Documentation as Format to Challenge Software Architecture in Extreme Programming Method

Kamyar abdolmohamadi , Mansour esmaeilpour , Mohammad Mehdi shirmohammadi

***Abstract*-As we know the software development methods are divided into two old and agile methods. Old methods which are based on design and program have many problems regarding the changes. Change of customer needs from system, change in technology development, and change in software development environment facing the software development to serious problems. Agile methods are samples of software development methods which emphasize on rapid progress, speed, and flexibility regarding the changes. But these methods also have weak points. Agile methods have diverse ways to development which one of the most popular one is extreme programming concerning its operation. This method also has both challenges and weaknesses. One of these challenges is related to little attention to qualitative characteristics and software architecture activities. Different solutions are suggested to solve this challenge. Each of these solutions has its own weaknesses and fails to present a suitable approach. In this article, we are going to present an approach to solve this challenge. The suggested approach is designed in a way which tries to find a relation between architecture and challenge way, and achieve to both qualitative characteristics and software architecture advantages so that it is compliant with the values and agility principles of extreme programming.**

***Index Terms*—agility, extreme programming method, software architecture, architecture activities, qualitative characteristics**

I. INTRODUCTION

Agile development method is one of the software development methods which emphasize on the speed and flexibility in confronting with the changes. As we know, the customer needs of system, the environment in which the software develops, and also the technology, changes during the time. We also know that the software develops for and by the people [2]. Since old methods which are based on design and program, cannot control these changes, the need to have a development method is of utmost importance. Today agile methods as a new version of development methods, try to resolve the problems of old methods. Anyhow, agile methods also have weaknesses among which the most important ones are related to software architecture and the lack of attention to software architecture advantages such as qualitative characteristics [1, 3, 4, 5, and 6].The main problem of old and traditional methods is the heavy documentation which exists in all stages of software development which leads to a delay throughout a software. Agile methods are lightweight processes which need little documentation and reduce the delay in development. In this method there are also strong and repetitive relations between customers and software developers which try to rapidly control the changes and reduce the required time and costs. Among agile development methods, extreme programming is one of the lightweight methodologies which are emphasized by software population in recent years and a lot of experts confirmed its superiority compared to traditional methods [7].

II. INTRODUCTION OF EXTREME PROGRAMMING METHOD

Extreme programming method is a software development policy which is shaped on the basis of four values: communication, simplicity, feedback [8], and dare, to which the value of respect was added in the year 2000[9]. Values, principles, activities, and roles of people alongside with a process model, underpin the basis of extreme programming method. Communication is the first value of extreme programming. Extreme programming values verbal communication [1, 10, and 12]. One of the failure reasons of software projects is the lack of desired relations. Simplicity is the second value of XP and it is expected that the simplest way is used to swing us to the goal. Investigating the customer reaction facing with the product and applying his/her views is very important. In extreme programming, the system will always be faced to a feedback. When a problem occurred in macro designing of system development, and as a consequence that problem is shown in testing a system, the team should maintain its coherence and should try to fix the related problem. The value of respect also emphasize on maintaining the development respect among team members [8]. In XP, development has been done in several publications and repetitions and at the end of each publication a set of needs is implemented. During each repetition, the programming couples program the account stories which were applied previously by the customers. Each programming couples receive a duty as an input and then try to design and test the duty and then development is done which the code reconstruction will be done by the programmers. When the application of duties finishes, it will be integrated with existing codes and the stability of the final codes is ensured. This process is called task completion. And since this process is done in parallel and without any

qualitative supervision by different couples, there is possibility of shaping weak architecture structures which affects qualitative properties and the software architecture will be highly problematic [11].

III. RELATED WORKS

Founders of extreme programming say that we paid attention to architecture enough and have the following reasons for that. According to Mr. Back, nailing and metaphor in extreme programming focus on system architecture. Nailing solution which resembles sampling is a simple program with the aim of solving a specified problem [8]. Architecture nailing is a sample of nailing used for solving architecture related confusions [8]. Mr. West & Mr. Hersbolb equate metaphor with architecture [11, 12]. XP teams present a usual view of how the program works and call it metaphor. In the best position, the metaphor is a narrative expression of how a program operates so that everyone could understand how the system works and where to search the intended action and to find a suitable place to add its intended action. Anyhow the existential philosophy of metaphor is providing a channel to a simple and common understanding among beneficiaries of the project [8]. While it shows architecture, communications, structures and the interactions between components.

