What we are looking for in each paper:

1. Background info on IoT
2. Background info on any domain of IoT
3. Generic IoT patterns
4. Domain Specific IoT patters
5. Usages of non-IoT patterns in IoT

Add a commend on any line that is one of these saying which it is and any pertinent information.

Use <https://www.naturalreaders.com/online/> to read the articles aloud while following along.

Add at least one diagram for each of the subdomains.

Domains that I will write about:

Smart Water Management System

BCI

Health Care

-

Smart Metering

Smart Grids

Industrial IoT (such as Smart Cities)

Notes on the articles:

nf

Design Patterns for the Industrial Internet of Things:

* lots of general information about IoT and it’s structure
* several design patterns used in some domains

Fog and IoT- An Overview of Research Opportunities

* networking architecture pattern and reasons for it
* also includes information about general needs of IoT as it grows and limitations of current IoT

mp

An Integrated IoT Architecture for Smart Metering:

* information Smart meters (electricity, water, gas)
* the articles proposition on a system for it (including benefits for the utilities and customers)

IoT Architecture for Smart Grids:

* talks about why we need smart grids (benefits)
* lists lots of types of energy sources and how they connect to IoT (like distribution to local areas, micrograms, smart cities, building, etc.)
* ^as well as storing energy to fix the fluctuations in these energy generation types

Study of IoT - Understanding IoT Architecture, Applications, Issues and Challenges:

* has a lot of domains of IoT (also divided into sub-domains)
* general information about IoT
* pros and cons of IoT
* challenges of IoT

Decentralised IoT Architecture for Efficient Resources Utilisation:

* like the Fog one it talks about removing some of the work from the could due to the increasing work load on them
* ^offers solutions for that issue (such as a more intelligent edge device and more)

p

Dependable design for elderly health care

* Healthcare of the elderly, trustworthiness
* (check the main one to see what patterns are in it)

A simple security architecture for smart water management system:

* Issues with security regarding smart water (for instance the option of physically accessing the devices and not just hack into it computing)
* some general architectural patterns regarding their solution
* security solutions

BCI ontology- A context-based sense and actuation model for brain-computer interactions:

* general info on BCI
* design patterns for BCI
* ontology stuff… (conforming to existing things and so on …)
* integration with AI

Cataloging design patterns for Internet of Things artifact integration:

* a lot of design patterns in general and how they made IoT patterns from existing things
* (I think these can go in the general IoT category)

IoT design patterns- Computational constructs to design, build and engineer edge applications:

* several of general IoT patterns

BCI –

BCI, that is Brain-Computer Interfaces, is quite an interesting domain of IoT connecting neurology, electrical engineering, machine learning in addition to some of the more standard aspect of IoT such as networking connectivity.

Using various devices for sensing electromagnetic waves the body, and specifically the brain (emitting brain-waves), sends out, such as and electroencephalogram (EEG), it analyzes these electromagnetic waves and classifies them using machine learning algorithms to train the associated software making it “understand” the intention behind those waves. This is usually done by a device such as an EEG coming into physical or near physical contact with the body. After classification by the machine learning (generally deep learning) software actuation is required.

Although this may seem like a futurist idea, perhaps something out of a science fiction novel, real world examples have already been developed. They range in purpose and field, from military to healthcare, not to mention entertainment.

Although we will mainly talk about healthcare in the remainder of this paper, we will deviate from that for one example to solidify an image of what BCI can do. This example may cause and upset in the world of gaming. The NextMind device with its SDK for the Unity game engine allows users, when wearing the device, to control certain aspects of project created with then Unity game engine simply by concentrating. It works by analyzing the brainwaves of the wearer when concentrating on special graphics displayed on the screen and with integrating the software this can be used to cause a myriad of effects, simplest of which would be perhaps movement in a game. This is not limited to a simple trigger of is the user is concentrating on the visual queue either, it can analyze the intensity of the concentration. Via software interface any effect can be given to the values read from the brain.

Now that we understand some of the capabilities of BCI we will see in the field of healthcare that the same type of technology isn’t limited to video games. It can be used for instance to move around a wheelchair of a patient who does not have the ability to walk. Once the software is trained for the patient and a classifier is created, it can be used to analyze the patient’s brainwaves and interpret how to actuate and move the wheelchair. Here the IoT technology is allowing all the various parts to communicate with each other such as the EEG device, the actuated device (the wheelchair) and the processing unit which may be on a separate device.

To more efficiently work with BCI as a subdomain of IoT ontology design patterns were developed. Here we will look at a couple, both targeting compatibility and maintainability as ontology patterns.

