

NASHVILLE, TENNESSEE | AUGUST 5-9, 2013



Adapting Agile Methodology to Overcome Social Differences

- Motivating Members by Fostering Individuals Interactions

Hitoshi Ozawa and Lan Zhang

[OGIS-RI Co., Ltd.](#)

August 5, 2013

2:00PM - 3:15 PM

Room: Governors E

Are you here to find a solution to succeed in your agile project?

Maybe, that's not the question I should be asking.

Our experience report may show you why.

A word of caution before starting this presentation.

First, there's no mention of

Second, I just flew in from Tokyo, Japan yesterday. 2pm here is 4am in Tokyo. I'll try not to fall asleep before you do but please forgive me if I do.

Let me tell you about our story  
of transition from a traditional company



**Wholly subsidiary of Osaka Gas**  
– the second largest natural gas company in Japan

**Over 3,000 employees**

**Capital: US\$4.4 million**

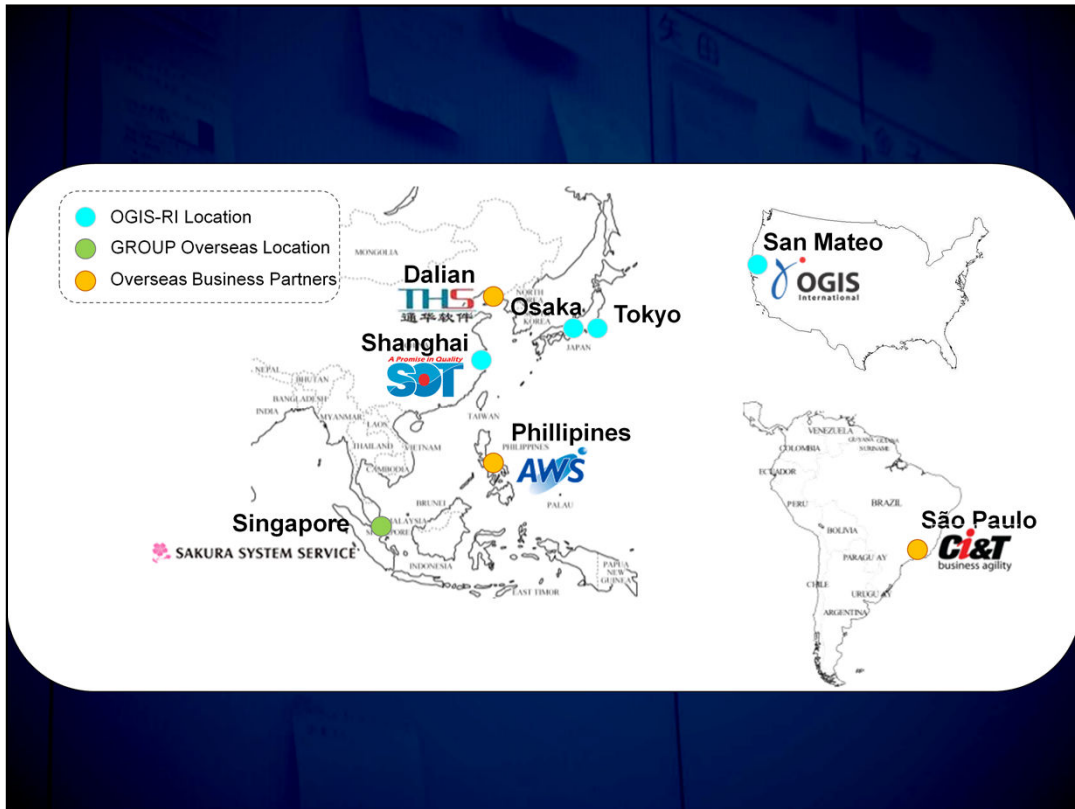
**Revenue: US\$668 million**

I'm working with OGIS-RI. OGIS-RI stands for Osaka Gas Information System Research Institute.

OGIS-RI used to be IT department of Osaka Gas before it was spun out.

As Osaka is the second largest city in Japan, Osaka Gas is the second gas company in Japan servicing many households.

Consolidate number of employee is greater than 3,000 so we'll not a small company.



Our subsidiary includes Sakura Information System and Ube Information System

We are have offices and partners in other countries providing not only software development, but business process outsourcing, training and support services as well.

# Our Agile initiative

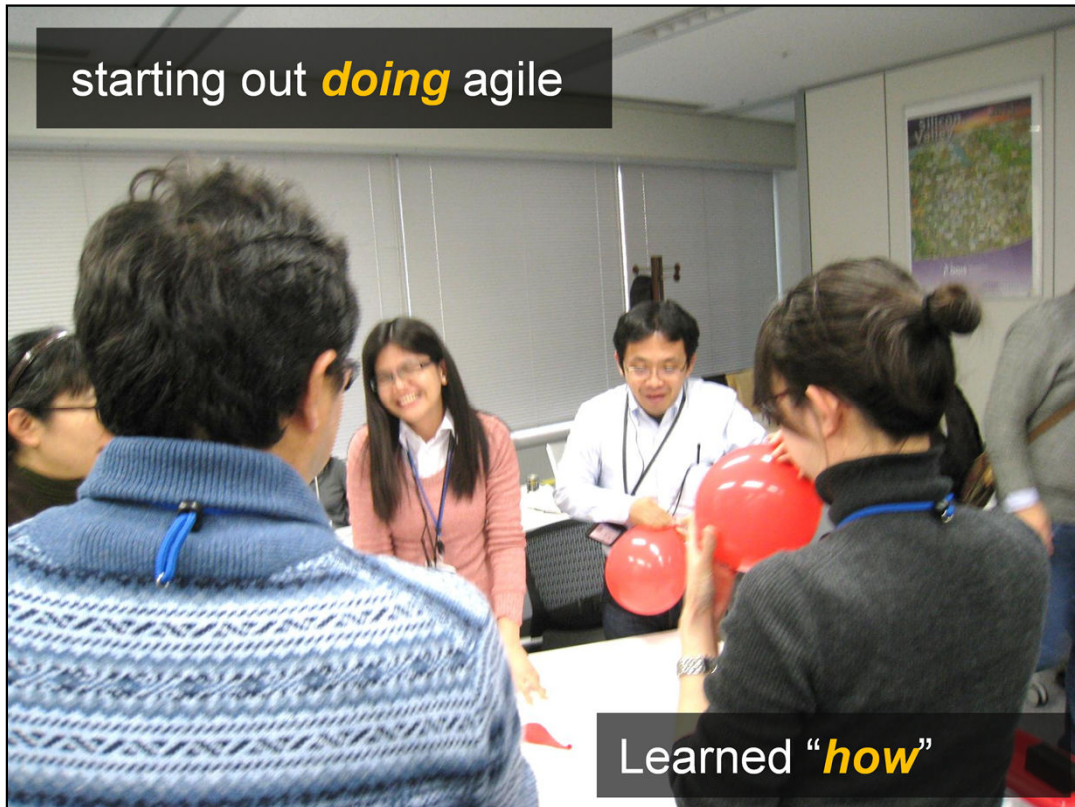
Agility in



Our company wide agile initiative started after a new company president was appointed in year 2009.

The aim was to make the company more competitive.

We are trying to be agile not only in software development but also in support, operation, marketing, sales, personnel as well.



Just in year 2011, we had 98 internal training workshops to teach our employees about agile.

Many employees also took CSM and CSPO courses as well. The photo is of a CSM course that was held at our office. It's a balloon challenge.

Awareness of and understanding of agile and Scrum has risen but everybody was still not using it in their daily work too much.

Agile games are fun but how do one apply them in our actual daily work?

We've learned "**how**" but not **why**

then **becoming** agile



Needed to learned “**why**”

In this presentation, I'll try to discuss **why's** with the **how's**.

There are 2 major obstacles when trying to spread agile usage in a workplace.

First, there's risk involved in trying something new.

Second, people don't want to be the first one to actually start doing something new.

We now needed motivation to have everybody use what they have learned in agile courses.



Books, lectures, workshops were necessary but they're not enough to transforming a company to becoming agile.

So, we moved to a new office – an office without rooms and fixed desks. We can select where to sit depending on what we are doing at the moment.

For example, when I'm working on a creating a new service, I can sit with people in marketing and sales. Later in the day when I'm working on an on-going project, I can change desk and sit with people from development and support. There's no seat reservations. First come, first serve.

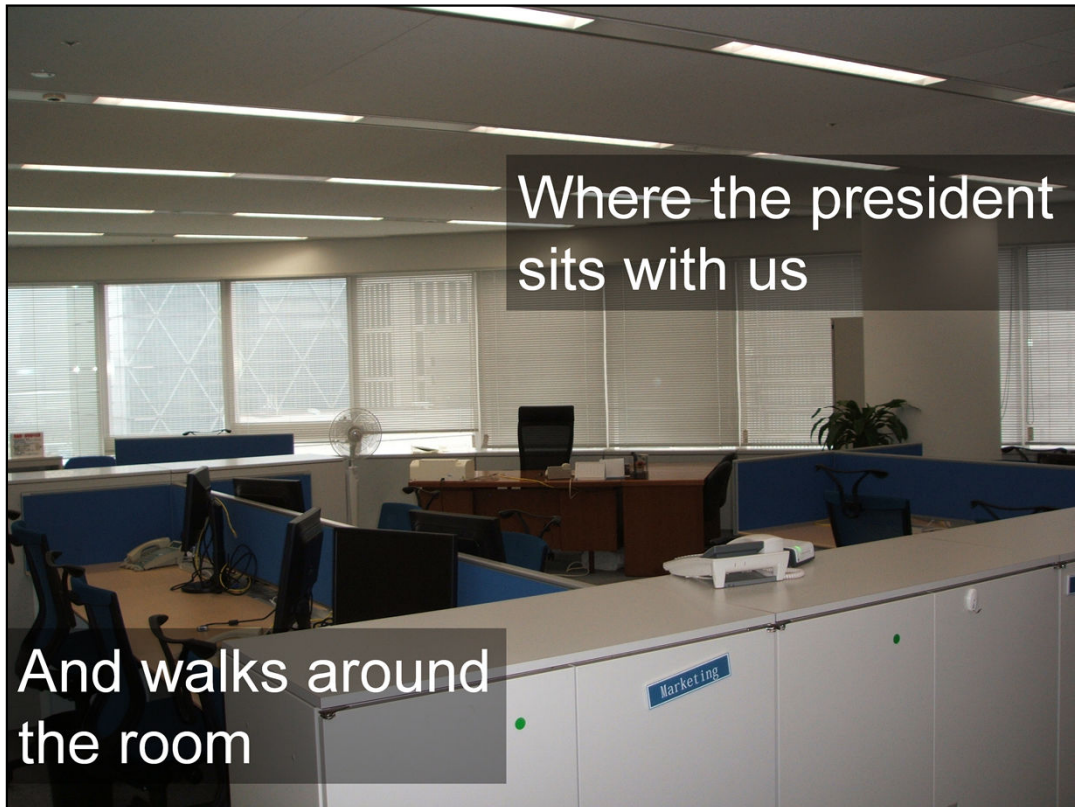
No walls, so I can hear what other people are talking and other people can hear me. If I hear something interesting going on, I can just pull up a chair and sit.

