiddie may attempt a (D)DoS attack. Currently, the Painkiller Software Platform does not offer any server-side protection to detect and mitigate such an attack. However, the likelihood of this event occurring is low as the adversary must be connected to the hospital network since he is not capable of breaking the firewall between the hospital's local network and the internet. The effort involved in bringing enough computing power for a (D)DoS attack to the hospital site is assumed to be too big for the adversary to attempt such an attack. The impact is medium as the network administrator should be capable of detecting and shutting down the excessive traffic.

Finally, a skilled hacker may exploit a software vulnerability in the PCOA device to gain control of it, for example with the purpose of making it part of his botnet. The likelihood of this event is medium as countermeasures are in place (firewall, policy for updating software regularly), but a truly skilled hacker will be capable of circumventing them. In fact, the countermeasures may even be a motivating factor as the adversary is, among other things, motivated by challenge. It is assumed that the hacker will try to remain undercover in order to keep the PCOA device part of the botnet for as long as possible. There is hence no reason for the hacker to shut down the Painkiller Server application, and the users of the Painkiller Software Platform will therefore only experience disruptions when the hacker puts the PCOA device under heavy load. Therefore, the impact is low.

7.5.2 Medication

The threats targeting the medication are summarized in table 7.19. First on the list are staff members who make mistakes as part of configuring or refilling the PCOA device. As mentioned in sect. 7.3, this may lead to too much or the wrong medication being dispensed which is ultimately a

#	**Threat Source(s)**	**Threat Action**	**L**	**I**	**Risk**
1	Patient, Staff	Physical access, water: these individuals may cause accidental damage to, or misplace, the PCOA device.	Low	High	Low
2	Patient, Staff	Electricity: these individuals may forget to recharge the PCOA device.	High	Low	Low
3	Thieves	Physical access: a thief may steal the PCOA device, for example to get his hands on the medication it contains.	Low	High	Low
4	Script Kiddies	Software Vulnerabilities: render the PCOA device unresponsive by flooding it with requests.	Low	Medium	Low
5	Skilled Hackers	Software Vulnerabilities: take over device and make it part of botnet.	Medium	Low	Low

Table 7.18: Threats targeting the PCOA device.

potential health hazard, hence the impact is high. According to White [67], operator error is one of the most common problems for the safety of PCA pumps. Additionally, the work of Vicente et al. [65] suggests that operator errors for a (specific) PCA pump result in a mortality rate which is – at the very least – comparable to the mortality rate for conventional analgesia. While it is acknowledged that these findings are specific to intravenous PCA, they still indicate that human error is a factor that should not be overlooked, and the likelihood is therefore deemed to be medium.

The second threat is terrorists who, by exploiting vulnerabilities in the software stack, increase the amount of medication being dispensed with the purpose of making the patient overdose. The impact of the event is certainly high, as it may entail bodily harm. However, the likelihood is low since the adversary lacks motivation, namely because a successful attack does not guarantee overdose when dealing with oral as opposed to intravenous medication. For oral medication, the patient must actively consume the medication. For intravenous medication, the patient remains passive while the medication is injected. In the former case, an alert patient will realize that too much medication was dispensed and refrain from consuming it.

Theft is the final threat on the list, but it has already been discussed in sect. 7.5.1, and there is nothing new to add for this asset.

7.5.3 Medical Staff iPads and Patient iPhone

The threats targeting the medical staff members' iPads and the patient's iPhone are shown in table 7.20 and 7.21, respectively. All threats on both lists have already been discussed in sect. 7.5.1, and there is nothing new to add for these two assets.

#	**Threat Source(s)**	**Threat Action**	**L**	**I**	**Risk**
6	Staff	Human Error: misconfigure PCOA device, load PCOA device with wrong medication.	Medium	High	Medium
7	Terrorists	Software Vulnerabilities: (employ skilled hackers to) increase the amount of medication being dispensed.	Low	High	Low
8	Thieves	Physical access: steal the medication, e.g. by breaking into the PCOA device.	Low	High	Low

Table 7.19: Threats targeting the medication.

#	**Threat Source(s)**	**Threat Action**	**L**	**I**	**Risk**
9	Staff	Physical access, water: accidental damage to, or misplacement of, the iPad.	Low	High	Low
10	Staff	Electricity: forget to recharge iPad.	High	Low	Low
11	Thieves	Physical access: steal and resell the iPad.	Low	High	Low
12	Skilled Hackers	Software Vulnerabilities: take over device and make it part of botnet.	Medium	Low	Low

Table 7.20: Threats targeting the medical staff members' iPads.

#	Threat Source(s)	Threat Action	L	I	Risk
13	Patient	Physical access, water: accidental damage to, or misplacement of, the iPhone.	Low	High	Low
14	Patient	Electricity: forget to recharge iPhone.	High	Low	Low
15	Thieves	Physical access: steal and resell the iPhone.	Low	High	Low
16	Skilled Hackers	Software Vulnerabilities: take over device and make it part of botnet.	Medium	Low	Low

Table 7.21: Threats targeting the patient's iPhone.

7.5.4 Hospital Network

The threats targeting the hospital network are summarized in table 7.22. First on the list is accidental damage to network equipment. It is assumed that an on-call IT staff member and/or a network administrator will be able to repair or replace the damaged equipment within reasonable time, so the impact is only medium. Finally, a vandal may walk into the hospital and start a network jammer. Software tools for jamming Wi-Fi signals are available online (e.g. [44]), so it requires little effort for the adversary to exploit this vulnerability, and hence the likelihood is deemed to be medium. The impact, on the other hand, is low as everything returns to normal at the time the attack ends.

#	Threat Source(s)	Threat Action	L	I	Risk
17	Staff	Network Issues: accidental damage to network equipment.	Low	Medium	Low
18	Script Kiddie (or other vandal)	Network Issues: Make use of a network jammer to disrupt network connectivity.	Medium	Low	Low

Table 7.22: Threats targeting the hospital network.

7.5.5 Patient Data, Login Tokens, and Cryptographic Keys

Threats targeting the patient data, login tokens, and cryptographic keys are summarized in tables 7.23, 7.24, and 7.25, respectively. They are treated as one as they are mostly subject to the same threats. All impact values are high as damage is permanent: the confidentiality of information cannot be restored.

Common to all three assets is that they are targets of skilled hackers and script kiddies. These two adversaries exploit the same vulnerabilities, but as the skilled hacker is the more capable of the two, the likelihood will generally be greater for this type of adversary. In fact, countermeasures (firewall, software update policy) are in place that greatly reduce the likelihood that a script kiddie will be able to carry out a successful attack. The exception to the rule is threats where the adversary exploits the eavesdropping vulnerability. As there are no countermeasures in place that prevent the use of tools such as Wireshark [15] for inspecting network traffic, both the script kiddie and the skilled hacker are highly capable of carrying out this type of attack. The cryptographic keys are not subject to this type of attack as they are not transferred over the network.