* Actuation-Actuator-Effect Ontology Design
* Stimulus-Sensor-Observer Ontology Design

Without going into detail of the tagging and models of the ontology itself, these two patterns work hand in hand cataloging the full picture of BCI model, that is the connection between the Sense Model (described by the Stimulus-Sensor-Observer Ontology Pattern) and the Actuation Model (described by the Actuation-Actuator-Effect Ontology Design). Each pattern is aligned to and expands upon existing ontologies in the relevant fields including, of course, the field of IoT.

In the Figure below we can see the relationship between the two patterns:



Figure [fill in citation: BCI Ontology: A Context-based Sense and Actuation Model for Brain-Computer Interactions]

Smart Resource Distribution –

With the emergence and popularization of IoT vital resource (i.e. gas, water and electricity) distributors saw a tremendous opportunity in integrating the new technology into their systems. IoT based devices for monitoring and distributing resources were developed.

Major components of these system are:

* Smart Meters, the devices responsible for transmitting the state of each the meter in the system as well as actuating on the meter when receiving such a command,
* System specific hardware (sometime part of the same device as the meter), these are used to actuate on components of the system such as windmills or water pumps.
* Control Centers, the computers where the data is consolidated and analyzed and then decisions (made by computers or people) are made such as changing the state of an edge device.

These meters can be spread out across wide areas from densely populated to sparse rural areas. As we will see in a later, certain security issues arise from such setups. Many systems are in fact sub-systems of a larger overarching system. As such, a

The responsibilities of such systems may include:

* Load balancing. This lowers maintenance requirements by monitoring the edge devices and sending data to a control center. That center will compare the loads and send command back to the edge devices adjusting the loads. The edge devices will then actuate on the hardware, adjusting it to the specifications provided by the control center.
* Load diversion. This lowers lowers the impact of failures. By monitoring the state of edge devices in an interconnected Smart Meter system, if a failure is detected control can channel the flow of the resource from a different source. In such cases the end user may never know that a failure took place.
* Failure and maintenance alerts. This too lowers maintenance demands as the edge device will alert the command center of failures or when it requires maintenance. Instead of routinely doing maintenance for all devices, only those that need attention can be pinpointed and dealt with.
* Efficacy evaluation. This can save great costs on the system. A simple example of this is making sure that solar energy farm providing as much energy as possible by tilting the solar panels to follow the sun to generate as much energy as possible. This can greatly increase the benefit of the benefit to cost ratio.

As one of the most varied resources we’ll specifically mention smart resource management systems for electricity. Such a is commonly referred to as a Smart Grid. A plethora of types of Smart Grids exist around the world, dealing with different energy sources. Aside traditional sources such as fossil fuel Smart Grids for green energy sources such as wind and solar have been a major source of development as well. In fact, several such such Smart Grids may work together forming a larger grid deriving energy from various sources.



Figure 2 [citation for: IoT Architecture for Smart Grids]

As mentioned above, there is a major security issue with smart resource or smart meter systems. In addition to networked cyber-attacks, the threat of which all network systems face, in this case there is a fear of physical attacks. Some of the meters, which communicate with the control center, are spread out in vast areas out and cannot feasibly be constantly monitored. As such, they are left open to physical attacks. We are not referring to simply damaging the devices, we are referring to maliciously tampering with them causing intentional false reading and sending false information to the control center.

Three elements were identified that needed to be secured:

* Booting, prevents unauthorized devices from functioning.
* Communications, prevents unauthorized communication.
* Firmware updates, prevents unauthorized firmware from being installed.

Though none of these can physically safeguard the device, together they prevent a device that has been tampered with from hurting the system aside from that particular device being rendered not functional.

To address these three elements one method utilizes design patterns both existing and new. With these patterns, solution to the three elements above are provided using a combination of public key cryptography where time isn’t vital (booting and firmware updates), stateless authentication where time is of the essence (regular communication) along with offloading heavy security calculations so as not to hard the real-time nature of the edge devices.

These new design patterns developed by this method:

* Computation Offloading: Due to the nature of the edge devices heavy computational tasks would harm their efficacy, especially for real-time devices, therefore these tasks will be offloaded to trusted gateways where the heavy computation will take place.
* Stateless Authentication of edge devices: Similar to existing stateless authentication such as in some smartphone apps, the device will use a token for authentication and no session data will need to be stored.

Health Care –

Healthcare has been radically transformed in recent years with the integration of IoT technologies. Various aspects of the healthcare domain have been affected from app connected to peripherals monitoring one’s basic vitals while exercising, counting the amount of steps a person walked throughout the day, week or month and so much more. Although a lot of these may seem like they are for the utility of generally healthy users allowing them to keep track of themselves, that only scratches the surface of what is now possible in this domain.