It's something like breakfast and lunch table setting here at Agile2013. People who come early can pick their own desk to sit while the people who come later are kind of force to sit with somebody else – often with somebody who they have not talked with before. Greetings and information can be exchanged.

There's also no hoarding of documents because there's no personal cabinets.

In our old workplace, some of our employees who are at our customer's site used to have a desk in the office. With free space, people don't have an assigned desk so it became possible to reduce number of desks. Most of us are now sharing our desks and interacting more because we now have to share a desk with other people.





This is where our company president sits – in the same room as everybody else.

Instead of just relying on reports, he can just look across the room to see what everybody is doing. If there's something that concerns him he can just walk across the room to ask a question.

Employees, also, are now more able to talk with the president because he is around and there is no wall nor desk separating us.

Beside these, we also changed our corporate hierarchical structure and salary structure. Internal SNS system was also setup so employees are now able to share things outside of work.

Where we help each other to do better



To further promote using agile in workplace, a **Kaizen** team initiative was started. It's on a voluntary bases and a leader (not necessary a manager) may start it up for the team.

Currently, this initiative is more about agility in the workspace instead of on software development. As the first step, we're trying to strengthen the team spirit and to encourage more teams into participation.

Most of the problems are those they've known to have existed, but nobody have put in an effort to correct them. As an example, some employee came to his/her desk in the morning, sat down, did work, and just went home when the bell chimed. If you're doing this, a company isn't very fun place to be.

There's 3 points to the kaizen tea initiative:

First, try to visual as much as possible. We're just sticking postIt and taping paper to a board. Visualization allows everybody would be a able to **know, share, join**. Other people would be able to pass by and will be add comments by writing them on a postIt and sticking them on. Visualization also allows peer recognition which motivates the person who's doing it. The photo is those of a KPT (Keep, Problem, Try). We're using KPT often so they'll come up again.

Second point is that there shouldn't be negative remarks.

Third is that the initiative should be self-governed without manager interface on what and how it should be done. There's more of inter-coaching between teams rather than having somebody "teach"



Employees have been more motivated because they've found that THEY can do something about the problems they knew existed.



Agile is about making everybody **happy** so projects will **succeed**.

It is **not** about finishing a project to “win” but on **succeeding** so more individuals will be happy.

So, we were accustomed to thinking that success will bring happiness - succeed in a project and get a reward that will make us happy. As most of us know, this doesn't work out too well. Giving a raise may promote competition, but it also creates “winners” and “losers”. These tend to bring stress and isolation as well.

It may be that it's not all about finishing a project, but on helping individuals create a “success” – “happiness” cycle. Letting individuals “succeed” so they will become more “happy” which will then promote more success.




## Can being agile overcome social-cultural differences?

Can this concept work across cultures?

Japan and China societies seem to share many common values – They have (1) hierarchical social structure, and (2) emphasis on social network. They, however, have subtle differences.

Differences may be the result of Japan being an island with agricultural society where most citizens not accustomed to other culture, while China is part of a **continent** and with society more **accustomed** to diversified cultural exchange.

Before we go any further, I want to state that “social network” here implies “tatemae” (建前) rather than “honne”(本音). Tatemae is more about how a person communicates based on one’s social status rather than his/her real feelings.



## Our experience of trying to improve off-shore development.

- Experience over 10 years
- Packaged application development
- Use development team in China

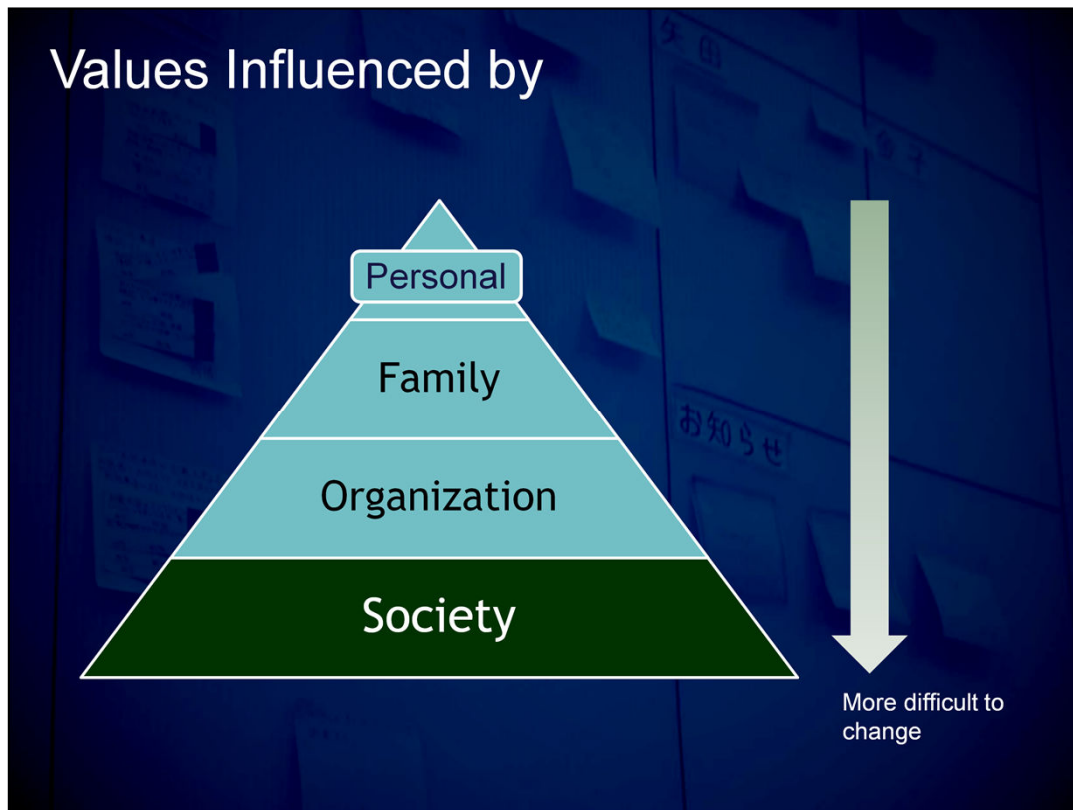
In **2005**, we decided to off-shore to cut development **cost** and because there was a **lack** of developers.

Difference in language, cultural differences (what is considered as norm), distance were some of the problem we experienced.

We started with waterfall because we knew that the best.

Specification written in Japan,  
sent to China to be coded,  
test specification is written in Japan,  
test conducted in China,  
review results in Japan

This steps would have worked when outsourcing development to another Japanese company, but it didn't when we outsourced to China.



So, what was the problem? Socio-cultural issues.

Values embedded in society is more difficult to change than personal values from surrounding pressure.

To changing personal habit only is about changing one individual but to change family, organization, and society, there are many more people involved.

It's just not the number, but their influence and tendency to remain status quo.

Family has tradition

Organization has rules

Society have customs, laws, and constitution

People often seems to just follow them. They seems to forget that they are the ones who are making these and deciding to stick with them.

It's much harder to move out of a society than to move out your home.

Your parents may say,

Hitoshi, you're old enough. Get your own apartment.



I haven't heard anyone say,

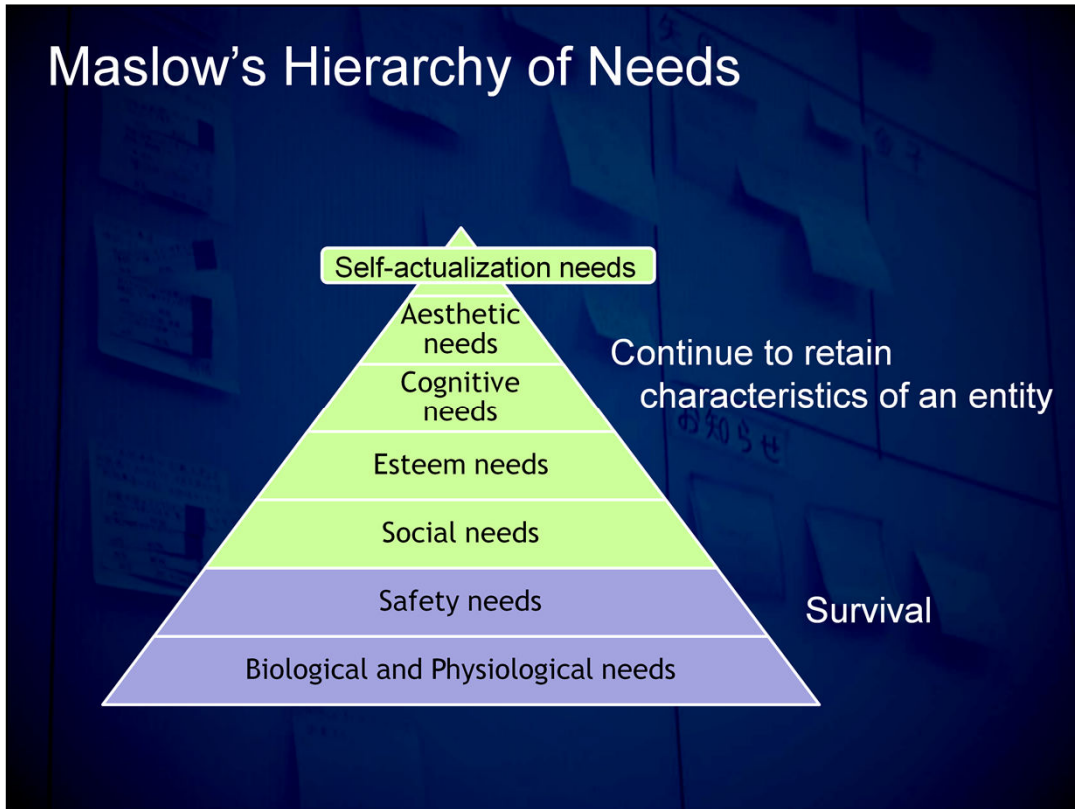
Hitoshi, you've been with the society long enough. Move out and become a hermit.

As a social animal, human being needs to "belong" to a community. That is, ties between people are desired.

If it was just one person that's different, that one person can be persuaded to "adjust" with everybody else.

It's more difficult when there groups of people from two societies meet because they often want other group to adjust to their norms.

# Maslow's Hierarchy of Needs



But, people and their surrounding are always changing.

People came to be because they were capable of adapting to the changing environment.

But as a human being,

We need to change not only for survival but to continue to retain our characteristics of an entity.

Do we want to retain who we are? What makes us unique? Or just want to retain our name?



Before we try to change, we should decide on what we do **not** want to change.

An entity has elements that makes it unique – we don't want to change these.

Because the surrounding environment is always changing. If the entity resists changing totally, it risk losing what it values or its entity.

An entity needs to change to protect what it does not want to change because the environment (surrounding) is changing.

Therefore, before doing a change initiative, it is necessary to know what we do not want to change so we will be able to know what needs to be changed.