#	**Threat Source(s)**	**Threat Action**	**L**	**I**	**Risk**
19	Skilled Hackers	Software vulnerabilities, physical access: access patient data by (1) exploiting (un)published vulnerabilities in the software stack, or (2) abusing physical access to execute administrative operations or inject malicious code.	Medium	High	Medium
20	Script Kiddies	Software Vulnerabilities, physical access: access patient data by (1) exploiting published vulnerabilities in the software stack, or (2) abusing physical access to execute administrative operations or inject malicious code.	Low	High	Low
21	Skilled Hackers	Eavesdropping: snoop on patient data transferred over the network by employing existing, or developing new, eavesdropping tools.	High	High	High
22	Script Kiddies	Eavesdropping: snoop on patient data transferred over the network by employing existing eavesdropping tools.	High	High	High
23	Staff	Human Error leading to Physical Access: expose patient data by leaving iPad unlocked in a public area.	Low	High	Low

Table 7.23: Threats targeting patient data.

#	**Threat Source(s)**	**Threat Action**	**L**	**I**	**Risk**
24	Skilled Hackers	Software vulnerabilities, physical access: obtain login tokens by (1) exploiting (un)published vulnerabilities in the software stack, or (2) abusing physical access to execute administrative operations or inject malicious code.	Medium	High	Medium
25	Script Kiddies	Software vulnerabilities, physical access: obtain login tokens by (1) exploiting published vulnerabilities in the software stack, or (2) abusing physical access to execute administrative operations or inject malicious code.	Low	High	Low
26	Skilled Hackers	Eavesdropping: snoop on login tokens transferred over the network by employing existing, or developing new, eavesdropping tools.	High	High	High
27	Script Kiddies	Eavesdropping: snoop on login tokens transferred over the network by employing existing eavesdropping tools.	High	High	High

Table 7.24: Threats targeting the login tokens.

#	**Threat Source(s)**	**Threat Action**	**L**	**I**	**Risk**
28	Skilled hackers	Software vulnerabilities, physical access: obtain copies of the cryptographic keys by (1) exploiting (un)published vulnerabilities in the software stack, or (2) abusing physical access to execute administrative operations or inject malicious code.	Medium	High	Medium
29	Skilled hackers	Software vulnerabilities, physical access: obtain copies of the cryptographic keys by (1) exploiting published vulnerabilities in the software stack, or (2) abusing physical access to execute administrative operations or inject malicious code.	Low	High	Low

Table 7.25: Threats targeting the cryptographic keys.

Finally, if a nurse leaves his/her iPad unlocked and unattended, patient data may be viewed by anyone who passes by. The likelihood of this event is deemed to be low as it is already part of the IT policy of Danish hospitals that staff members cannot leave their computers unlocked.

7.6 Conclusion

The risk analysis has found that all threats with a high risk-level involve the exposure of confidential information. This may seem surprising at first as the system under analysis regulates medication intake. Indeed, when dealing with intravenous medication, overdispensation due to system failure causes inevitable overdose and the resulting risk-level would probably be high. However, the system under analysis targets orally administered medication, and here the alert patient is given an opportunity to detect system failure as she must actively pick up and consume the drugs.

All threats with high risk-levels exercise the eavesdropping vulnerability. In a production environment, these threats should be mitigated by applying state of the art encryption.[1] The proposed countermeasure would greatly reduce the likelihood of these threats. In particular, script kiddies would no longer be capable of exercising the vulnerability, and the group of capable skilled hackers would be narrowed down to the absolute elite, namely those who have knowledge of zero-day vulnerabilities in the encryption algorithm.

It has also been shown that skilled hackers may gain access to confidential information by exploiting software vulnerabilities. For example, the attacker could gain access to the database file through a security flaw in the operating system of the PCOA device. Some countermeasures (firewall, software

[1] Encryption has been left out in the prototype implementation as it is mostly a server configuration task and not a programming task.

update policy), which reduce the likelihood of these threats, are already in place, but little has been done in terms of reducing the impact of a successful attack. To this end, the obvious action is to encrypt the database contents.

Last, but certainly not least, it was discovered that human error during system configuration should not be overlooked as it is the most common cause of overdose in similar systems. To help prevent the set of errors that result from mistyping, a screen that summarizes the created configuration and prompts the nurse to confirm it could be presented just before the configuration is put to effect. Another obvious countermeasure is to require an independent "double-check" by a second caregiver as suggested by Vicente et al. [65]. To help the nurses save time, this could be kept digital: when nurse Alice completes the configuration from her iPad, a notification prompting for confirmation could appear on nurse Betty's iPad. The downside to the digital solution is that it will not bring any extra reassurance that the PCOA device is loaded with the correct tablets.

8

Discussion

This chapter reflects on the results produced by the project activities and examines the consequences of some aspects of the design of the Painkiller Software Platform. Section 8.1 discusses how the findings of the risk analysis (see ch. 7) affects the target audience of the Painkiller Software Platform. The utility of fingerprint authentication is discussed in sect. 8.2 as it is a feature that sets the Painkiller Software Platform apart from other PCA devices. Section 8.3 discusses what ramifications some aspects of the user study (see ch. 5) has on its findings. Section 8.4 considers how the automation of pain assessments could be further improved to produce even better treatment data. Finally, ideas for future work are presented in sect. 8.5.

8.1 Patient Selection

The risk analysis presented in ch. 7 arrived at a conclusion that was perhaps a little surprising. A system that helps a patient manage his/her medication has a direct impact on the patient's health. It would therefore be sensible to anticipate that the greatest risks entailed by such a system stem from events and behavior that result in incorrect amounts of medication being dispensed. However, it was found that the greatest risks actually involve the exposure of

confidential information, and not overdose due to a malfunctioning system.

The conclusion was based on the fact that oral medication, as opposed to intravenous medication, requires patient action which leaves room for the patient to detect system failure and thereby avoid overdose. This constrains the target group as it relies on the assumption that the patient is alert and does not trust the system blindly. One could therefore argue that the assumption defeats the very purpose of the system as such patients would also be capable of managing the medication manually. However, the purpose of the Painkiller Software Platform is not *just* to ensure safe self-administration of medication, but also to provide treatment transparency and automation of tasks such as (follow-up) pain assessments. Furthermore, a similar assumption is already in place for intravenous PCA as indicated by a safety alert sent out by the Institute for Safe Medication Practices (ISMP) which clearly stresses the importance of proper patient selection [25]: "[...] candidates for PCA should have the mental alertness and cognitive, physical, and psychological ability to manage their own pain."