IV. **4-** SOFTWARE ARCHITECTURE DOCUMENTATION

Software documentation is a written text delivered with computer software. These texts usually expound the software operation and the way of its use which have different meanings to different people in various roles and are usually used as a means to make a relationship between project people. It is one of the important parts of software engineering because if software is not documented, making changes will be difficult and sometimes impossible. In relation to documentation, the software engineering institution defines a format for documentation which includes seven parts. If documentation is based on this format, then it can be used instead of architecture. The primary view is the first part of documentation which is the graphic shape and shows the constitutive components and the relationship between these components. In the next part which is called elements catalog, there is a complete expression of components and the manner of their communications. In the third part the text chart, software interaction to environment, the structure and other parts of the software are expressed in a graphic manner. In the fourth part the changeable points are specified, those which mostly have the possibility to change. In the fifth part the logic of design is expressed and the last two parts called glossary and other information. The words and expressions are expressed in the glossary and some information about the writer, the history and … are presented in the other information part [1, 14].

V. 5- THE SUGGESTED APPROACH

Our approach is a quest to achieve software architecture advantages which the most important of them is to achieving qualitative characteristics in the agile method and the more effective interaction and achieving to a comprehensive architecture model without damaging values and extreme programming principles. So we should make a relationship between agility and architecture in order to approach software architecture to agility. As we know, one of the aims of architecture is documentation with the help of which we can provide an effective relationship between beneficiaries, so we used documentation for achieving the aims of this approach. One algorithm is suggested to create an agile approach in order to achieving to qualitative characteristics and software architecture advantages which considers the values and agility principles and extreme programming method. On the other hand it provides qualitative characteristics in an acceptable manner. In order to actualize this activity we should first pay attention to qualitative characteristics since as we suggested before, if no attention is paid to qualitative characteristics in architecture structures, the architecture will have a weak structure. In order to implement our activities first a modeling team under a supervision of a senior architect should be created. All of the members including beneficiaries, programmers, and … should be informed of the aim and motivation of the system development. It means that the entire environment of the system should be specified to analyze its qualitative characteristics. In other word there should be a general recognition of the domain. Then we decompose the system on the basis of qualitative properties and the following algorithm format and finally a system will be created based on qualitative characteristics, which includes the following steps:

1) First the functional needs and then the aim of system development are specified.

2) A top and macro model of the system is designed by the architect.

3) Due to the macro model system, we divide the system into several repetitions according to the rules of extreme programming.

4) We specify account stories for each repetition.

5) We find the qualitative characteristics of each account story and then add it to other accounts.

6) In the next stage, a time planning is specified for performing account stories. Project manager, development manager and the senior planner plan the order of account stories based on their dependence and work load of the development team.

7) Now we deliver this account story to the programming couple which is first implemented by the customer and also its qualitative characteristics are specified, and the couple are required to create a simple conceptual model based on documentation and should send it to the senior architect at the end of delegated task development.

8) At the end of each repetition, created models are composed by programmers and senior architects and a general model is created for each repetition based on documentation.

In this stage, the architect also can discover the weak points by revising the structure and present some solutions for that. Like the primary models, this model changes during the development and while faced with needs changes. In the event of encountering changes, we return to stage 4 and simply declare the structure change in the total architecture. With the addition of qualitative characteristics to the first stage, the primary architecture of the system is designed based on qualitative characteristics, because these qualitative characteristics specify the system structure. With performing the above activity, there is always a conceptual and coherent model which is available for all members of the development team.

VI. CASE STUDY

This study is the project of software implementation of product purchase system, and it means that the organization wants to know its employees ideas or a specific unit the product purchase matter. For doing so, a message which contains a discussed question with a format of yes/no is sent to the system of all the employees by the central system. The software implementation is the question asked from the development team by the customer. Now we are going to solve this problem by the suggested algorithm to understand that if the suggested algorithm has the ability to add qualitative characteristics and own a general and abstract model of the system at every moment or not. Also we want to know how the suggested algorithm works in the face of changes.