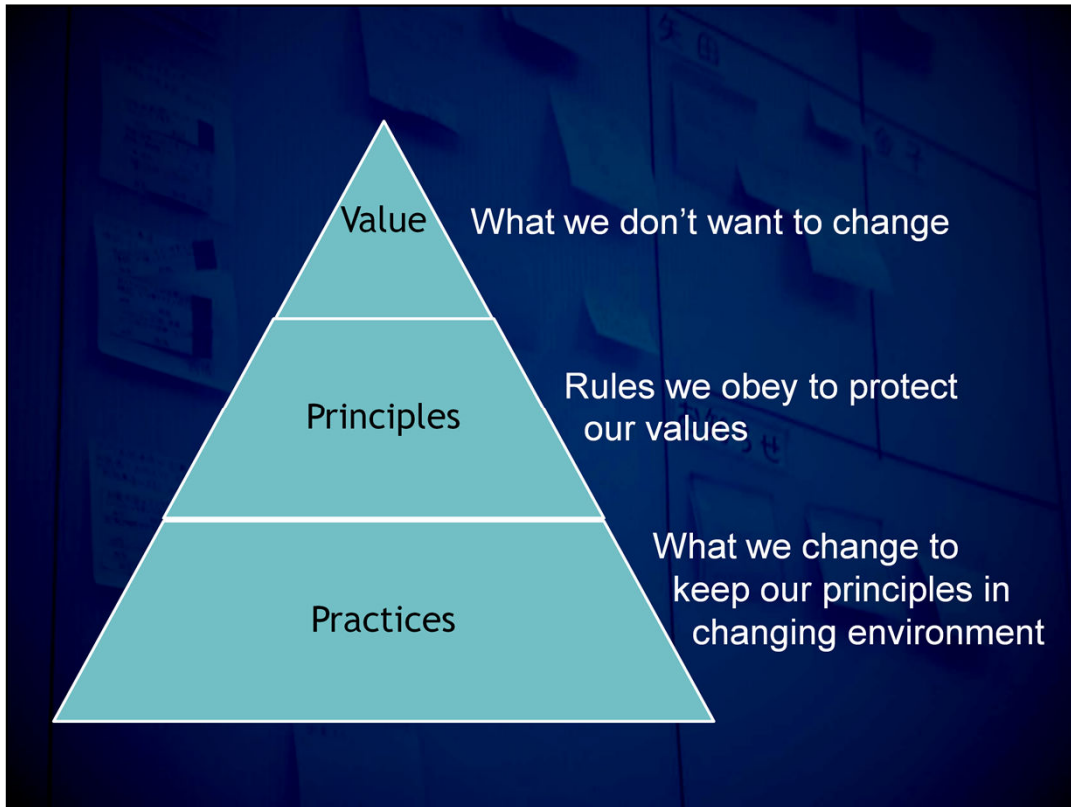
It's like getting married

you need to adjust to get along

if you want to live like when you were single, you'll probably headed toward a divorce

Things probably changes after you have children,  
when they begin going to school  
and after they leave your home to be on their own.

We need to change to protect our love for them.



Values are those things which we do not want to change

Principles are rule of code or conduct to protect our values

Practices are what we change to keep our principles in changing environment.

Dictionary.com, Webster

Values: what we think has worth, merit, or importance. Relative worth, utility, or importance.

Principles: A rule or code of conduct

Practices: What is carried out. To do or perform.

To become agile, one needs to understand what we really value and want to protect from changing, and to change our principles and practices in response to the changing environment

We want to protect at OGIS-RI

誠

Sincerity

技

Skill

This is our principles at OGIS-RI.

We decided that we wanted to protect “makoto” and “gi”. Makoto means sincerity. “Gi” means skill – high standard of skill.

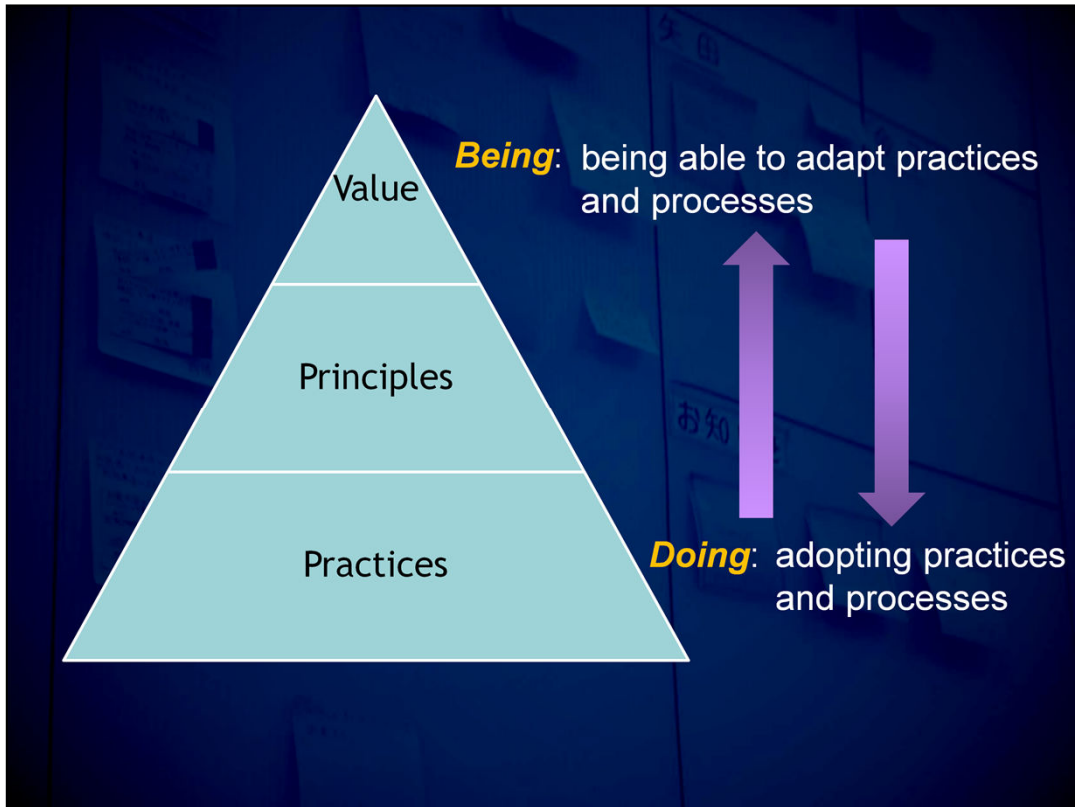
Our customers needs changes over time and technology is ever changing rapidly.

To provide sincerely our customers with the highest skills we can offer, we needs to change our practices.

It should be mentioned that the company doesn't provide these. The company just provides an environment where their employees can and is motivated to sincerely provide high skills. As an example, to do this, an employee may decide to read books and attend seminars to improve skills. A company may buy books and sent their employees to seminars, but without knowing the why's, employees may just be “doing” without attaining the objects.

Social practices are rooted in society's values but we sometimes just continue doing the practices and forget the why's even when the surrounding conditions changes. Thus, doing the practices are no longer

protecting the actual values.



When we first started learning agile, we started by following “recommended” agile practices – assigning agile roles and agile practices. - we were “doing” agile. This helped us to get started in understanding agile values and principles.

As we came to understand and feel more comfortable with Agile, it became possible to merge our values and principles with those of agile – that is, adapting agile.

Difficulty with working with different societies with different cultures is caused by differences in value, principles, and practices. However, we have found that it is not really necessary to obtain a single common value and principles between two cultures if we develop understanding of each other’s values; with common understanding of values, it is possible to discuss and come up with common practices even when values behind them are different.



## Hofstede's Social-cultural Dimensions

Dimensions	Japan	China
Individualism - Collectivism	46	20
Power Distance	54	80
<b>Uncertainty Avoidance</b>	<b>92</b>	<b>30</b>
Masculinity - Femininity	95	66
Long-term Orientation	80	118

Geert Hofstede identifies social culture dimensions that can be used to measure differences in how society prioritizes some basic social concepts.

China and Japan seems to be socially similar, but China and US have lower uncertainty avoidance than Japan - the level of tolerance for uncertainty and ambiguity within the society. Japan has a very high UAI value of 92 while China has a very low value of 30.

A culture like Japan with high uncertainty avoidance presents little tolerance for ambiguity and prefers detailed planning compared to a culture like China with lower uncertainty avoidance index.

In view of people involved in software development, Chinese developers are more like US developers - they are more willing to adopt new practices such as agile practices, but they are also more likely to take a "risk" by changing jobs. Japanese developers, on the other hand, emphasized more on reducing "risk" on improving software quality and using "proven" practices.

This does not mean Japanese people are less interested in new technology but that they are less prone to adopt it in their on going project. For example, books on Scrum was ranked 2<sup>nd</sup> and 3<sup>rd</sup> in computer book ranking even though adoption of agile is less than 2.4%. Compared to

agile adoption to 35%, this is very low.

Japanese love new things. Coca Cola Japan to remain competitive in the market, they ship new product every 1 to 3 weeks. Compared to over a month in US, I think you'll see that Japanese do like new things.

Buying and drinking new drinks poses very little risk compared to adopting a new software methodology.

In offshoring software project, this subtle difference in view of risk seems to be cause of the conflict between Japanese members and Chinese members.

Japanese members emphasized more on reducing "risk" and on improving software quality using "proven" practices. They were also more tolerant of conflicts because they are less likely to change jobs.

Chinese developers, on the other hand, were more interested in using new technology, and they were also more likely to take a "risk" by changing jobs when a conflict occurred.

## Risk aversion causes:

Difference in **openness** of society

Difference in willingness to **adopt** new techniques and technology

Difference in **communication**

During our offshoring experience with a company in China, we detected three Sociocultural challenges related to risk avoidance:

- (1) openness of society,
- (2) difference in willingness to adopt new techniques and technology
- (3) problem encountered with implicit communication over explicit communication.

### Openness of society

=====

Japanese society is more closed and close knit. It's often said this is because Japanese are agricultural society. People do not move much and still hold community gatherings as well as neighborhood activities like daily trash cleanup, neighborhood circular notices. A common phrase “一所懸命” means to protect land provided by ancestor with your life (Japanese medieval era).

Japanese employees, also, think of employment as a lifetime commitment to a company. Japanese companies prefer to recruit fresh graduates from school and “teach” them to fit with their group instead of hiring experienced people who likely have their own ways of doing things. Japanese society has a concept of “senpai” (senior) and “kohei” (junior). Senior is assigned to look after the junior member especially in OJT (on the job training). OJT is the preferred method to teach junior members and

to welcome them into the team. Classrooms are used to teach general information while OJT is used to team new members to the “way” of the team. This sometimes includes teaching them on task not directly related to work such as on how to fold report papers and how to staple them.

Generalized. On the other hand, if a junior member is sick, a senior member will phone or even go to junior’s member’s apartment. They may even go shopping if a member is too ill to go out.

In such an environment, “和” or harmony is preferred - gaining consensus to reduce internal conflict is valued than gaining higher skills to obtain better position

On the other hand, Chinese members tended to switch job after acquiring necessary skills for a new position.

These differences created confusion and conflict between Japanese and Chinese members.

#### Difference in willingness to adopt new technique and technology

=====

Japanese society and organization places more emphasis on avoiding failure instead of attaining success. Therefore, they are more likely to use existing methodology and technology and only adopt minor changes. Chinese members were more interested in acquiring new skills by trying new techniques and technology. Difference in organization cultural rate in adopting new techniques and technology caused friction between the organizations.