8.2 Use of Fingerprint Authentication

Section 5.1.4 found that patients were not concerned about the possibility that anyone could pick up the remote control of the PCA pump and force a dose upon them against their will. The nurse also considered such a scenario to be unlikely as she had never experienced it [Nurse01 00:42:57-00:45:10]. Although malevolent activation of the PCA pump was considered unlikely by users in the user study, the ISMP warns against PCA by proxy[1] as it can have significant deleterious effects and has even led to fatal incidents [8].

[1]The term PCA by proxy is used when someone other than the patient activates the PCA device on the patient's behalf (usually done in good faith).

PCA is intended to be *patient*-controlled, and an inherent safety feature is that the sedated or sleeping patient usually refrains from activating the PCA pump [68]. This safety feature is overridden when someone else, for example a loved one who has misunderstood the concept of PCA, activates the PCA pump on the patient's behalf.

As a solution, ISMP suggests that PCA equipment should be labeled with warnings about the dangers of PCA by proxy [8]. The Painkiller Software Platform addresses the problem in a more pragmatic way as it *prevents* anyone but the patient from activating the PCOA device by requiring the patient to authenticate themselves using their fingerprint. Admittedly, the solution is not bulletproof as the proxy could simply position the patient's finger on the iPhone's fingerprint scanner. However, such an act has identity theft written all over it which may make the proxy reconsider if it is something that she should be doing or not. Another issue is the fact that iOS allows for the registration of multiple fingerprints (up to 5) and that fingerprint verification can be bypassed by providing the password. If a couple shares an iPhone (both parties have their fingerprints registered and/or know the password), the spouse may activate the PCOA device on the patient's behalf.

Despite these flaws, the Painkiller Software Platform does add an extra layer of security that may help reduce the number of incidents of unauthorized PCA by proxy. However, PCA by proxy is presumably a (very) small problem in PCOA as the patient must be awake to swallow the medication. It would therefore be interesting to investigate if the idea of using fingerprint authentication could be transferred to PCA devices for intravenous medication. As discussed in sect. 8.1, the risk analysis carried out as part of this work assumes that the patient is (to some extent) capable of detecting if the system dispenses excessive amounts of medication. This assumption does not hold for intravenous medication, and it would therefore be necessary to conduct a new risk analysis if one would want to pursue this.

8.3 Limitations of User Study

This section discusses some aspects of the user study that may have had an impact on its results. Section 8.3.1 discusses how the user inclusion criteria may have produced an overly positive view on technology-assisted PCA. Section 8.3.2 discusses how the size of the user study may affect the generality of its results.

8.3.1 Patients' Views on Technology-Assisted PCA

As described in ch. 5, interview subjects were selected based on the criteria that they had first-hand experience with technology-assisted PCA. This requirement was necessary in order to allow for questions regarding the interviewees' experiences with such technology. However, the inclusion parameter is also slightly problematic as there is a chance that it may have produced an overly positive view on technology-assisted PCA since it may have excluded users who are skeptical of such technology. The PCA pumps were used as part of a clinical trial, and patients were free to choose if they wanted to participate or not. It is more or less guaranteed that patients, who choose to participate in such a study, are *not* die-hard opponents of technology-assisted PCA. Add to this that both patients seemed to be technology accustomed: both patients were iPhone users, and one was a marine engineer by profession. In fact, one patient even declared that he was enthusiastically minded when it came to technology by saying "I think technology is good. If it can be beneficial, then it should be put to use." [Patient01 00:06:28-00:06:35].

The idea of technology-assisted PCA may not necessarily be as positively received by other groups, e.g. elderly people or people who are afraid of technology. While widespread adoption across different patient groups is

desired, a system such as the Painkiller Software Platform does not need to achieve a 100% adoption rate to be successful. In the end, pain is an individual experience and its treatment should therefore be patient-centric. This includes leaving the patient with a choice between technology-assisted PCA and nurse-administered analgesia. Furthermore, the Painkiller Software Platform does not pretend to be a universal solution as it employs a Bring-Your-Own-Device design and hence only targets users who own compatible devices.

8.3.2 Size and Diversity

Generalization of the findings from the user study to the entire population of patients and nurses is problematic due to the small size of the user study and the lack of diversity among questioned patients (both were middle-aged males). Although the study does not allow for definitive conclusions, it provides *a strong indication* that end users would find the functionality offered by the Painkiller Software Platform beneficial. Such an indication is sufficient here as present work does not claim to be an in-depth market study. The work presented here is more holistic in nature as it seeks to uncover *both* the relevance and technical feasibility of the Painkiller Software Platform – at the cost of a large-scale user study.

8.4 Missing Information

Section 5.2.2 revealed that the procedure for nurse-administered PRN analgesic medication is cumbersome. As a result, important steps are often omitted, either consciously or because of forgetfulness. Specifically, follow-up pain assessments are often forgotten. This is unfortunate as one misses out on the

opportunity to optimize pain treatment; for example by replacing a suboptimal analgesic medication with a different kind.

The Painkiller Software Platform can help decrease the number of forgotten follow-up pain registrations. The system automatically reminds the patient to reassess their pain and hence removes the mental burden imposed on the nurse. Furthermore, the system seamlessly copes with delays – e.g. in the event that the patient is asleep when it is time to reassess their pain – by logging the follow-up pain level at the time it is provided rather than at the time it was scheduled for. However, the Painkiller Software Platform is still a prototype implementation and there are some aspects of the pain assessment functionality that could be improved further. These are discussed in the following subsections.

8.4.1 Simplistic Pain Registrations

The current notion of a pain registration is too simplistic as it *only* consists of an NRS-11 pain level. The nurse explained in the interview that the pain assessment also involves asking the patient where the pain is situated and how it feels (e.g. if it is a pounding, searing, or burning pain) [Nurse01 00:14:31-00:15:33]. In addition, when asked if she had ideas for other useful information that could be collected by the Painkiller Software Platform, the nurse suggested that it would be useful if it could log side effects of the medication [Nurse01 00:57:18-00:58:27].

While this functionality did not make it into the prototype presented in this thesis, the use of JPA and the design of the data model of Painkiller Server (see sect. 6.2.2) makes it simple to extend the server-side component of the system with such additional information. Specifically, this would merely be a matter of augmenting the `PainRecord` entity with properties for the

additional information. There would be no need to manually change the database schema nor database queries as JPA automatically generates the database schema from the entities and handles the object-relational mapping.