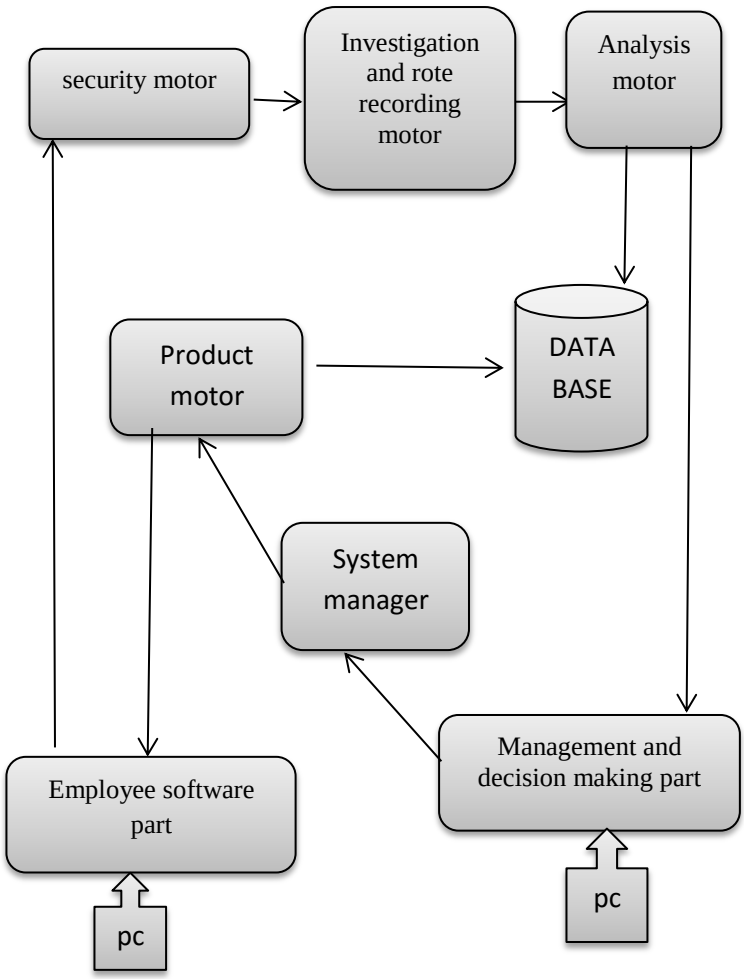
6-1 IMPLEMENTATION WITH THE SUGGESTED ALGORITHM

- **FIRST STAGE OF ALGORITHM**

In the first stage of algorithm, we specify the functional needs of system in a meeting with customer. By investigating the customer questions, we analyze the functional needs of system. When there is a need to make a decision about an important matter in the organization, software called central system sends a question about the matter which the organization wants to make a decision about, in a format of a message which exist in the cell phones of all the employees. This message is a two option yes/no question. The installed software on the employees systems is designed in a way which each employee can vote once. Irrelevant messages to the proposed question and also blank or damaged messages should be identified and deleted by the system. The central system should send the question and receive the answers and ideas, gather, analyze, and notify the result.

- SECOND STAGE OF ALGORITHM

In the second stage of algorithm, the senior architect presents a top and abstract model of the system so that it has a model before planning and implementation.



Shape 1-6 Simple and general architecture from poll system of product purchase

- **EXPLANATION OF THE PARTS OF ABOVE SHAPE**

The task of central system is to create poll, sending to users, doing security affair, record analysis, and declaring the results. Poll production motor designs a poll based on the application of the related unit. In security motor sent messages are investigated and according to central attendance system, attendance status of the personnel and the accuracy of sent messages are investigated by the authentic personnel. In investigation and sign vote motor, the information related to poll is investigated and recorded in the system. In result analysis and sending motor, the recorded results in data base are analyzed and a general form is produced to show the poll results and sent to management unit. In management and decision making part, the production process of a poll is investigated in administrative system, and after that the final confirmation is sent to management center. After investigating and identifying items related to each product, the product unit sends an application including the necessary explanations to the head of IT unit. After the confirmation, the head of IT unit export command of new poll to the software management part. In management and decision making part, an application including buy of intended products is sent to the head of the product. In software part which is related to employees based on the poll distribution strategy identified in the central part, a poll is distributed among related employees. After filling the poll form, the employees do the final record. Then the poll form is sent to the system management part and security motor.