#### Differences in Communication

=====

Differences in communication is cause by (1) difference in meaning of common word and gestures and (2) difference in implicit knowledge. Miscommunication and misunderstanding are caused mainly by differences in interpretation caused by differences in social context rather than from syntactical differences.

China and Japan both uses “Kanji” written characters however meaning of some characters differ slightly. As an example, a word “以上” means greater than or equal to in Japanese while it simply means greater than in Chinese. There are, also, misunderstanding of contextual meaning of verbal and nonverbal communication even in face to face meetings and misunderstanding of written specification and messages. These differences, however, can be avoided by writing up a “translation list” between Japanese and Chinese and having somebody proof read the documents.

Implicit difference, however, poses a more difficult problem because they are not written nor explicitly said. Implicit knowledge is socially embedded within social norms and not easily shared out of context.

Japanese tend to have more "implicit" unwritten meanings in sentences because Japanese members usually form a very tightly closed group where members understand each other without explicitly saying everything.

In a closed hierarchical society like Japan, it is seen to be better if a person "understands" and serve one's superior before it is actually said. Another cause of implicit communication may be that in Northern Japan, Japanese used to live in wooden windy houses that gets very cold in winter. People seems to mumble and to shorten words and sentences so they won't have to move their mouth too much.

A typical conversation between a Japanese man and his wife is as follows:

Husband: (sitting at a table and reading a newspaper. Talking without taking eyes off the newspaper)

Will you get me that? (not pointing)

Wife: Yes dear, here are your glasses.

Husband: (putting on his eye glasses)

Get me that too. (not pointing)

Wife: Here is your tea.

Husband: (sipping his tea and hiding behind the newspaper he is reading)

And ...

Wife: Not tonight dear. I have a headache.

Try to image this conversation between an American man and his wife. Well, the last part maybe the same.

In a Japanese company, a senior would say a junior is "kawaii" to signify that they share common knowledge and have tight bound between them. Unfortunately, Chinese members were "outsiders".

Japanese Product Owner were required to write detailed complete specification in consideration of these differences. Specification was reviewed and corrected to avoid miscommunication, misunderstanding, and any ambiguities. This required considerable time and cost and Japanese Product Owner became wary from writing these detailed specifications.

## What we learned from off-shoring projects using waterfall methodology

Problem with **ambiguous** specifications

Different stances on **quality** requirement

Lack of information **sharing**

Low **motivation** in development team

Initial offshore development was done using a waterfall methodology because the Japanese members had experience with it and felt more comfortable. Most common anticipated problems encountered in offshore development such as problems arising from geographical dispersion, different time zones, and common problems associated with difference in languages were resolved during this stage.

We were able to cut initial development cost by 10% using waterfall methodology in off-shoring software development.

Did we stop? No!

Some problems remained.

Quality decreased and users were not really satisfied with the quality of the software. These defeats were often found during the acceptance test after the software were coded. Japanese member just wanted it fixed, but Chinese members were also reluctant to fix the defeats because the Chinese society associates “concession” with willingness to compensate. Win – lose mentality.

Resubmitting a fixed specification and redeveloping the software until

quality were satisfactory required time and expense such that it resulted in no cost benefit of off-shoring a project.

Upon analysis, following 4 reasons were cited as the cause of the problem:

(1) Specification was ambiguous,

Cause by members of project not understanding each other and specification.

Japanese members usually work in a tightly closed group and understand each other's requirements without having to express them explicitly. It is only necessary to explicitly write differences from their implicit implied norms. However, it is necessary to write a detailed specification to get the software they wanted when asking Chinese members to develop a software system. This added time and cost of creating documents that were unnecessary when developing in Japan.

(2) There was a cultural difference in quality acceptance level.

Japanese customers request very high quality from the beginning. Competition is very high in Japan because it's a closed market. It was a "norm" to have a very high quality software from the beginning without explicitly stating as such. Chinese developers and testers, on the other hand, were conducting tests on normal conditions only without any test on error nor abnormal conditions. They only added these tests when the Product Owner found the software was not behaving as expected even after it passed testing..

(3) Information sharing was insufficient

Both Chinese and Japanese educational systems have a teacher talking and students just listening. It is seen as a "humiliation" to say they did not fully understand what the teacher is talking about. Chinese Development Team members were reporting they were not having any problem to avoid "humiliation" of saying they didn't fully understand the specification. Instead, they would try to "resolve" the problem based on their cultural norms, knowledge, and experience.

## Conflicts within the team

- Lower involvement
- Less cohesiveness
- Decreased motivation



## Effect on the project

- Low quality deliverables
- High turnover rate

(4) These resulted in conflicts and caused decrease in members' **motivation, involvement, and cohesiveness**. Both Japanese and Chinese members were unsatisfactory with how the project was progressing and Chinese employee turnover rate was high. In 2010, motivation of Chinese members deteriorated to old time low when a key member left the company.



People are **social** by nature

Waterfall model, however, tends to isolate members by assigning them tasks to do.

(Factory workers are not too social while doing tasks in their lines.)

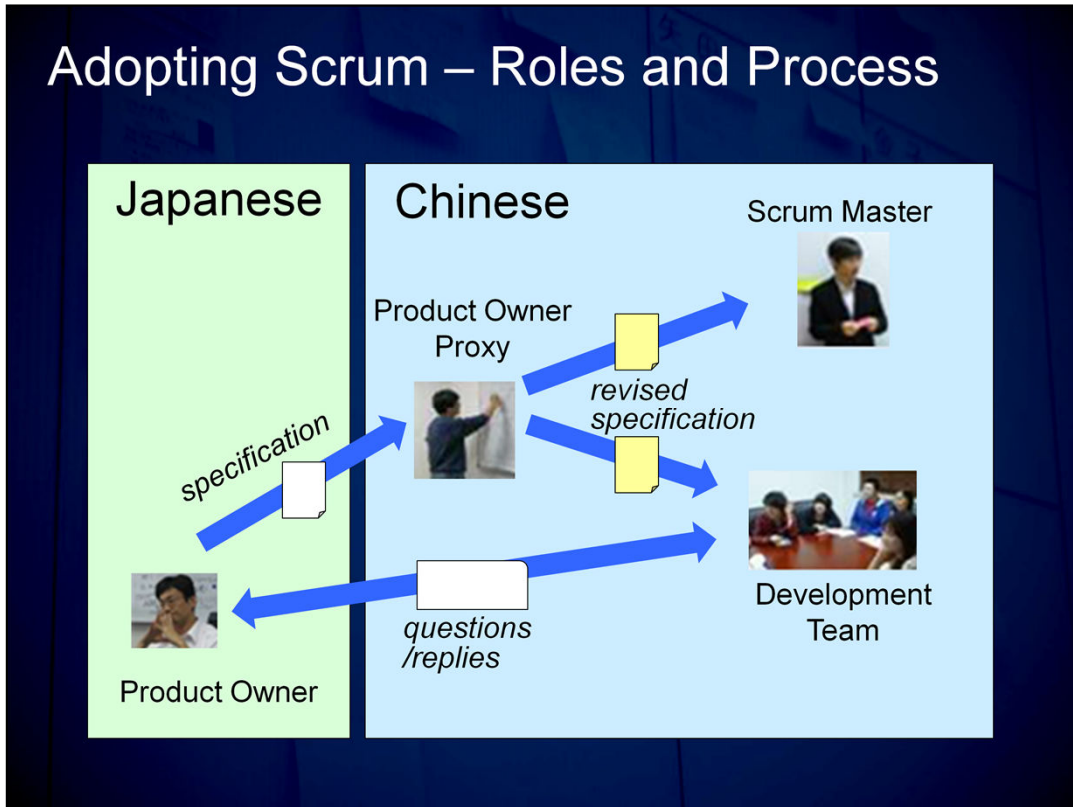


This is going against human nature to be social.

In the initial effort to alleviate these problems, Japanese Product Owner was asked to write much detailed specification that included all implicit understandings. The finished specification was further reviewed to weed out miscommunication, misunderstanding, and any ambiguities. This required considerable time and cost and the Japanese Product Owner became wary of writing further detailed specifications. He complained about having to write these detailed specifications but diligently completed writing them. On the other hand, the turnover rate of Chinese developers continued to be very high.

There had to be a solution. May be, it was the waterfall methodology we were using – it was based on writing a perfect specification that would then be passed on to developers. It was isolating members. May be, we should be focusing more on member interaction to resolve problems instead of trying to write a “perfect” documents.

# Adopting Scrum – Roles and Process



Both Japanese side and Chinese side agreed to use Scrum in a hope a new methodology and practice will improve the situation but with different reasons. Japanese Product Owner is held accountable for the finished deliverables and he wanted to reduce risk by having shorter development cycles and more transparent status reports. Chinese members, on the other hand, wanted more authority on development so they would be more able to adopt new technologies.

After Scrum was adopted, Chinese Team decided to use XP practices during each sprint.

## Roles

=====

All our customers were Japanese, so a role of product owner was assigned to a Japanese member residing in Japan. Roles of scrum master and the development team were assigned to the Chinese members in China. The Chinese Scrum Master was chosen because a Scrum Master needs to communicate frequently with the Development Team to find and remove any obstacles a development team member may have. It was better to have a person who understand them socially. Furthermore, to reduce misunderstanding and miscommunication between Japanese Product owner and Chinese scrum master a role of Product Owner **Proxy**

was created to bridge implicit communication gap between Japanese Product owner and Scrum Master and development team. Product Owner **Proxy** was assigned to a Chinese member with deep understanding of both cultures and languages. It is the responsibility of a Product Owner Proxy to act as a liaison between the Product Owner on-site with the Development Team so all members would have the same project expectation of the deliverables and to arbitrate conflicts when they do happen.

## Process

=====

Agile development process initially followed those used during waterfall development. Japanese product owner gathered requirements from customers in Japan with which user stories were written. User stories were sent to the Chinese Product Owner Proxy who reviewed the user stories and rephrased sentences that may cause misunderstanding or miscommunication.

A Japanese Product Owner who wrote the user stories went to the Chinese office for 3 days to explain the specification to the Chinese Product Owner Proxy, Scrum Master, and to the Development Teams. Chinese Development Team wrote backlogs from these user stories that were again reviewed by the Product Owner Proxy to make sure there were no misunderstandings or miscommunication. Product Owner, Product Owner Proxy, and the Team worked to convert user stories to the Product Backlog and then to Sprint Backlog. Software was developed by the development team based on this Sprint Backlog with a sprint cycle of 2 weeks. Product Owner is responsible for clarifying the user stories during the planning and during the development stages. If the Development Team needs any clarification of the Product Backlog during development, they contacted the Product Owner. Questions were initially thought to be answered by the Product Owner Proxy collocated with the Development Team, but it was decided to send to the Production Owner directly instead of to the Product Owner Proxy to alleviate responsibility issues when a defect was detected.