In contrast, changing the client applications, i.e. Painkiller Patient and Painkiller Staff, to support the additional information would require substantial work. First, one would need to add new controls for specifying information about the type of pain and side effects of the medication to the UI for submitting a pain registration in Painkiller Patient. As evident from fig. 2.13b, there is no space available for the necessary extra controls in the current UI. The simplest solution would be to add additional screens, one for each piece of extra information to be collected, and present these in succession whenever the patient registers his/her pain. However, such a design may be frustrating to the user as it becomes unclear how many more steps remain before the task is complete (this is an example of the *step-by-step extreme* which Lauesen advises against [35, pp. 124-126]). A better alternative would be to redesign the current screen (fig. 2.13b) to fit the necessary extra controls by making use of expandable and collapsable content (similar to how one specifies the time and date when creating an event in the native Calendar application in iOS 10).

Second, one would also have to consider how to embed the additional information in the charts that visualize drug use and pain history. One approach could be to present the additional pain information in a popover when the user taps a pain registration data point in the chart (i.e. the blue circles in figures 2.14 and 2.15). The problem with such a design is that it defeats the purpose of the chart, namely to allow for a quick overview of the data by glancing at the chart. Alternatively, one could present the additional pain information in textual form using a label next to each data point. However, this could be hard to read because of the lines that connect each data point, especially for dense data sets. Section 5.2.4 revealed that nurses are fond of

color codes as they are already widely used in their daily work. With this in mind, a better solution could be to alter the graphical representation of a pain registration data point whenever it contains additional information besides just the pain level. For example, one could display a red cross instead of a blue circle when the patient has specified a side effect as part of the pain registration. However, it would be confusing to use a unique graphical representation for each kind of side effect as the list of possible side effects may be long. One should therefore consider combining this solution with the popover approach. The graphical representation could remain generic and simply indicate the presence of *some* side effect, while detailed information, such as the specific type of side effect, could be deferred to the popover.

8.4.2 Discarded Pain Registrations

Whenever the patient attempts to dispense a dose, the Painkiller Patient application will prompt the patient to score his/her pain and send this information to Painkiller Server as part of the request to dispense a dose. However, in the current implementation, Painkiller Server will *not* persist the pain registration if the request to dispense a dose is denied due to an ongoing lockout period. This behavior was chosen because it is in line with the RESTful design of Painkiller Server. In concordance with good RESTful style, Painkiller Server expects a request to dispense a dose to be an HTTP POST request (see table 6.1), as it is an attempt to create a new resource, and uses the status code of the HTTP response to signal the outcome of the request to the client. The status code will be `403 FORBIDDEN`[2] when the error is due to an ongoing lockout period. From a RESTful API design perspective, it would be misleading if Painkiller Server returned an error code, yet still went on to create a new resource for the pain registration bundled

[2]It is debatable whether this is the most appropriate error code. `409 CONFLICT` is also a good candidate.

in the request to dispense a dose.

In retrospect, however, the pursuit of a clean RESTful API got in the way of useful functionality. For example, consider a patient who only attempts to dispense a dose when he/she is in pain. If this patient makes an attempt to dispense a dose some time *during* the lockout period, it must mean that the medication has been unable to maintain analgesia throughout said period. If the pain level was persisted even though the dose was denied, the medical staff would be able to see exactly when (some of) the effect of the medication faded away and use this to make judgements as to whether or not to alter the treatment.

The generated pain charts (see figures 2.14 and 2.15) will also not mirror reality correctly when the system does not log pain registrations submitted as part of disapproved dispensing attempts. Consider the following scenario. Patient Alice successfully dispenses a dose, indicating that her pain level is seven. When prompted to reassess her pain 30 minutes later, Alice specifies a pain level of two. Two hours pass, and Alice suddenly starts to feel moderate pain. She attempts to dispense a dose, specifying a pain level of five in the process, but the dose is denied and the pain registration discarded since there is still one and a half hour remaining of the lockout period. Another two hours pass, and Alice successfully dispenses the second dose, while specifying a pain level of six. In this example, the chart will incorrectly indicate that Alice's pain level increased steadily from two to six, although her pain level *actually* took an early jump up to five and then slowly increased to six.

From this discussion, it is clear that the choice not to log pain registrations, submitted as part of disapproved dispensing attempts, was wrong. One should log the pain level, but deny the dose.

8.5 Future Work

This section presents a few ideas for future work. Section 8.5.1 first explains why an evaluation of the functional prototype did not make it into this work. It continues by listing ideas for how to approach such an evaluation. Section 8.5.2 suggests how treatment data, collected using the Painkiller Software Platform, could be analyzed with the purpose of improving pain treatment.

8.5.1 Evaluation of Functional Prototype

In the ideal world, the functional prototype should have been evaluated by conducting a pilot test, but this turned out to be impractical due to several reasons. First and foremost, the work presented here is only the software component for the PCOA device. In order to be able to conduct a meaningful pilot test, one would also need to build the physical dispenser. While this is an interesting challenge, it is more relevant to mechanical and/or electrical engineering than it is to software development. It could be argued that this obstacle could be circumvented by employing a Wizard of Oz approach in which the author would hand out the medication manually whenever the system would approve a dose. However, in order to be able to evaluate the usefulness of being able to track drug usage and pain development over time, the patient would need to use the system for an extended period of time (at least over night). In the Wizard of Oz approach, this would require more project members who could take turns at the patient's bedside. Second, as described in sect. 5.2, it turned out to be quite difficult to get a one-hour meeting with a professional nurse. A pilot test would be even more time consuming for the nurses and perhaps require management's approval of an increase in staffing during the test. Third, one would also need to obtain legal clearance due to the potential health hazards inherent in the experiment.

Although an evaluation of the functional prototype did not make it into this work, a few ideas for what an evaluation *could* examine are presented next.

The user study presented in ch. 5 concluded that the initial designs for the visualization of drug usage were not in line with what the users wanted. The users' requests for more granular data were incorporated into the functional prototype. This provides reassurance that the final design is rooted in the users' preferences. However, there is no guarantee that the adjusted design matches the users' expectations down to the last detail as the functional prototype was never tested. For example, the author may have misunderstood what the users said. In addition, a user may not necessarily realize the drawbacks of something she imagines to be a good idea until she gets to try it out in practice. This aspect could be evaluated in a quantitive fashion by performing a survey. Patients and nurses could be presented with screenshots of drug use and pain charts (see figures 2.14 and 2.15) generated from artificial data and asked to answer questions about the data presented in the charts (e.g. "How much Paracetamol did the patient consume between 9 AM and 4:30 PM?"). It would be beneficial to perform the survey using a digital tool such that response times could be automatically recorded. This would allow for further analysis of how long it took users to obtain the necessary knowledge from the charts.

