FORUM 2: THE ORGANIZATION OF SOFTWARE TEAMS IN THE QUEST FOR CONTINUOUS DELIVERY: A GROUNDED THEORY APPROACH

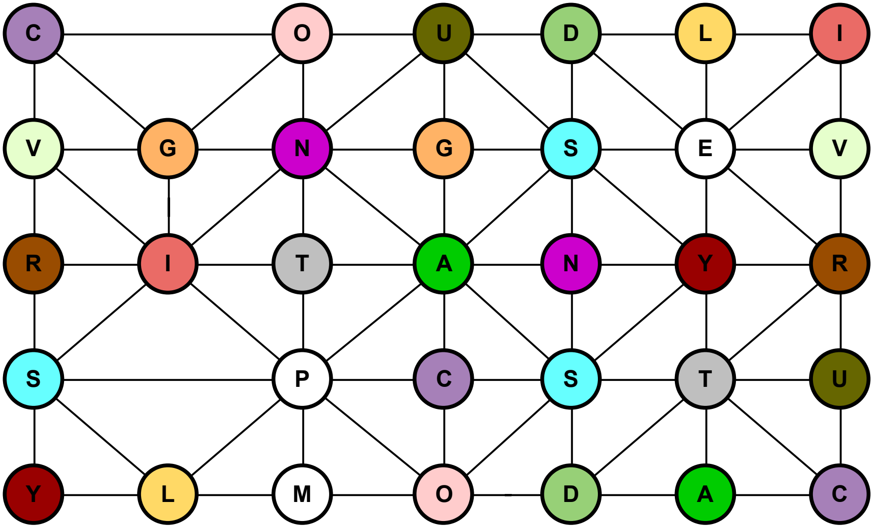
DIEGO VALENTÍN OSORIO MARÍN

JAIME ANDRÉS MONSALVE BALLESTEROS

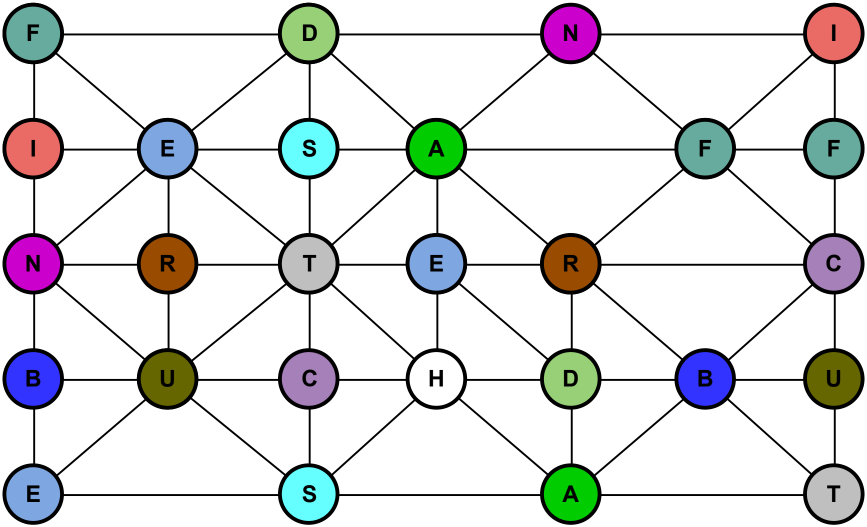
SANTIAGO CASTRO TABARES

FREDY ALBERTO LOAIZA OROZCO

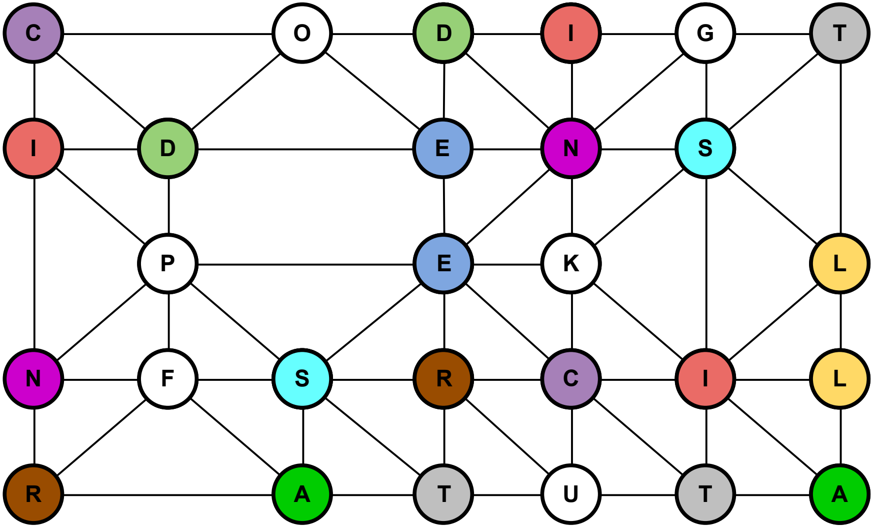
Computing and Decision Sciences Department. Faculty of Mines. Universidad Nacional de Colombia.



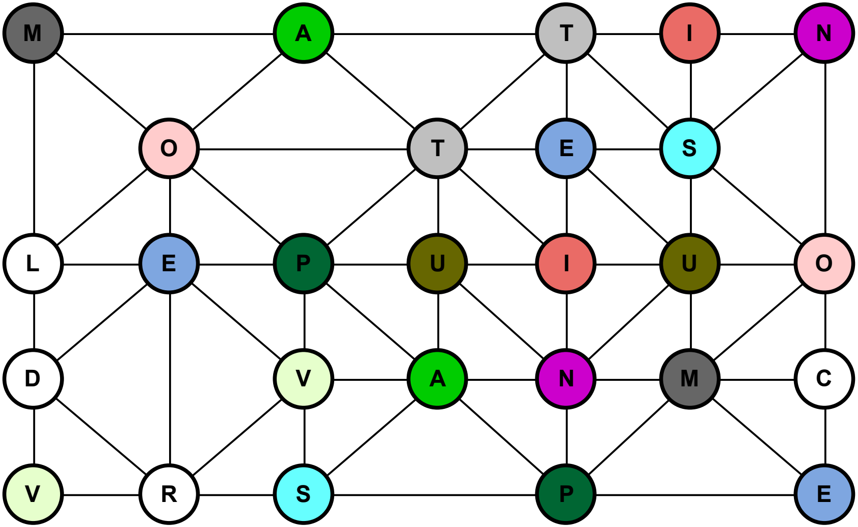
ADOPTING CONTINUOUS DELIVERY RELIEVES GAPS IN A COMPANY STRUCTURE



INFRASTRUCTURE STAFF CREATES AND FINE TUNES THE DATABASE



INFRASTRUCTURE SPECIALISTS NEED CODING SKILLS



DEVELOPERS AUTOMATE UNIT TESTS IN SOME COMPANIES

Glossary

A: Alpha.

Definition: An essential element of the software engineering endeavor that is relevant to an assessment of the progress and health of the endeavor.
Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 4. Paragraph 5.*

B: Bug

Definition: An error, flaw, or fault in a Software System that causes the system to fail to perform as required.
Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 190. Paragraph 3.*

C: Constraints.

Definition: Restrictions, policies, or regulatory requirements the team must comply with.
Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 4. Paragraph 5.*

D: Debugging

Definition: The process for detecting, locating and correcting faults in a computer program. Techniques include use of breakpoints , desk checking, dumbs, inspection, reversible execution, single-step operation, and traces.
Source: *IEEE Standard Glossary of Software Engineering Terminology - IEEE Std 610.12-1990, 1-55937-067-X, New York: Institute of Electrical and Electronics Engineers, 1991. Page 25, Paragraph 5.*

E: Error.