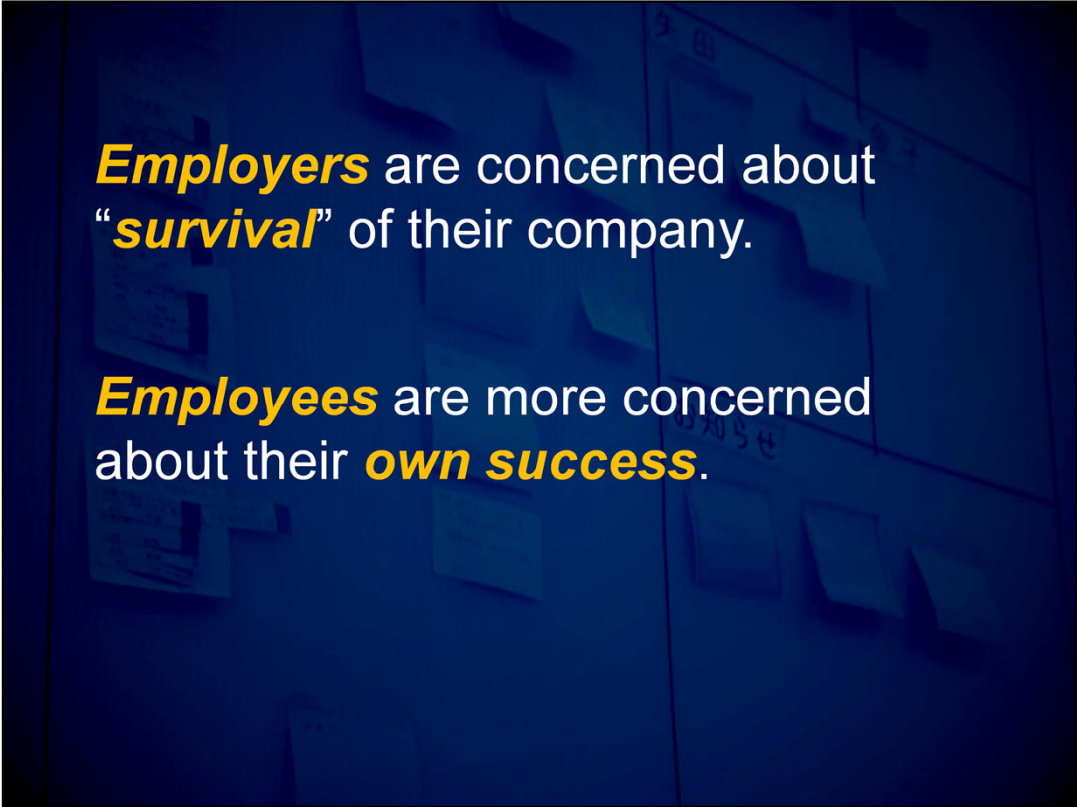
Developed software was tested by the Chinese testing team. When it passed all tests, the software was delivered to Japanese Product Owner who conducted acceptance tests. A Japanese Product Owner sometimes went to China to join the testing team to see the quality before conducting an acceptance test on his own.

Software quality improved from increase interaction. More bugs were detected and fixed before the final acceptance test.

Was this the best we can do? Did we stop kaizen? No!!

Quality of the software has improved but it was still not at the level as when

software was developed in Japan. We wanted to improve the quality more.



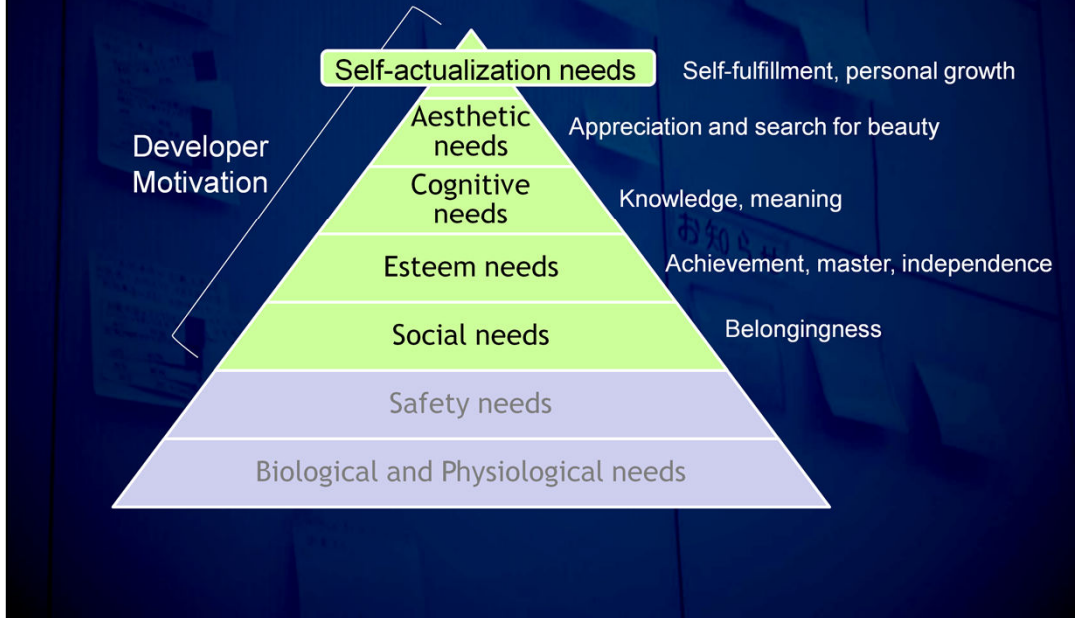
**Employers** are concerned about “**survival**” of their company.

**Employees** are more concerned about their **own success**.

Software development isn't a simple mechanical task and most developers are also making enough to live by. That is, most aren't concerned about survival – they can move to a different company, some can even retire. To them, agile is not about survival. Simply telling them they need better performance so that the company would survive from tough competition is not enough. Employees can just leave and work for a different company when a business fails. In a large company, the common idea among employees are that the company they are working for won't go bankrupt before their retirement.

How it is possible to motivate these developers? Our Chinese members already were given better wages compared to similar position at other companies. Why weren't they motivated more?

# Maslow's Hierarchy of Needs



Looking at Maslow's hierarchy of needs, it may be they wanted more self-actualization needs rather than safety needs.

## Difference in “**success**”

1. **Personal** success
2. **Organization** success
3. **Technical** success

Chinese member:

Technical



Personal

Japanese member:

Organizational



Personal

We all try to obtain personal success but Chinese member wanted to gain technical success so they would be able to a better company while Japanese wanted to obtain organizational success because they are less prone to move to another company and wanted to succeed in the company there were in.

## We all want to develop our career paths



Japanese by succeeding in a project within an **organization**.

Chinese by acquiring new **technical** skills to help them find better jobs.



Chinese were more concerned about using new techniques and technology they can “sell” to get better jobs rather than making the project succeed. We’ve found they sometimes intentionally neglected time consuming daily tasks that didn’t increase their technical skills. Japanese, on the other hand, was more concerned about succeeding the project rather than using new techniques and technology. They wanted to use “safer” more “worn” techniques and technology.





After each development cycle, agile practices were re-evaluated during agile retrospective. KPT (Keep, Problem, Try) sessions were held. Instead of just looking back at events, the current situation between project members were analyzed and detected and predictable problems were written into the "Issue analysis table".

## Issue Analysis Table

	Difference anticipated		(3)Difference unanticipated
	(1)Known outcome	(2)Unknown outcome	
(a)Can change	Difference in meaning of words	Implicit meanings	
(b)Cannot change	National holidays	Distributed locations	High turnover rate
(c)Does not affect	Time zone difference		

Issues in a project can be classified into 3 types

- (1) anticipated issue with knowable outcome
- (2) anticipated issue with an unknown outcome
- (3) unanticipated issue.

Furthermore, the outcome can further be divided into

- (a) outcome that can be resolved,
- (b) outcome that cannot be easily resolved within the current situation,
- (c) problems that do not affect project outcome.

As an example, there are some characters such as which are common to both Chinese and Japanese but have different meanings. Characters such as these are known in advance and are known to affect developed software.

However, they can be circumvented during the translation process.

On the other hand, it is known there are some specifications that are implicitly defined and differ between the two cultures, but cannot be determined beforehand. In our case, error handling was expected to be implemented by the Japanese members and was not explicitly written in

the documents. Chinese member, however, only implemented what was written in the specification and did not implement error handling.

An example of unanticipated difference was the high turnover from lack of motivation of the Chinese members. Lifetime employment is customary in Japanese society and turnover by project members were not expected.

Hofstede's cultural dimensions are a good model to start when analyzing culture but in an actual project, member interaction does influence each other and the cultural dimensional values change over time. To take advantage of these changes, situations were re-analyzed and roles and processes were adapted to resolvable problems after each cycle. Factors influencing problems classified as that cannot be changed were investigated to see if they still cannot be changed or if they can now be changed if practices and conditions were changed.

Those that can be changed were planned for implementation in the next cycle. Those that cannot be were further analyzed to find factors and possible solutions were proposed. If all members agree to the solution, it was implemented.

If there was a disagreement, another solution was thought or it was marked as irresolvable at a current time.

# Manifesto for Agile Software Development

**Individuals and interactions** over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Went back to Agile Manifesto find a clue to resolve our problem.

One that stuck was the following:

“Individuals and interaction over processes and tools”

Within the Scrum framework, we also considered followings as well:

1. Allow members (Pigs) to do what they want instead of trying to trick them to do what you want

2. Try to use minimum technology possible – keep it simple

It's about interaction of individuals not about interaction of an individual to a computer. Technology used wrong cause individuals to distance themselves instead of bring them together.

As an example, social network like Facebook can brings people together, but recent problems of teenage suicide cases is an example where it is used wrong.

3. Decide on constraints. **Complete freedom** is not sustainable. Be willing to **negotiate** to decide on the constraint.



**“People”** are assets that may be replaced

**“Individuals”** are assets that can not be replaced

**Motivation** is about individual’s value.

During the early stage of off-shoring our project using waterfall methodology, we were trying to create a “software assembly” line – the goal was to make the line so the people in the assembly lines can easily be replaced.

Statistics on members and projects were measure to improve on development productivity.

But, motivation is not about **people’s** value but about what **each individual** value. When we switched to agile, it was necessary to treat each member as individuals rather than a replaceable asset.

It's not about the “*development team*”.  
It's about...



Agile is not about generalization and finding a “perfect” rules to do things.  
It's about personalization, acknowledging that human beings are imperfect.

## Cause of the Problem!

Was trying to make a “**project**”  
**succeed** instead of allowing  
**members to succeed**.

Wasn't being “**sincere**” to project  
members.

So, we decided to give members opportunities to succeed.

We were trying to “trick” developers so the project would succeed but switched to thinking of ways to help developers attain their success.

It's not about succeeding in a project so a person would get rewarded but about helping members find their happiness so a project would succeed.

It's not about putting customers first, but about putting employees first. Happy employee would treat their customers more nicely than a disgruntle employee.



Stop trying to find what's “*right*”

Stop trying to create a common culture because people in general are less prone to change their customs.

So how do we put this into practice in our off-shoring project?

Don't argue which culture is “better” because they lead to arguments which widens the gap. Some social values, principles, and practices are very difficult to change. Trying to change will often lead into dispute and conflict.





Accept that social differences are going to exist.

Focus on “*succeeding*”. Not winning.

Winning an argument means there is a “loser”. A person may feel better when he/she wins an argument, but a team with “winner” and “loser” usually does not succeed.

Try to find a way so everybody involved to feel that they’ve succeeded – a way for all party to be satisfied.