Section 7.5.2 established that operator error is a significant source of mishaps with PCA devices. Most current PCA devices are programmed using on-device user interfaces (see for example fig. 5.2). The nurse must hence learn how to use this specialized, single-purpose UI. In contrast, the Painkiller Software Platform user interface is built using standard iOS controls. Some nurses will have experience with these controls from using iOS devices as part of their personal lives. This knowledge could possibly help them grasp the UI for configuring the PCOA device more easily. It would therefore be interesting to conduct a quantitive experiment similar to the

one carried out by Lin et al. [37] in order to test this hypothesis. Lin et al. compared the number of errors, the programming times, and the differences in cognitive workload when a group of nurses programmed a PCA pump using two different user interfaces. They found that the application of human factors to the design of the user interface of the PCA pump reduces the amount of programming errors, lowers programming times, and reduces cognitive workload. In addition, they also found that a lack of overview of the programming sequence and multi-purpose buttons deteriorate the usability. This further indicates that a UI on an iPad may be superior. First, there is more screen space available on the iPad than on a regular PCA pump (e.g. the one shown in fig. 5.2), and hence more (or perhaps even all) settings can be shown in a single screen, leaving the user with a better overview. Second, in contrast to the physical, multi-purpose buttons present on standard PCA pumps (e.g. fig. 5.2), the UI of the iPad is virtual (on-screen), and controls can therefore be updated to reflect any task-sensitive changes to their behavior.

The nurse will be responsible for introducing the system to the patient, which includes helping the patient connect his/her iPhone to the PCOA device. As the nurse described that she was by no means an IT technician (see sect. 5.2.5), this process must be as seamless as possible. To this end, the pairing procedure (see sect. 6.3.2) makes use of a QR code to prevent the need for manual entry of the IP address of the PCOA device. One way to evaluate this design could be to conduct a workshop with a couple of nurses. The pairing procedure should be explained briefly, for example by slideshow presentation or live demonstration. Nurses should then try out the pairing procedure in practice themselves, and subsequently be asked questions about how they perceived the experience (e.g. if the order of subtasks seemed natural).

8.5.2 Analysis of Treatment Data

The Painkiller Software Platform provides the necessary groundwork for the analysis of treatment data by automating the collection of drug usage and pain information. It would be interesting to investigate if the application of data mining techniques on the collected data could help guide treatment.

For example, one could define patient attributes such as surgery type, age, sex, and initial pain score (pain score when the first dose is dispensed) and then use the k-Nearest Neighbors (k-NN) algorithm to find the k previous patients most similar to the current patient. Next, one could calculate the average of the k patients' first follow-up pain scores (i.e. the pain score submitted as follow-up pain score after dispensing the first dose). One could then raise a warning if the current patient's first follow-up pain score exceeds this average by a predefined margin as this *could* indicate that the medication has suboptimal effect on this particular individual.

It could also be interesting to explore if the k-NN algorithm could be used for constructing functionality requested by users in the user study (see ch. 5). For example, a patient suggested that it would be beneficial if the PCA device could recommend when to take a dose based on an analysis of his drug usage and pain level (see sect. 5.1.3). To this end, the patient could be presented with the (average of the) pain and drug use charts of the k most similar previous patients. This could possibly give the patient an idea of how his/her pain might develop. In addition, the system could also recommend a preemptive dose x[3] minutes before each pain-peak on the average chart in an attempt to prevent pain from reaching critical levels.

[3]This should be a predefined value based on the type of medication.

9

Conclusion

Technology-assisted intravenous PCA has been practiced for decades, and medical studies have documented that patients favor PCA over conventional (nurse-administered) analgesia. In contrast, technological aids for PCOA are only just starting to appear, and their existence is unbeknownst to nurses. This is unfortunate since tablet medication is in some cases superior to intravenous medication, and since Danish hospital culture favors tablet medication over intravenous medication. In fact, PCOA is already practiced manually in Danish hospitals by allowing the patient to keep fast-acting tablets at their bedside. However, this comes at the cost of traceability and safety as there is no log of when the patient chooses to consume the medication.

A software solution for PCOA, the Painkiller Software Platform, has been proposed. The Painkiller Software Platform differentiates itself from other PCOA technology by its integration of the patient's personal smartphone. The design allows for enhanced treatment transparency – something that has been shown to be in high demand among patients – as the patient can access information about their drug usage directly on their smartphone. In addition, the fingerprint scanner on the smartphone is used as an authentication tool, which ensures that only the patient can operate the PCOA device. Finally, the smartphone automatically prompts the patient to submit information about their pain – a task that is currently carried out manually by the nurse,

and is often forgotten. Information about pain and drug usage is presented in chart form to medical staff in order to allow for more informed decision making when evaluating the efficacy of the analgesic medication.

A user study found that it is crucial that the Painkiller Software Platform is easy to operate. This presents a technical challenge as the hospital-owned PCOA device and the patient-owned smartphone are initially unaware of each others existence. A scheme that allows the two devices to locate each other and communicate securely without the need for tedious manual configuration is therefore a necessity. The Painkiller Software Platform solves this by bundling machine-readable configuration information in a QR code, and makes use of JSON Web Tokens for authentication and authorization. With this design, all the patient has to do to connect their smartphone to the PCOA device is to scan a QR code.

A risk analysis has been conducted in order to understand the consequences of making the PCOA device a networked device and employing the BYOD scheme in a hospital setting. Interestingly, the greatest risk is not health related, but rather the risk of exposure of sensitive personal information. This is because PCOA differs from intravenous PCA in that the medication is not directly injected, leaving the alert patient with a chance to spot if an excessive amount of medication is dispensed. While the benefits provided by the Painkiller Software Platform may also be relevant to intravenous PCA, the added risk of overdose due to a compromised system may not warrant them. PCOA is therefore a better candidate than intravenous PCA for early experimentation with the integration of the patient's smartphone.

In summary, the technical feasibility of integrating the patient's smartphone with the PCOA device has been demonstrated by software prototyping. Interviews with end users and a study of literature on PCA have helped identify where the design has potential for improving PCOA treatment. With

these findings in mind, the prototype demonstrates how the integration of the patient's smartphone can enhance treatment transparency, automate collection of pain information, and help ensure that only the patient can activate the PCOA device.

Bibliography

[1] AcelRx Pharmaceuticals, Inc. Zalviso. `http://www.acelrx.com/pipeline/zalviso.php`. [Online; accessed 2016-12-06].

[2] Hyemin Ahn, Hyunjun Kim, Yoonseon Oh, and Songwhai Oh. Smartphone-Controlled Telerobotic Systems. In *2014 IEEE International Conference on Cyber-Physical Systems, Networks, and Applications*, pages 77–80, Aug 2014.