Using IoT systems hospital staff can monitor at-risk patients either on site or even from the comfort of their homes. Capturing data of general vitals such as pulse or blood pressure as well as more case specific data such as brain function, network enabled devices can be worn by, or sometimes implanted in, patients where the data is collected and sent out either directly to experts or to a management center. The collected data is analyzed and can sent out alerts a situation warrants. Given a relevant situation a healthcare professional may give instructions to an on or off site patient using speakers, meaning that a patient at home may be able to receive immediate medical advice from a professional when they need it.

These healthcare management centers may also employ the use of prediction models trained using artificial intelligence algorithms on dig data to attempt to predict diseases and doing so help in reaching effective treatments.



Figure [citation for: Challenges and opportunities in IoT healthcare systems- a systematic review]

All this advancement, though positive overall, bring a new set of challenges to the healthcare practitioner and patients alike. Although we will focus on the patient one of the larger concern’s from the practitioner’s perspective is a question about the data’s quality may be called into question as noise may added during the transfer. This can be mitigated with better architecture and noise removal techniques.

From the patient’s perspective we see very different types of questions. Some of the new solutions require patients to give up a certain amount of confidentiality to allow their data to be processed and used by the systems which may cause doubt and suspicion on the side of the patients. Some systems require patients be tech-savvy to a certain degree. Generally, these are considered minimal but for some patients who do not use much technology at all, this can be quite the hurdle. Another concern is that some patients simply distrust technology on and security or stability. These can cause great discontent among patients when presented with new healthcare system. A prime example would be healthcare for the elderly. The elderly is demographic that is not used to dealing with technology and find it intimidating and untrustworthy. Coupled with the fact that the elderly as a demographic require more attention from the healthcare systems, a demand to better the situation arose.

To address these issues a number of safeguards can be setup. First of all, patients may specify what information they will share as well as who has access to it and how that information will be used, as such making the system more trustworthy to the patients. To enforce this requirement, the software run in the healthcare systems must support this link between information and role and only allow access to those with an allowed role. Exiting design patterns used for this are the Role Based Access Control pattern and the Contextual Based Role Based Asses Control. Both of these deal with giving access to specific information based on the role of the accessor and the context of the access (in our case the context may be an emergency and so access will be provided even for roles not generally authorized if the situation allows it).

The second point of interest regarding the trustworthiness of the the system is the sensors themselves and the data they gather. As mentioned above many different sensors may be used in healthcare systems be them wearables, implants or any other type of sensor. These sensors collect data from the patient which is later used for analysis. In order to protect this data, the Sensor Design Pattern was introduced. Including authorization techniques, it protects the data within the sensor from being access by unauthorized parties. In addition, it dictates that minimal data is stored in the sensor devices themselves, most of the data stored being what is necessary to authentication and collected data is sent into the system and stored there. This means that even if the sensor device is stolen, past collected data cannot be extracted from it.

General IoT Networking Patterns – [the networking protocols D2D and so on…]

IoT is a rapidly growing technology with a main focus on data collection and analysis then reacting to the results of this analysis. The data is collected from sensor edge devices and sent over a network to a control hub of some sort where the majority of the analysis takes place and instructions may be sent back over a network to actuate on the decisions made from the analysis. Just in this short and simplistic description of IoT we see that the network transfer is a major component of IoT. This should be obvious when considering the acronym is for Internet of Things implying many devices interconnected within a network. As such, a questions arises. How is all this data sent over a network?

In order to answer the question, we must ask another. Between what parties is data being transferred. Depending on the sender and receiver the transfer method or protocol may vary. To support these needs design patterns were developed for various networking types or layers. The layers we will discuss are Device to Device (D2D), Network to Network (N2N), Middleware to Middleware (MW2MW), Application and Service to Application and Service (AS2AS), Data and Semantics to Data and Semantics (DS2DS) and CROSS-Layer. We will not talk about all the layers here but it is useful to know what layers have been looked into.

We will list some of the derived design patterns:

* IoT Gateway Event Subscription pattern. This pattern from the D2D layer dictates that the gateway is used as a subscription mechanism allowing different IoT devices to communicate with each other. Various devices will send messages to the gateway. The devices can subscribe to only certain messages and the gateway will forward messages to subscribes devices of the message type. This decouples the devices as a device does not know what devices if any are listening to its messages, and on the other side a device does not know what devices are send the messages it is listening to.
* D2D REST Request/Response Pattern. Another pattern of the D2D layer in which the IoT devices should be able to communicate using HTTP/REST such that when a request is sent a response will be sent back.
* IoT artifact’s Middleware Message Broker. This MW2MW layer pattern dictates that there will be a common messaging interface between the various components of the middleware. This allows the components to interact regardless of the component’s design. Each component interacts with the broker which, in turn, interacts with the relevant components.
* IoT Artifact’s Middleware Self-contained Message. Another MW2MW pattern for message structure. It states that each message should provide all necessary information needed for a given action. This allows the middleware components to contain only little information and still be able to successfully route and act on the message.
* IoT SSL CROSS-Layer Secure Access. This CROSS layer pattern deals with the systems security. In a broad IoT network, one consisting of multiple layers, all access to layers and interactions between them must be secured. With this pattern each layer will have a public API only through with it can be accessed and interacted with and all such interactions will be done using SSL (Secure Socket Layer).