## Step toward resolution:

People may have different values but all people want to:

- (1) be successful
- (2) get along with other people



Increase communication to find how this is possible.

The problem of development team members fully not understanding the specification persisted in our project. But we knew why AND also knew that members actually wanted to communicate.

When we were using the waterfall methodology and during our initial attempt at Scrum, detailed specification was written so developers will be able to code programs without fully understanding the value that it was suppose to add. Writing detailed specification required enormous time but let to an activity based mentality rather than value based mentality and resulted in Chinese member just mechanically rewriting step by step specification to computer code. Programmers were detached from each other and from the software which they were writing.

Instead of trying to make the document clearer, we rethought about the problem and concluded that the problem was not that the documents were unclear; it was that the members of the project were not given opportunity to communicate with each other to try to understand the specification thoroughly and offer suggestion for improvements.

## How to increase communication

### Make specification *vague*

We were trying to make the specification more precise but ambiguity remained.

Chinese developers were choosing their own interpretation.

Counter to intuition, our next decision was to make the initial user stories very vague so that the Development Team would be induced to contact the product owner to find the actual detailed meaning. Generally known Scrum practices recommends Product Owner to be responsible for expressing user stories clearly. Taking this to mean to clearly express all user stories clearly in a limited period of time in the early stage of sprint planning was actually hindering the Development Team from fully understanding the user story to the level needed because it was leading to the lessened requirement to communicate.

## Making the initial specification **vague** forced Developers to contact the Product Owner

### **ambiguous:**

Open to or having several possible meanings or interpretations

### **vague:**

Not clearly or explicitly stated or expressed

<http://dictionary.reference.com/>

Our attempt to verify developer's understanding of the specification by making him retell the specification in his/her own words only had a limited improvement because it was just rewording of what was written in the specification. Implicit specification that were not written were not included in the retelling of the story. Chinese developers still didn't ask the Japanese Product Owner when they found ambiguities during development.

Making the user stories very vague forced the development team to contact the Product Owner for clarification from the beginning. Developers were required to think and to propose how they wanted to implement a feature and to convey this information to the Product Owner. Members were required to communicate to reach a common understanding.

This initiated much closer interaction between Japanese and Chinese members and broke the ice between the two cultural groups. Communication became much more frequent and defects were detected and fixed earlier. Members were also trying to learn each other's language and increased communication was assisting them in this effort. It wasn't that they weren't communicating because they didn't know each other's language, but that they didn't see strong enough reason to learn each other's language.

Vague specification, however, requires the Product Owner to spend more time to reply to questions from the development team. This increase can be lessened if a Product Owner actually creates a regular detailed specification along with the vague user stories. Detailed specification may be consulted to reply to developers' questions or updated when a better alternative solution is found.

## Results

### Higher individual motivation

#### Changes in the team

- Tighter team
  - Product owner became part of the team member
- Trust
  - No commands, no assignment
- Sharing the common goal
  - Improving the product became the common goal of the team

It used to be them against us.

We, however, were able to obtain success by

1. acknowledging that differences are going to exist
2. increasing communication between members
3. adapting agile practices in consideration of the existence of differences empowerment to individuals

After 4 months, we were able to see change in motivation of the members. They became more cooperative, felt more responsibility in one's own, and more participative during the meeting because their ideas were now being used to develop and improve the product.

Employee retention has been a major problem in offshore development because employees tended to switch job after gaining skills to find another job. After adopting and adapting agile methodology in our project, members began to better understand each other and they became more motivated - Employee turnover rate from member dissatisfaction decreased from **20%** to **0%** and quality of the software became more stable by retention of skilled members.



Japanese members initially consented adopting Scrum to hold more periodic reviews and to better track development progress, but later shifted to give the Chinese group more autonomy because they seem to know more about what motivated them and had a better understanding between members - Chinese members were actually able to contribute much more to making the finished software much better because of the accumulated knowledge.

As the result, turn-over rate became lower and software quality became better.

Members are still not completely proficient in each other's language but members now see this as their own issue that needs to be resolved. On the other hand, we still have not been able to resolve the issue surrounding QCD (quality, cost, delivery) reports - Chinese members are complaining about requiring too much resource to write these reports.



In our project...

"If you specify very little upfront (vague specification), you should expect very high implementation cost"

Cohn 2010, "Succeeding with Agile Software Development Using Scrum"

As expected, making specification vague increase communication.

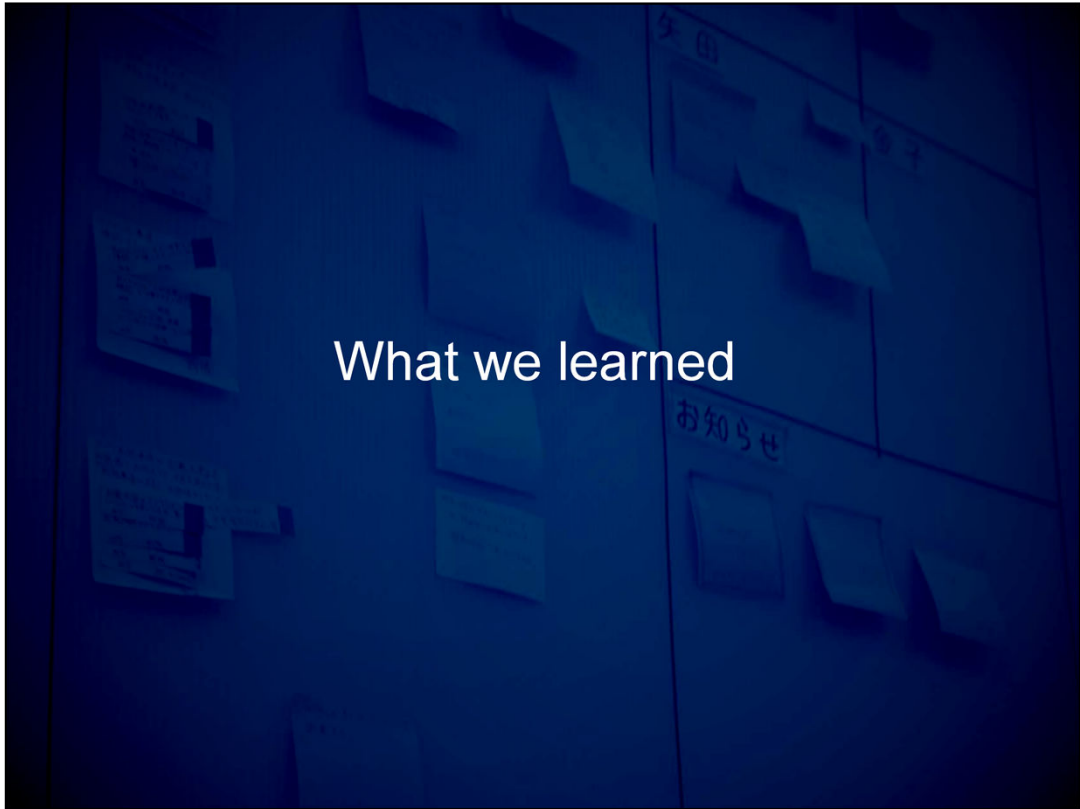
Product Owner was getting more than 100 questions per feature request. This requires time and cost – what we really don't want.

However, once members began communicate, it became possible to make the specification more precise to lower the number of questions. The problem now was adjusting the granularity of the initial specification so the questions would not overwhelm the product owner. Granularity of documents should be adjusted to initially promote communication and later to reduce cost. Interaction of individuals change over time and should be noted. Instead of just following a set of practices and patterns, it is better to think of values and let practices change over time as condition changes.





Happy members produce better results!  
Individual values over practices.



## What we learned

What we've learned from our experience.



## Lesson 1

Continuously adapt based on  
**current** conditions.

Continuously adapt roles and processes based on the current conditions.

If the relationship between members change, modify process and tasks assigned to roles to take advantage of it.

For example, review members and organizations trust after a sprint or during the sprint retrospective and modify roles and process accordingly.

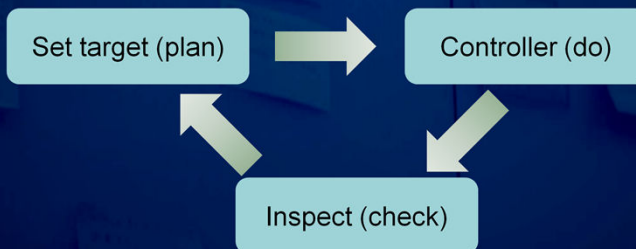
## Traditional (Reduce change)

### Open Loop (Analytical, Predictive)



## Agile (Embrace change)

### Closed loop (empirical, Adaptive)



During waterfall, we were planning and then doing

We adopted Scrum, but we were trying to apply practices rather than the promoting interaction between individuals. Interaction between individuals change and practices should be changed with it.

“Why is happiness so elusive? Our reason is that the definition of ***happiness changes*** every three to five years throughout one's life.”

page xv

"one individual's happiness can affect another's for as much as a ***year.***"

page xviii

"The Dragonfly Effect: Quick, Effective, and Powerful Ways To Use Social Media to Drive Social Change"  
Chip Heath, Dan Ariely, Jennifer Aaker, Andy Smith, Carlye Adler

If you're married, you may already know this.

Feeling of honeymoon may last a year, but unless both parties communicate and change themselves, marriage will go down the hill because it's often not possible to live like when you were single - no more nightly beer bashes.

Once a child is born, things will change once again. Things will change again when the child starts attending school.



"When my information changes, I  
alter my conclusions. What do  
you do, sir?"

John Maynard Keynes

We live adapting to our current conditions, yet we often try to find an absolute practice in software development. Maybe, adaptive patterns (transitional patterns) are what we really need instead of the current practice patterns, which seems to be absolute.

Alexander's patterns are about architectural objects – a timeless ways. Human are not timeless. We live and adapt within time.



This is a sample of a KPT (keep, problem, try) board.

It's just messages written on Post-it - its pasted to a paper. It's very **visual** and people are able to see it and provide input.

Something as simple as this can be used to visualize what we currently want to keep, current problems, and what we want to try.