- **THE THIRD STAGE OF ALGORITHM**

Regarding the system macro model, we divided the system into several repetitions based on rules. For designing the above model, we consider 3 repetitions, which we implemented the first repetition in this article. The first repetition designs a new poll on the basis of managers' applications, the second repetition includes sending, receiving, and recording the messages, and the third repetition do the security affairs.

- **THE FOURTH STAGE OF ALGORITHM**

We identify account stories for each repetition. As the naming of this repetition shows the purpose is to create a poll in office.

1) The system investigates the manager’s application to create a poll.

2) System should have the ability to create poll items based on related unit application.

3) System should identify persons or poll receivers units from the first.

- **THE FIFTH STAGE OF ALGORITHM**

We add qualitative characteristics to the account stories. Investigating these stories, the development team considers the qualitative features, performance, security, availability, and confidence. We can add qualitative characteristics and the usability for the account story number 2. It means that we plan the ordering way and the use of suitable literature to designing the questions of a poll relative to the responsive personnel. For the account story number 3 we can add security qualitative features, which mean that the sending direction of the questions is used for specific units or specific groups to prevent any poll from the invalid persons or units.

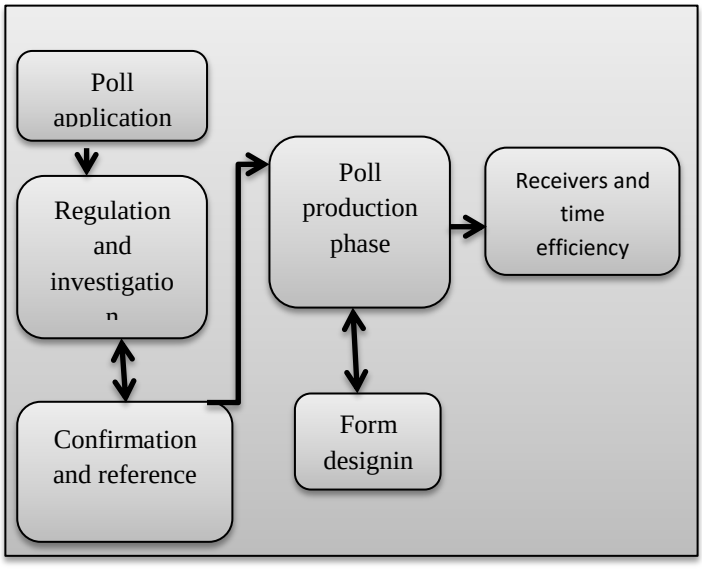
- **THE SIXTH STAGE OF ALGORITHM**

In this stage we plan the account stories based on their dependence to identify the order of their implementation. As specified, the part of software is put on the system of all employees and the other part is considered as a central system, so we divided the account stories into two parts, a part which is intended to be put on the employees systems and the part of central system the related account stories which we put on a special place.

- **THE SEVENTH STAGE OF ALGORITHM**

The implementation of account stories, as it is specified, we add qualitative characteristics simply to the account stories. Now we have account stories which not only have the customers' ideas but also the qualitative features. Now we deliver it to programming couples and at the end of each development, the programmer is required to create a simple conceptual model of architecture and send it to the senior architect.

- **THE FIRST REPETITION DOCUMENTATION**



Shape 2-6 first view of the first repetition in documentation

COMPONENTS CATALOG

1-Poll application: this application is written on behalf of the poll applicant unit which can include application items and their related explanations.

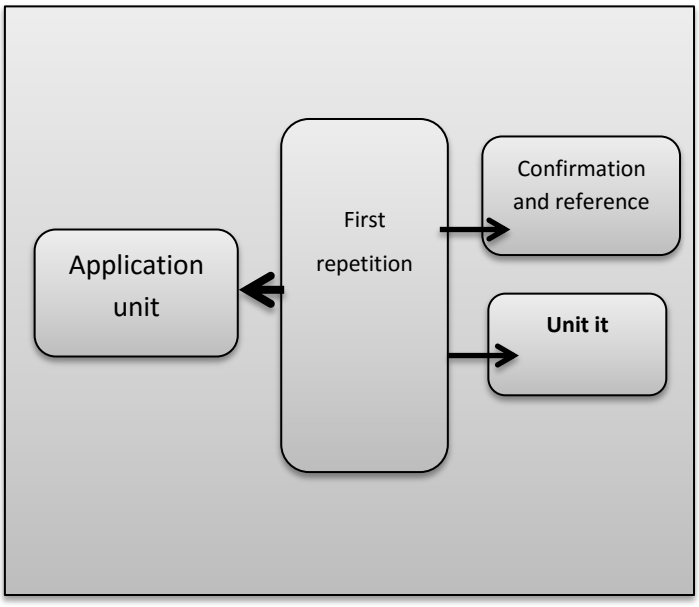
2-Regulation and investigation: this part is completed by the product unit and the poll items and questions and their related explanations are specified in this part in order to inserting in poll form.