[3] Jane C. Ballantyne, Daniel B. Carr, Thomas C. Chalmers, Keith B.G. Dear, Italo F. Angelillo, and Frederick Mosteller. Postoperative patient-controlled analgesia: Meta-analyses of initial randomized control trials. *Journal of Clinical Anesthesia*, 5(3):182 – 193, 1993.

[4] Jakob E. Bardram. Applications of Context-aware Computing in Hospital Work: Examples and Design Principles. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pages 1574–1579, New York, NY, USA, 2004. ACM.

[5] David Basin, Patrick Schaller, and Michael Schläpfer. *Applied Information Security: A Hands-on Approach*. Springer Publishing Company, Incorporated, 2011.

[6] David Benyon, Phil Turner, and Susan Turner. *Designing Interactive Systems: People, Activities, Contexts, Technologies*. Addison-Wesley, 2005.

[7] Andrea Cangialosi, Joseph E. Monaly Jr., and Samuel C. Yang. Leveraging RFID in Hospitals: Patient Life Cycle and Mobility Perspectives. *IEEE Communications Magazine*, 45(9):18–23, September 2007.

[8] Michael R. Cohen, Robert J. Weber, and Janet Moss. *Patient-Controlled Analgesia: Making It Safer for Patients*. Institute for Safe Medication Practices, 2006.

[9] Avancen MOD Corporation. The MOD – Oral PCA Device. `http://www.avancen.com/the-mod-oral-pca-device/`. [Online; accessed 2017-01-07].

[10] Claudio Crema, Alessandro Depari, Alessandra Flammini, Mirko Lavarini, Emiliano Sisinni, and Angelo Vezzoli. A smartphone-enhanced pill-dispenser providing patient identification and in-take recognition. In *2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) Proceedings*, pages 484–489, May 2015.

[11] Danske Medier Research. Pressemeddelelse Mobile Devices 2015 – Danskerne foretrækker smartphones frem for tablets. `http://danskemedier.dk/wp-content/uploads/2015/06/Pressemeddelelse-Gallup-11-6-2015.pdf`, June 2015. [Online; accessed 2017-03-02].

[12] Kathryn M. Davis, Matthew A. Esposito, and Bruce A. Meyer. Oral analgesia compared with intravenous patient-controlled analgesia for pain after cesarean delivery: A randomized controlled trial. *American Journal of Obstetrics and Gynecology*, 194(4):967 – 971, 2006.

[13] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor, June 1999. `http://www.rfc-editor.org/rfc/rfc2616.txt`. [Online; accessed 2017-03-06].

[14] Roy Thomas Fielding. *Architectural Styles and The Design of Network-Based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[15] Wireshark Foundation. Wireshark. https://www.wireshark.org/, 1998. [Online; accessed 2017-02-26].

[16] Saul Greenberg, Michael Boyle, and Jason Laberge. PDAs and shared public displays: Making personal information public, and public information personal. *Personal Technologies*, 3(1):54–64, 1999.

[17] Java Persistence 2.1 Expert Group. *JSR 338: Java^{TM} Persistence API, Version 2.1*. Oracle America, Inc., April 2013.

[18] John Halamka. Early Experiences with Positive Patient Identification. *Journal of Healthcare Information Management*, 20(1):25–27, 2006.

[19] Daniel Heß and Christof Röhrig. Remote Controlling of Technical Systems Using Mobile Devices. In *2009 IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pages 625–628, Sept 2009.

[20] Jana Hudcova, Ewan D McNicol, Cheng S Quah, Joseph Lau, and Daniel B Carr. Patient controlled opioid analgesia versus conventional opioid analgesia for postoperative pain. *The Cochrane Library*, 4(Art. No.: CD003348), 2006.

[21] Apple Inc. iOS Security Guide. Technical report, Apple Inc., May 2016.

[22] Apple Inc. Keychain Services Concepts. https://developer.apple.com/library/content/documentation/Security/Conceptual/keychainServConcepts/02concepts/concepts.html, September 2016. [Online; accessed 2017-03-13].

[23] Apple Inc. About Swift. https://swift.org/about/, 2017. [Online; accessed 2017-03-29].

[24] Apple Inc. Find My iPhone. http://www.apple.com/icloud/find-my-iphone.html, 2017. [Online; accessed 2017-02-05].

[25] Institute for Safe Medication Practices. Safety issues with patient-controlled analgesia, Part I: How errors occur. https://www.ismp.org/newsletters/acutecare/articles/20030710.asp, July 2003. [Online; accessed 2017-04-27].

[26] Dorothy Jackson. A Study of Pain Management: Patient Controlled Analgesia Versus Intramuscular Analgesia. *Journal of Intravenous Nursing*, 12(1):42–51, 1989.

[27] M. Jones, J. Bradley, and N. Sakimura. JSON Web Signature (JWS). RFC 7515, RFC Editor, May 2015. http://www.rfc-editor.org/rfc/rfc7515.txt. [Online; accessed 2017-03-10].

[28] M. Jones, J. Bradley, and N. Sakimura. JSON Web Token (JWT). RFC 7519, RFC Editor, May 2015. http://www.rfc-editor.org/rfc/rfc7519.txt. [Online; accessed 2017-27-02].

[29] Mark Joselli, José Ricardo da Silva, Marcelo Zamith, Esteban Clua, Mateus Pelegrino, Evandro Mendonça, and Eduardo Soluri. An Architecture for Game Interaction using Mobile. In *2012 IEEE International Games Innovation Conference*, pages 1–5, Sept 2012.

[30] Patti Kastanias, Kianda E. Snaith, and Sandra Robinson. Patient-Controlled Oral Analgesia: A Low-Tech Solution in a High-Tech World. *Pain Management Nursing*, 7(3):126–132, 2006.

[31] Mike Keith and Merrick Schincariol. *Pro JPA 2*. Apress, New York, NY, USA, 2nd edition, 2013.

[32] Ross Koppel, Tosha Wetterneck, Joel Leon Telles, and Ben-Tzion Karsh. Workarounds to Barcode Medication Administration Systems: Their Occurrences, Causes, and Threats to Patient Safety. *Journal of the American Medical Informatics Association*, 15(4):408, 2008.

[33] Andrew Krioukov and David Culler. Personal Building Controls. In *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, IPSN '12, pages 157–158, New York, NY, USA, 2012. ACM.

[34] Andrew Krioukov, Stephen Dawson-Haggerty, Linda Lee, Omar Rehmane, and David Culler. A Living Laboratory Study in Personalized Automated Lighting Controls. In *Proceedings of the Third ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '11, pages 1–6, New York, NY, USA, 2011. ACM.