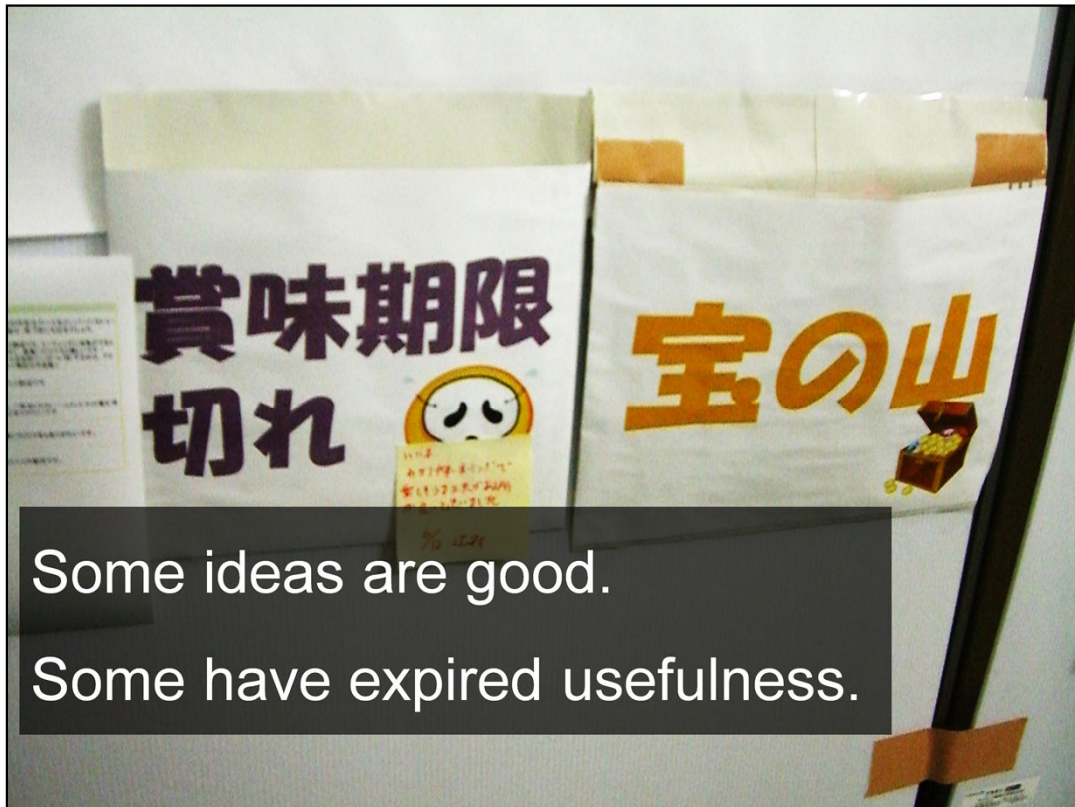


The content doesn't have to be anything complex.

This is a photo of trying to increase communication between employees. It just shows number of people a person communicated on each day of the week.

It's visual and everybody can see it. If a person hasn't talked with anybody else, others will be able to see it quickly and go to that person to talk.





KPT board needs to be updated continuously **by members** because they and their environment are changing.

Good ideas that has been fulfilled or has pasted their usefulness are sorted into the “**passed expiration**” envelope (envelop to the left).

Ideas that should be continued are sorted into the “**treasure box**” envelope (envelope to the right).

The key here is to let the members decide which envelope to put the idea into and sort items periodically.

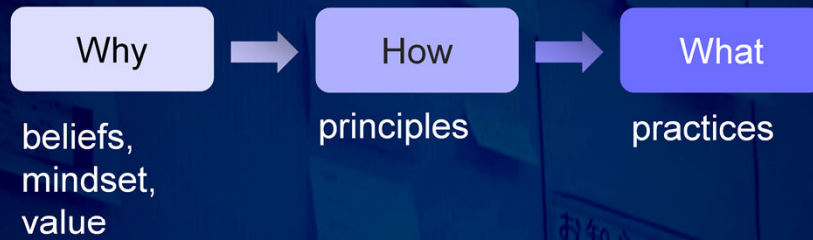


## Lesson 2

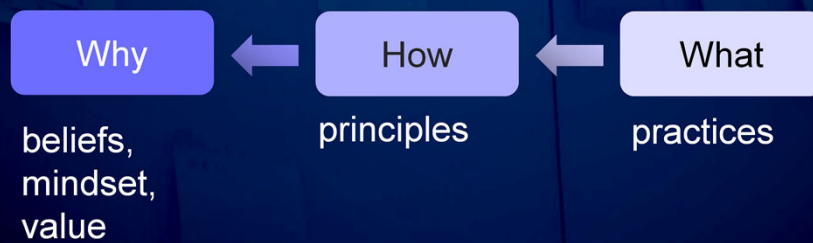
Focus on **why** instead of **what**  
the differences are.

We tend to focus on what the differences are instead of why there is a difference.

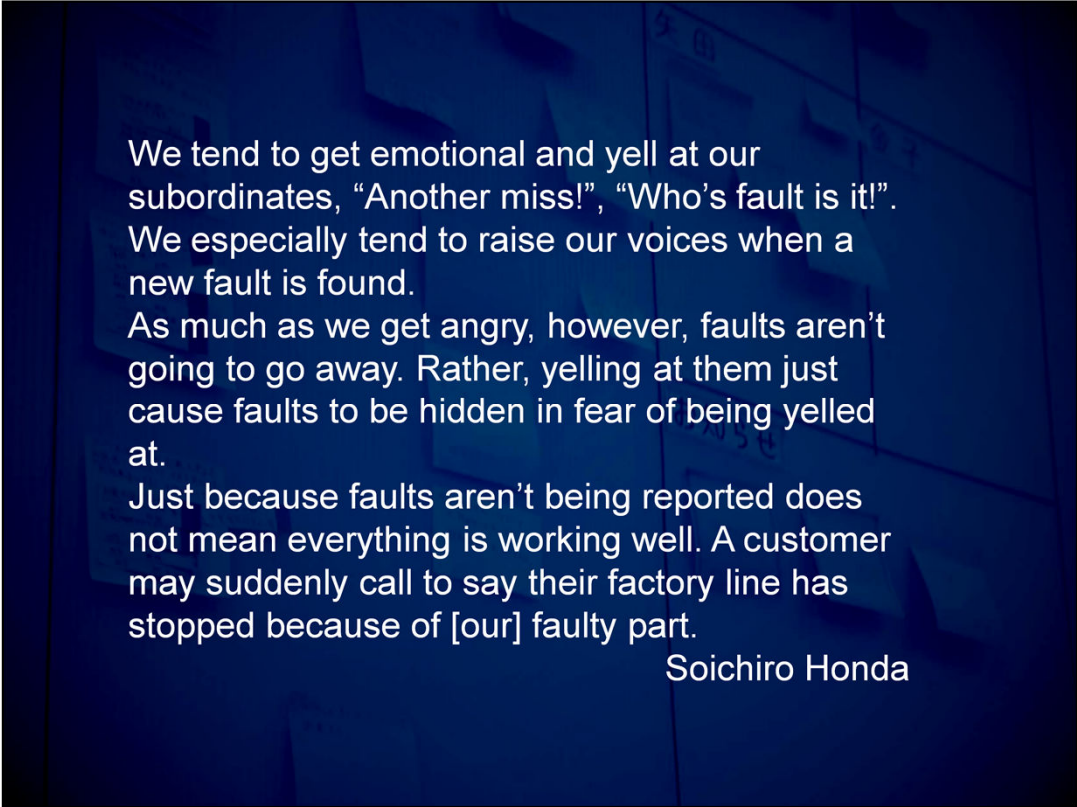
## To motivate, focus on



## instead of



Deciding on which way to do things, however, causes one side to “win” and the other to “lose”. Discussing on “why” enables us to negotiate to reach a conclusion so all may all “succeed”.



We tend to get emotional and yell at our subordinates, “Another miss!”, “Who’s fault is it!”. We especially tend to raise our voices when a new fault is found.


As much as we get angry, however, faults aren’t going to go away. Rather, yelling at them just cause faults to be hidden in fear of being yelled at.

Just because faults aren’t being reported does not mean everything is working well. A customer may suddenly call to say their factory line has stopped because of [our] faulty part.

Soichiro Honda

Focusing on who did what does not resolve the underlying problem. Kaizen is really about the “why’s” instead of “who” and “what’s” because the “why’s” relate to the values we want to protect.

The first step in our Kaizen initiative is to acknowledge and to respect what human beings is with all its faults. Note, that this is different from respecting a person who we tend to eulogize.



"I've learned that people will forget  
what you said, people will forget  
what you did, but people will never  
forget how you made them feel."

Maya Angelou

People often forgets the "what's" but remember the "why's" (how they felt).  
Basing the "why's" on the "what's" may lead to an undesirable situation.  
It's better to think of the "why's" first so you can make it to what you want  
the person to remember.

It's better to concentrate on the "why's" that will motivate the individual.



## Lesson 3

Resolve **few** issues at a time  
over issues **you** have control over

Don't try to force to solve all difference issues. Assess how much the difference affects the result of the project. Some differences cannot be overcome because they are inherited deeply into the culture. Accept those differences and try to find a way to take advantage of it instead of trying to eliminate it.



Make effort on what **you** have control over and are capable of doing

Do the best on what **you** are capable of doing (succeeding) instead of trying to be better than somebody else (winning)

Agile is about what an individual is capable of doing and enabling it to happen. It's not about controlling other people. Don't measure other people but measure oneself and always try to improve.



## Lesson 4

Give time to let issues get resolved.

Interaction between individuals change over time. Even if an issue can not be resolved now, it may be resolvable in the future as relationship changes.