3-Confirmation and reference: The IT unit investigates the received form from the product unit and if debugged, refers it again to the product unit and after the final confirmation, sends the application to the system management part.

4-Poll production phase: this phase is controlled by the system manager which can include IT part expert and it is formed from two parts which the explanation of each is sent to the related unit.

5-Form designing: format, questions ordering, outward designing and the poll form view is completed by the related IT expert.

6-Receivers: Receiver's persons or units of the poll are specified in this part.



VARIABILITY GUIDANCE

1-Investigation and regulation: there may be a communication between product unit and IT about how to design.

2-Form designing: there may be a communication between the technology expert and the system manager from the point of view of designing and outward shape.

3-Form designing: we can change the order of questions, outward designing, type of questions and options.

VI-2 INVESTIGATING THE SYSTEM PERFORMANCE CONFRONTING WITH THE CHANGES

We suppose that the customer needs of the system change, so we want to measure the system performance regarding the changes. It is supposed that a customer requests that the center system could deactivate the system of those employees who are absent in the office so that no one can vote with their systems. Development team can update the office systems before sending a message, and prevents the sending of messages to those systems which their users are not present in the office. With investigating this account story and which this need is among the duties of the third repetition which is the same confirmation of persons' identities, the development team refers to the third repetition and continues the algorithm from the beginning of the fourth stage; we want to add qualitative features to this account story. The intended qualitative feature of this account story is based on the usability functional needs. We will be informed from the presence or absence of the users by designing an interface user and prevent from sending the message to those users who are absent in the office. In the sixth algorithm stage, we have an account story, so we don't have the order of dependence, and composition of their implementation. In the seventh algorithm stage, we deliver the account stories to the programming couple which not only have the customers' ideas but also have the qualitative features, and the programming couple is required to create a simple conceptual model in spite of code writing, and send it to the senior architect. In the eighth algorithm stage, the senior architect will put the model created by the programming couples in to the specified place, and the account stories can be simply added or deleted or changed so that we can simply update the general model of the system.

VII. MEASURING THE SUGGESTED APPROACH

In 1993, a questionnaire was written for the company IBM in order to measuring the software usability by Louis. In this questionnaire, a series of questions with an index table are distributed among development team engineers. After studying the questions, the engineers are asked to carefully rate the questions with the help of index table and the answers of this questionnaire help us have an accurate understanding from software and know in which areas we were successful which need more work. We use Louis poll model to measure the suggested algorithm. So we need some criterions to measure how much this algorithm has approached to the main aims. To analyze the suggested method, a questionnaire including the intended questions (table 1-7) and 10 criterions in the area of software architecture advantages, principles and values of extreme programming, is distributed among four programmers of poll system development team.

In table 3-7 Programmers discuss about the extreme programming values which comply with algorithm regarding feedback and communication, and is in contradiction with simplicity which its reason is architecture and modeling of coded samples. So the programmers should completely learn the architecture knowledge, and this is the cost we pay to have an abstract model.

In table 4-7 Programmers discuss about software architecture advantages. Doing suggested activity in eighth stage, system has always a general and coherent model and can be used as a reference for beneficiaries' discussions, and programmers prove this theory completely. So we can claim that the suggested algorithm fulfill nearly all the intended criterions except simplicity.

Although the main aim was to achieve the advantages of software architecture and its concentration was on these advantages, but it is simply observed that the swiftness principles are considered well, and a suitable synchronization will be made between architecture activities and other extreme programming activities, and as a result a more suitable method will be made.