[section for Design Patterns intro]

This paper deals with IoT and sub-domains as well as their effect on architectural and design pattern. Before we can delve into those subjects though a basic understanding of the topics is necessary. We will start with a short instruction to design/architectural patterns and move on to IoT and then talk about a selection of its sub-domains.

Design patterns were popularized in the field of software engineering in the now famous book by the Gang of Four where a variety of patterns were described. The job of each pattern was to provide a generalized solution for recurring problems. The design patterns described in the book were object oriented patterns and they solved software development issues that showed up in that paradigm such as how listen to an event with the Observer pattern or how to enforce a single instance of class that can be accessed anywhere with the Singleton pattern. In all 24 patterns were listed and categorized into 3 pattern types, Creational, Structural and Behavioral.

These design patterns as well as others for other sources made a great impact on the realm of software engineering and were expanded in existing categories and new ones. Other well-known categories include Concurrency patterns dealing with recurring multi-threaded environment problems such as the Producer-Consumer pattern, Architectural patterns dealing with more macro architecture issues such as how to build a robust and maintainable app such as MVC (Model View Controller). Here we will note the distinction between architectural patterns and design patterns. They are similar in nature and the main difference is level of abstraction. Design patterns are more specific solving smaller scale problems while architectural patterns are more abstract and solve larger overarching problems. In the paper we may use the word “pattern” by itself to refer to either.

With the development of software engineering more specialized patterns were developed, still dealing with providing a general solution to a common problem with the exception that the problem in question was common within a specific domain and not necessarily the larger field of software engineering. In this paper we are interested to see how various pattern domains evolved due the introduction and popularization of IoT and its sub-domains.

[section for IoT intro]

We would now like to introduce IoT.

The subject of this paper, the Internet of Things or IoT put simply is the technological field dealing with interconnected devices over a network. These devices range from a common electric kettle to the cutting edge of green energy wind turbine and everything in between. In fact, it is no exaggeration that the limits of IoT aren’t bound to the Earth itself and has taken flight to the bounds of spaces.

IoT is often thought of as a new up and coming technology and although it certainly has experienced growth spurt in the last decade or so, it can be seen as far back as the 1980’s, from before the term was even coined, in old vending machines. What most see and an internet buzz word some hail as a major component of the fourth Industrial Revolution. As such I would like to discuss why IoT is such an important in today’s world and even more so in the world of tomorrow. What can it do to further society, what fields does it progress?

Like any field especially one so large and diverse there many ways to get a job done, some efficient than others. We learn from our experiences and device better mays to design and develop in the field.

IoT hardware and software component and I would like to focus on the software side. Software design or soft architecture is a vast and highly discussed field in and of itself. It is imperative model our software in a way that is robust, easy to maintain and flexible among many other traits. Software as a general field has much experience with this since the Gang of Four and the original wide spread design patterns became well known. The job of these design patterns was to allow software to be built in such a way that would counter problems that has come up in somewhere to that point, improving the quality and maintainability by a significant margin, but it didn’t stop there. More and more patterns were recognized and as spread of software engineer spread specialized patterns showed up, that is to say, ways of designing software that fit a specific field or sub-field.

IoT is no stranger to this. As the field of IoT developed, software patterns that were specific to IoT appeared one after another, furthermore, patterns for the fields or domains within IoT developed patterns specific to themselves.

Note the term “pattern” that I have been using has been used somewhat ambiguously. Here I am using it to mean an identified design rather than specifically an architectural pattern or design pattern. The difference between the two being their level of abstraction. In fact, some go so far as to include architectural styles under the same umbrella. The difference between a pattern and a style is that a pattern is meant to solves a specific problem while a style is simply a preference for how to compose software. As both deal with software design they may be lumped together as long as it is understood what “pattern” may be referring to. As such there are three levels of abstraction of patterns where architectural styles are the most abstract deal with the general idea of the software but no specifics, architectural patterns are in the middle, dealing with the specific design of the larger systems of the software and design patterns the least abstract specifying the design of individual components.