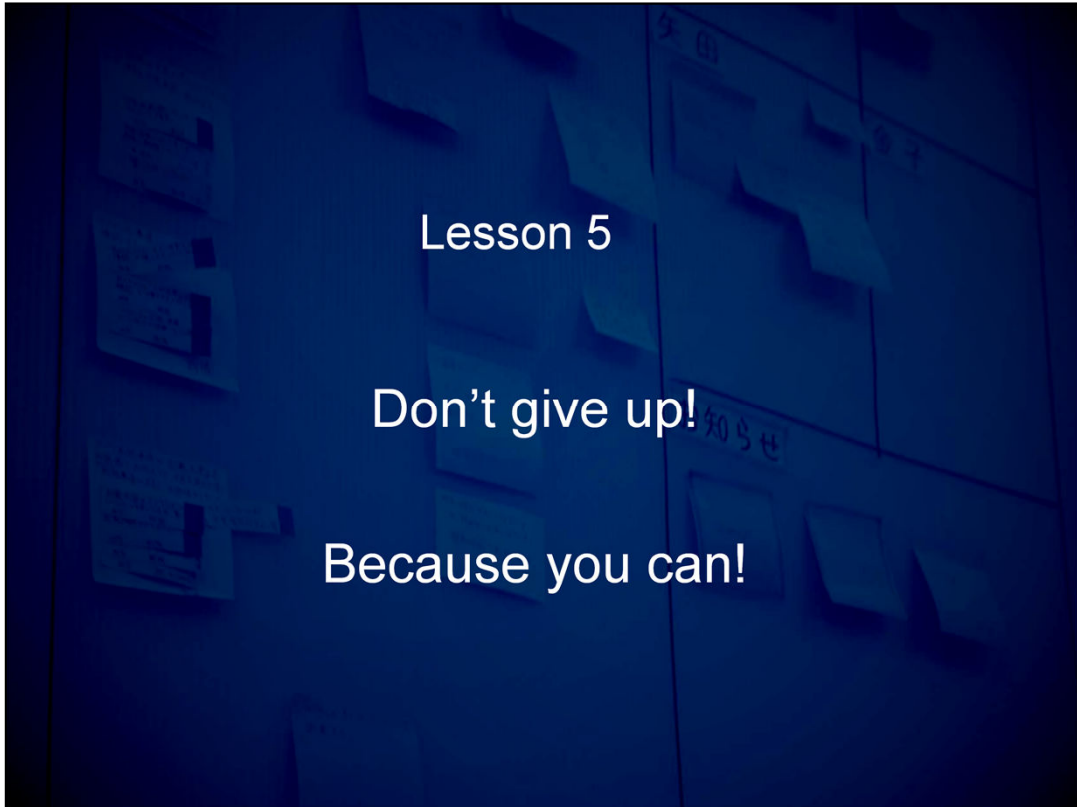




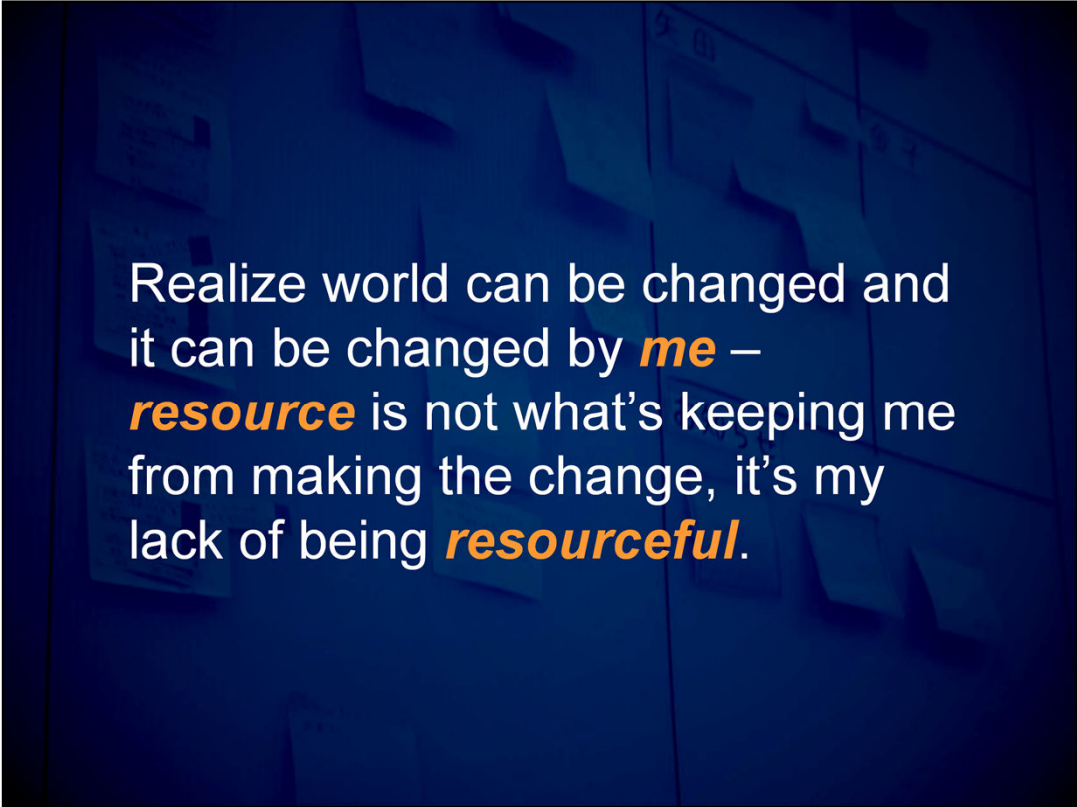
People are more satisfied when they take part in an effort

- Experience over books, lectures, workshops
- Give time to let everybody participate

Books, lectures, and workshops offer basic knowledge. However, people are more motivated on things they take part. To succeed in adopting agile in a workplace, give time and go through several sprints to get comfortable customizing practices. Get individuals to participate and offer their inputs.



Don't give up. Even if difference issues cannot be resolved now, it may be resolved as project environment change. Always keep changing the project environment so issues may become resolvable in the future.



Realize world can be changed and  
it can be changed by **me** –  
**resource** is not what's keeping me  
from making the change, it's my  
lack of being **resourceful**.

Agile to me is about what I **can do** instead of what I **can't** do.

Something may not be accomplished easily, but most things can be done  
if you really put an effort into it.

## Lessons Learned (recap)

1. Continuously adapt
2. Focus on **why** instead of **what** the differences are
3. Resolve **few** issues at a time
4. Give time to let issues get resolved
5. Don't give up!



Consultants are able to assist a project see the actual problem.

They, however, are not able to come in and give instructions to fully resolve all your project's problems.

A **coach** is also able to give advice to help members see a possible direction to solving the problem and to check if they are going in the right direction.

Coach only provides basic guidelines, direction we can take to solve **our** problems **ourselves**. It's our problem – we have to solve it ourselves. Transferring responsibilities is not the solution.



Because each project is different  
with their own diverse members.

Members have to adjust and find  
their own solution.

Only human can make other human happy. We can use tools such as Internet to assist us communicate and interact with other human, but we still need human on both ends.

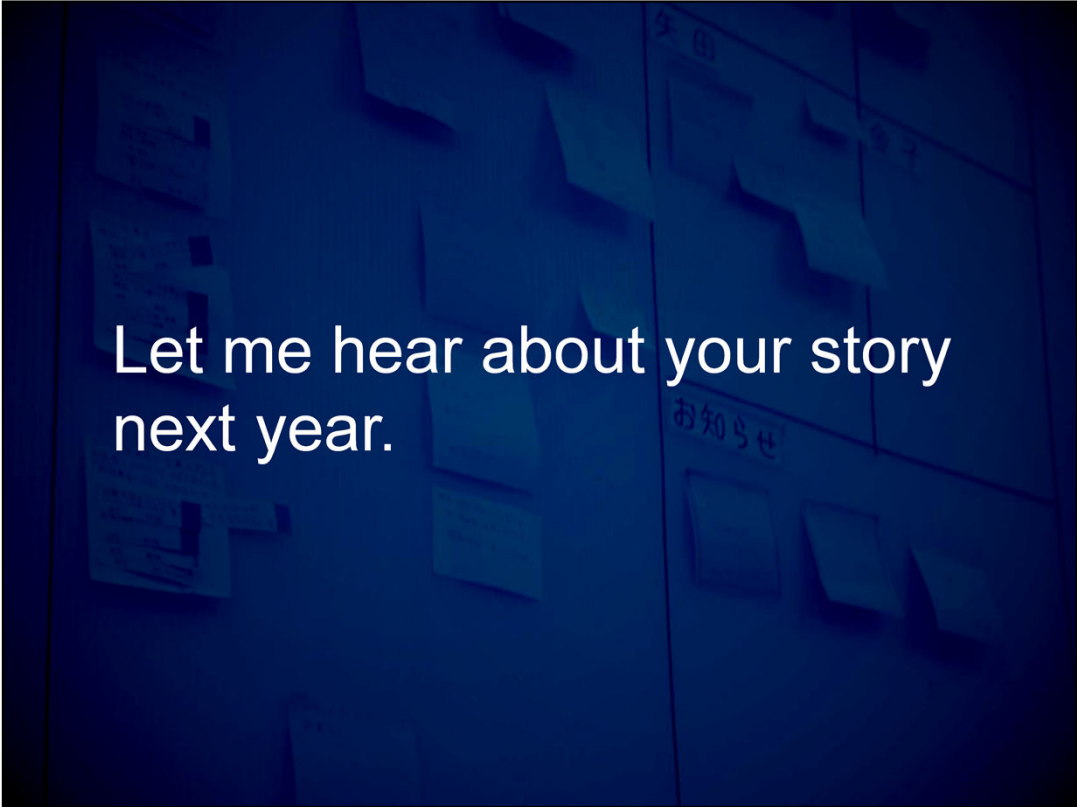
We were able to go beyond the difficulties of off-shore development and were able to gain advantages by adopting and adapting agile practices. We were able to overcome social differences in project members by promoting communications between members. We adopted Scrum, but it wasn't just assigning roles and process that really improved the project - it was creating an environment and practices where members were urged to communicate more and understand each other.

Some of the practices we found successful may not work in other environments, but our experience of continuously adapting roles and practices to take advantages of changing the relationship between members seems to be a general rule that can be followed. It was not just enough to define a process and setup tools to encourage better communication between project members - it was necessary to change our practices as well to encourage members to take advantage of them. As a result, we were able to increase the quality of the created deliverables with members more satisfied with the project.



I've now told you our story  
about what **WE** did

You'll have to think about  
what **YOU** can do

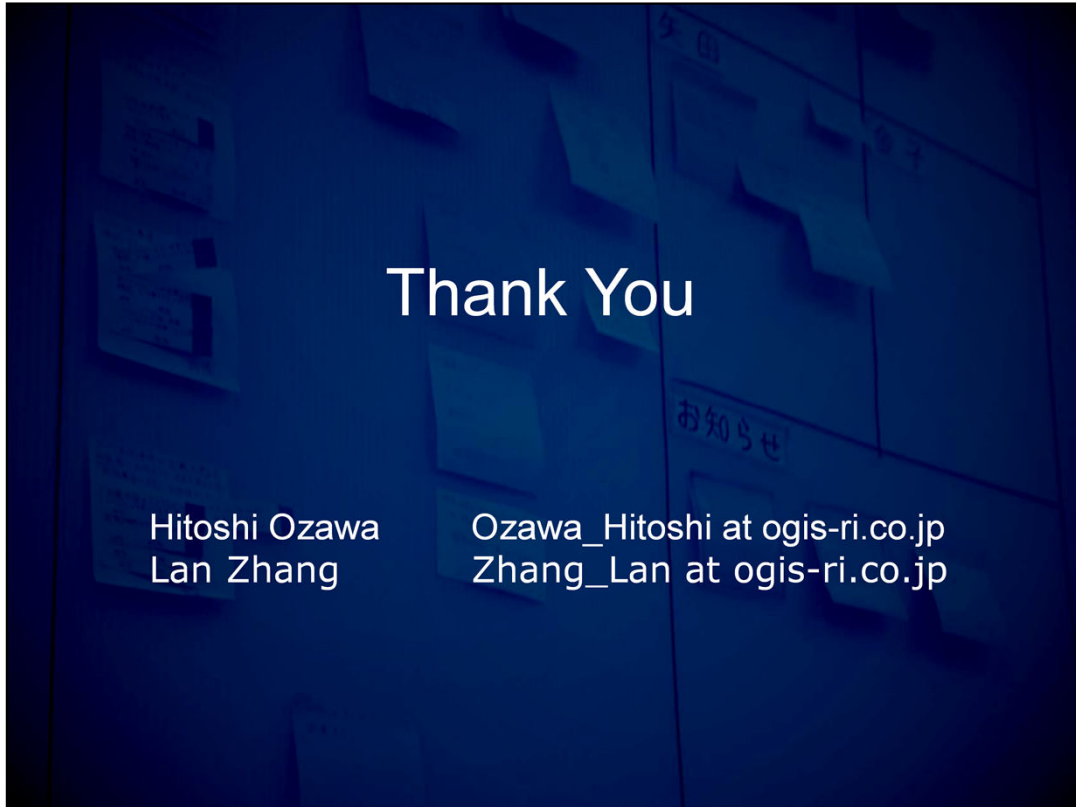


Let me hear about your story  
next year.





Back to the original question, the question I should have been asking should have been on how to make members more happy because they are the ones who can make a project and all of us a success.



I want to thank Johanna Rothman who was our shepherd to guide us through.

Agile Alliance, Sponsors, Rebecca Wirfs-Brock and Nanette Brown who are co-chair on the experience reports

Chris Matts, Tim O'Connor, Susan Burk, Karen Greaves, Linda Rising for their reviews

And finally to you all for coming to this session.