[35] Soren Lauesen. *User Interface Design: A Software Engineering Perspective*. Addison-Wesley, Boston, MA, USA, 2005.

[36] Jangho Lee, Jee-In Kim, Jiyong Kim, and Ji-Young Kwak. A Unified Remote Console Based on Augmented Reality in a Home Network Environment. In *2007 Digest of Technical Papers International Conference on Consumer Electronics*, pages 1–2, Jan 2007.

[37] Laura Lin, Racquel Isla, Karine Doniz, Heather Harkness, Kim J. Vicente, and D. John Doyle. Applying Human Factors to the Design of Medical Equipment: Patient-Controlled Analgesia. *Journal of Clinical Monitoring and Computing*, 14(4):253–263, 1998.

[38] Silvano Maneck Malfatti, Fernando Ferreira dos Santos, and Selan Rodrigues dos Santos. Using Mobile Phones to Control Desktop Multiplayer Games. In *2010 Brazilian Symposium on Games and Digital Entertainment*, pages 230–238, Nov 2010.

[39] N. Matsushita, S. Tajima, Y. Ayatsuka, and J. Rekimoto. Wearable Key: Device for Personalizing nearby Environment. In *Digest of Papers. Fourth International Symposium on Wearable Computers*, pages 119–126, Oct 2000.

[40] Corey McCall, Branden Maynes, Cliff C. Zou, and Ning J. Zhang. An automatic medication self-management and monitoring system for independently living patients. *Medical Engineering & Physics*, 35(4):505 – 514, 2013.

[41] Timothy I. Melson, David L. Boyer, Harold S. Minkowitz, Alparslan Turan, Yu-Kun Chiang, Mark A. Evashenk, and Pamela P. Palmer. Sufentanil Sublingual Tablet System vs. Intravenous Patient-Controlled Analgesia with Morphine for Postoperative Pain Control: A Randomized, Active-Comparator Trial. *Pain Practice*, 14(8):679–688, 2014.

[42] Brad A. Myers. Using Handhelds and PCs Together. *Communications of the ACM*, 44(11):34–41, November 2001.

[43] Brad A. Myers. Using handhelds for wireless remote control of PCs and appliances. *Interacting with Computers*, 17(3):251, 2005.

[44] Alex Nichol. JamWiFi. `https://github.com/unixpickle/JamWiFi`, 2012. [Online; accessed 2017-02-23].

[45] Jeffrey Nichols and Brad A. Myers. Controlling Home and Office Appliances with Smart Phones. *IEEE Pervasive Computing*, 5(3):60–67, July 2006.

[46] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol. Generating Remote Control Interfaces for Complex Appliances. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, UIST '02, pages 161–170, New York, NY, USA, 2002. ACM.

[47] Hui-Kyung Oh and In-Cheol Kim. Hybrid Control Architecture of the Robotic Surveillance System Using Smartphones. In *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 782–785, Nov 2011.

[48] Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065 USA. *JAX-RS: Java^{TM} API for RESTful Web Services*, May 2013.

[49] JuGeon Pak and KeeHyun Park. Construction of a Smart Medication Dispenser with High Degree of Scalability and Remote Manageability. *Journal of Biomedicine and Biotechnology*, 2012, 2012.

[50] Københavns Politi. Redegørelse vedr. Københavns Politis Virksomhed 2012, 2013.

[51] Marek Potociar and Santiago Pericas-Geertsen. *ContainerRequestFilter (Java(TM) EE 7 Specification APIs).* Oracle Corporation, June 2015. `https://docs.oracle.com/javaee/7/api/javax/ws/rs/container/ContainerRequestFilter.html`. [Online; accessed 2017-03-12].

[52] Jun Rekimoto. A Multiple Device Approach for Supporting Whiteboard-based Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, pages 344–351, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[53] Scott Robertson, Cathleen Wharton, Catherine Ashworth, and Marita Franzke. Dual Device User Interface Design: PDAs and Interactive Television. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '96, pages 79–86, New York, NY, USA, 1996. ACM.

[54] N. D. Rodríguez, J. Lilius, S. Björklund, J. Majors, K. Rautanen, R. Danielsson-Ojala, H. Pirinen, L. Kauhanen, S. Salanterä, T. Salakoski, and I. Tuominen. Can IT health-care applications improve the medication tray-filling process at hospital wards? An exploratory study using eye-tracking and stress response. In *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pages 423–428, Oct 2014.

[55] Jane Rosati, Mary Gallagher, Beverly Shook, Edward Luwisch, Gregory Favis, Ruby Deveras, Abdul Sorathia, and Sharon Conley. Evaluation of an Oral Patient-Controlled Analgesia Device for Pain Management in Oncology Inpatients. *Journal of Supportive Oncology*, 5(9):443–448, Oct 2007.

[56] Jim Rudd, Ken Stern, and Scott Isensee. Low vs. High-fidelity Prototyping Debate. *Interactions*, 3(1):76–85, January 1996.

[57] Cátia Santos-Pereira, Alexandre B. Augusto, Manuel E. Correia, Ana Ferreira, and Ricardo Cruz-Correia. A Mobile Based Authorization Mechanism for Patient Managed Role Based Access Control. In Christian Böhm, Sami Khuri, Lenka Lhotská, and M. Elena Renda, editors, *Information Technology in Bio- and Medical Informatics: Third International Conference, ITBAM 2012, Vienna, Austria, September 4-5, 2012. Proceedings*, pages 54–68, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[58] Micael Sjölund, Anders Larsson, and Erik Berglund. Smartphone Views: Building Multi-device Distributed User Interfaces. In Stephen Brewster and Mark Dunlop, editors, *Mobile Human-Computer Interaction - MobileHCI 2004: 6th International Symposium, MobileHCI, Glasgow, UK, September 13 - 16, 2004. Proceedings*, pages 507–511, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[59] Lauretta N. Stubbs and Diane Monsivais. Patient Controlled Oral Analgesia Delivery Device Promising Technology to Improve Pain Control in Acute Care Settings. *Online Journal of Nursing Informatics*, 18(3), Oct 2014.

[60] Sunhedsstyrelsen. Patientidentifikation ved brug af identifikationsbånd, June 2008.

[61] P. H. Tsai, T. Y. Chen, C. R. Yu, C. S. Shih, and J. W. S. Liu. Smart Medication Dispenser: Design, Architecture and Implementation. *IEEE Systems Journal*, 5(1):99–110, March 2011.

[62] Southern Cross University. Risk Likelihood and Consequence Descriptors. `http://scu.edu.au/risk_management/index.php/4`, 2017. [Online; accessed 2017-02-18].

[63] Tamas Vajk, Paul Coulton, Will Bamford, and Reuben Edwards. Using a Mobile Phone as a "Wii-like" Controller for Playing Games on a Large Public Display. *International Journal of Computer Games Technology*, 2008, 2008.