VIII. RESULTS AND CONCLUSION

From the beginning of extreme programming method appearance, a criticism was brought up to show that lack of attention to software architecture will cause problems for projects. During the recent years, several approaches are presented to resolve this weakness. Since none of the suggested solutions attend comprehensively to the matter, no efficient and codified method is presented to respond to multiple aspects of agility and software architecture. Considering both aspects of agility and software architecture, the suggested approach presents a solution based on agility values which is in comply with principles and activities of extreme programming method with the aim of achieving to qualitative characteristics.

Table1-7 questionnaire questions

Subject	Questions
Communication	Does it emphasize on communication increase between customer, programmer and other member?
Simplicity	Is the intended simplicity of xp preserved?
Feedback	Is there a possibility to receiving feedback at every moment?
Qualitative characteristics	Is it possible to access to qualitative characteristics based on this approach?
Abstract model	Is there are a top and macro model form system?
Beneficiaries relation	Can the created model be a reference to development team discussions?

Table2-7 indexes in programmer's statistical population

Parameter	score
Very agree	5
Agree	4
No idea	3
Disagree	2
Very disagree	1

Table 3-7 extreme programming values in programmer is statistical population

Responsive	communication	Simplicity	feedback
A	Very agree	Disagree	agree
B	Very agree	Very disagree	agree
C	Very agree	Disagree	Very agree
D	Very agree	Very disagree	Very agree
SUM	20	6	18

Table 4-7 software advantage in programmer's statistical population

Responsive	Qualitative characteristics	Abstract model	Beneficiaries relation
A	Very agree	Very agree	Very agree
B	Very agree	Very agree	Very agree
C	Very agree	Very agree	Very agree
D	Very agree	Very agree	Very agree
SUM	20	20	20

REFERENCE

[1]Shariflo a (2008), "embedding architectural practices into extreme programming"MSc. Computer group, shahid beheshti university, Tehran, Iran

[2] BroderickCrawford,ClaudioLe´ondelaBarra,RicardoSotoandEricMonfroy"AgileSoftware Engineering as Creative Work" CHASE 2012, Zurich, Switzerland IEEE,PP.20-26

[3] Turk D. France R. Rumpe B, 2012 ،"Limitations of Agile Software Processes "Proceedings of 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering، pp. 43-46، 2002.

[4] Beck K. Boehm B، 2003،"Agility through Discipline a Debate" Computer،vol. 36، no.6.

[5] Elssamadisy A. Schalliol G ، 2002. "Recognizing and responding to "bad smells" in extreme programming" Proceedings of the 24th International Conference on Software Engineering، pp. 617-622

[6] Jense R. M ler T. Sonder P, Tjr nehj G، 2006."Programming; Introducing "Developer Stories Proceedings of 7th International Conference on Agile Processes and Extreme Programming in Software Engineering، pp. 164 – 168

[7] Williams L, Upchurch R., 2001. "Extreme programming for software engineering education"31st ASEE/IEEE Frontiers in Education Conference

[8] Beck K. ، 2000.Extreme Programming Explained: Embrace Change، 1st ed. Addison-Wesel Professional

[9] Beck K, Andres C2004... Extreme Programming Explained: Embrace Change، 2nded. Addison-Wesley Professional،

[10] Boehm B. Turner R.، 2003. Balancing Agility and Discipline a Guide for the Perplexed، Addis Wesley

[11] Zafar Karimi,Sajjad Behzady, Ali Broumandnia, 2012, 'Achieving the Benefits of Agility in Software Architecture xp'،(JCSIT) Vol 4, No 5

[12] West D. Metaphor، 2002.، Architecture، and XP، Agile Alliance

[13] Herbsleb J. Root D. Tomayko J.، 2003. 'The extreme Programming (XP) Metaphor and Software Architecture'، Technical Report، Software Engineering Institute،Carnegie Mellon University

[14] Clements P.Bachmann. F. Bass L., Garlan D., 2002, Documenting Software Architectures: Views and Beyond, Addison Wesley.

First Author. M.A student, computer group, faculty of engineering, Islamic azad university, saveh, Iran, *Kamyarabd7@gmail.com*

Second Author.assistant professor, computer group, faculty of engineering, Islamic azad university, Hamadan, Iran, *esmaeilpour@iauh.ac.ir*

Third Author. Assistant professor, computer group, faculty of engineering, Islamic azad university, Hamadan, Iran, mmshirmohammadi@iauh.ac.ir