[64] Janus Varmarken. A Patient-Controlled Oral Analgesia Device for Inpatients, 2016. [Exam assignment for Thesis Preparation SDT, 2016 (KSTPSDT1KU)].

[65] Kim J. Vicente, Karima Kada-Bekhaled, Gillian Hillel, Andrea Cassano, and Beverley A. Orser. Programming errors contribute to death from patient-controlled analgesia: case report and estimate of probability. *Canadian Journal of Anesthesia*, 50(4):328, 2003.

[66] Eugene R. Viscusi and Leslie N. Schechter. Patient-controlled analgesia: Finding a balance between cost and comfort. *American Journal of Health-System Pharmacy*, 63(8 Supplement 1):S3–S13, 2006.

[67] Paul F. White. Mishaps with Patient-controlled Analgesia. *Anesthesiology*, 66(1):81–82, 1987.

[68] Elsa Wuhrman, Maureen F. Cooney, Colleen J. Dunwoody, Nancy Eksterowicz, Sandra Merkel, and Linda L. Oakes. Authorized and Unauthorized ("PCA by Proxy") Dosing of Analgesic Infusion Pumps: Position Statement with Clinical Practice Recommendations. *Pain Management Nursing*, 8(1):4 – 11, 2007.

Appendices

A

Interview Schema: Patient

A.1 Introduction

Hello, my name is Janus Varmarken. I am a student at the IT University of Copenhagen. I am developing an IT system that makes it possible for you as a patient to self-administer your analgesic medication using your smartphone. The system will also provide functionality that lets you and the nurse track your drug usage. I know from talking to the nurse that you have recent experience using similar technology. May I ask you a couple of questions about this experience, please? The purpose is to identify what works well and where there is room for improvement such that I can make the right decisions when building my system.

Before we start, I need to ask you if you understand and accept that your responses will be used in my thesis, and if it is okay that I make an audio recording of the interview? Please let me know now if you prefer to remain anonymous.

A.2 Technology-Assisted PCA

1. What are your thoughts on self-administration of analgesic medication using technological aids? [Questions that can be asked to get the conversation going or as follow-up questions:]
   (a) Do you have any concerns about the safety of such treatment?
   (b) There will obviously be less interaction with the nurse. How does that affect the treatment? Does it make the treatment feel impersonal?
2. Would you prefer to self-administer your analgesic medication using technological aids, or do you prefer that the nurse is in charge of administering the analgesic medication? Why?
3. How was your experience using the PCA pump for pain treatment? [Questions that can be asked to get the conversation going or as follow-up questions:]
   (a) What did you like, and what could have been better?
   (b) Do you have experience with nurse administered analgesic medication? If yes, how do the two compare?
   (c) How efficient was the treatment in terms of pain relief?
4. Did you feel safe using the PCA pump? Why/why not?

A.3 Treatment Transparency

1. What information about the treatment did the PCA pump make available to you as a patient (e.g. drug usage, dose availability)?
   (a) If any: how was it presented and how useful was it?

2. Do you consider it important to have insight into and/or co-responsibility for your own treatment? Why?

3. Would it be beneficial if the PCA pump could provide you with an overview of your drug usage? Why/why not? If yes:

   (a) How would you like to access such information?

   (b) How would you like the information presented? [Show sketches and ask for interviewee's opinion.]

A.4 Security

1. Have you thought about the security/safety of the PCA pump? For example, I could pick up the remote control and activate it against your will. I am working on a solution that can prevent this from happening. Would that add any value, or do you consider such a scenario unlikely?

B

Interview Schema: Nurse

B.1 Introduction

Thank you for taking time to talk to me.

I am developing a system that allows the patient to safely self-administer her analgesic medication using her smartphone. The system provides functionality that lets the patient and you, the nurse, track the medication usage. I know that you are currently using a similar system (PCA pumps) as part of a medical trial. I would like to ask you some questions about your experiences with this system in order to uncover what works well and what could be improved.

Please interrupt me during the interview if something is unclear. Please also rest assured that I will understand if we have to end or pause the interview abruptly as I aware of the volatile nature of your working environment. I do not want to get in the way of your work.

Before we start, I need to ask you if you understand and accept that your responses will be used in my thesis, and if it is okay that I make an audio recording of the interview? Please let me know now if you prefer to remain anonymous.

B.2 Technology-Assisted PCA and Treatment Data

1. Can you please explain to me what you do here? What is a normal day like? What are your daily tasks as a nurse in this ward?

2. One of the primary goals of my work is to make the administration of PRN analgesic medication easier and safer. Can you explain to me what challenges you face when you administer medication (both manually and using technology-assisted PCA)? What works well, and what could be improved?

3. How is the PCA pump configured (physical set up and programmatic configuration)? [If interviewee has experience with this] Are there aspects that are particularly difficult or require a high level of concentration?

4. What information about the treatment does the PCA pump provide? Does it provide you with insight into the patient's drug usage and/or pain history?

   (a) If yes:

      i. How do you access this information? Do you make use of the information? Why/why not?

      ii. Does this work well, or could you imagine a better way of accessing the information?

      iii. Is the information presented such that it is easy to understand? Can you imagine better ways of representing the information? [Show sketches and ask for interviewee's opinion.]

      iv. Can you think of any information that could be useful in your daily work and/or help optimize the pain treatment, but is currently missing?

(b) If no:

i. What information could be useful in your daily work and/or help optimize the pain treatment?

ii. How would you prefer to access this information? (For example monitor in your office, handheld device, monitor at the patient's bedside,...)

iii. How would you like the information to be presented? [Show sketches and ask for interviewee's opinion.]

5. (How) do you tailor treatment to match the specific needs of each individual patient? Could data about drug usage and pain history be of any use in that regard?

B.3 Conventional Analgesia

1. It is my understanding that one does not use PCA (equipment) for all kinds of PRN analgesic medication (e.g. tablets). Is this correct, and if so, can you please explain the procedure for handling such medication?

2. What is the reason that PCA (equipment) is not used in these cases?

B.4 Security and Safety

Low priority questions, may be skipped if there is no time for them.

1. How is the PCA pump secured against abuse/tampering?

2. Can overdose occur if you make a mistake during the configuration process?

3. Could I walk in an pick up the remote control and use it to give the patient a dose of medication against her will? If yes: do you consider this a security issue, or is it an unlikely thought experiment?

4. What prevents physical manipulation of the PCA pump (e.g. adjustment of dose size, theft of medication)?

5. How is the patient data, if any, secured against third party access?

6. Have you considered how the PCA pump could be made more safe/secure?