Definition: Human action that results in software containing a fault. Examples include omission or mis interpretation of user requirements in a software specification, incorrect translation or omission of a requirement in the design specification.
Source: *IEEE Standard Glossary of Software Engineering Terminology - IEEE Std 610.12-1990, 1-55937-067-X, New York: Institute of Electrical and Electronics Engineers, 1991. Page 18, Paragraph 6.*

F: Fault.

Definition: A manifestation of an error in software.
Source: *IEEE Standard Glossary of Software Engineering Terminology - IEEE Std 610.12-1990, 1-55937-067-X, New York: Institute of Electrical and Electronics Engineers, 1991. Page 19, Paragraph 13.*

G: Generality

Definition: Physical equipment used to process, store, or transmit computer programs or data.
Source: *IEEE Standard Glossary of Software Engineering Terminology - IEEE Std 610.12-1990, 1-55937-067-X, New York: Institute of Electrical and Electronics Engineers, 1991. Page 36, paragraph 14.*

H: Hardware

Definition: Physical equipment used in data processing, as opposed to computer programs, procedures, rules, and associated documentation.
Source: *IEEE Standard Glossary of Software Engineering Terminology - IEEE Std 610.12-1990, 1-55937-067-X, New York: Institute of Electrical and Electronics Engineers, 1991. Page 20, paragraph 14.*

I: Invariant.

Definition: Is a proposition about an instance of a language element which is true if the instance is used in a language construct as intended by the specification.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 5. Paragraph 2.*

K: Kernel.

Definition: Is a set of elements used to form a common ground for describing a software engineering endeavor.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 5. Paragraph 3.*

L: Leadership

Definition: This competency enables a person to inspire and motivate a group of people to achieve a successful conclusion to their work and to meet their objectives.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 63. Paragraph 2.*

M: Method.

Definition: Is the composition of a Kernel and a set of Practices to fulfill a specific purpose.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 81. Paragraph 2.*

N: Node.

Definition:The representation of a state or an event by means of a point on a diagram.

Source: *IEEE Standard Glossary of Software Engineering Terminology - IEEE Std 610.12-1990, 1-55937-067-X, New York: Institute of Electrical and Electronics Engineers, 1991. Page 25, Paragraph 10.*

O: Opportunity.

Definition: The set of circumstances that makes it appropriate to develop or change a software system.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 5. Paragraph 5.*

P: Pattern

Definition: Is a description of a structure in a practice.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 5. Paragraph 6.*

R: Requirements

Definition: What the software system must do to address the opportunity and satisfy the stakeholders.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 5. Paragraph 8.*

S: Stakeholders

Definition: The people, groups, or organizations that affect or are affected by a software system.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 5. Paragraph 11.*

T: Transition

Definition: Is a directed connection from one state in a state machine to a state in that state machine.

Source: *Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT. Page 6. Paragraph 3.*

U: Usability.

Definition: A measure of an executable software unit's or system's functionality.

Source: *ISO/IEC/IEEE 24765-2010(E), Systems and software engineering — Vocabulary 2010. Page 388. Paragraph 4*

V: Variable.

Definition: A quantity or data item whose value can change.

Source: *IEEE Standard Glossary of Software Engineering Terminology - IEEE Std 610.12-1990, 1-55937-067-X, New York: Institute of Electrical and Electronics Engineers, 1991. Page 80. Paragraph 16.*

Y: Yottabyte

Definition: Is a software-defined data center (SDDC) company headquartered in Bloomfield Township, Michigan.

Source: *Yottabyte Software Powers IT Infrastructure for West Island College. Page 1. Paragraph 12.*

DIS-ORGANIZATION GAME

For the idea of our game, we based our game on the classic "CLUE" where the main objective is to find the culprit of a murder, along with the murder weapon and the place where the crime was committed.

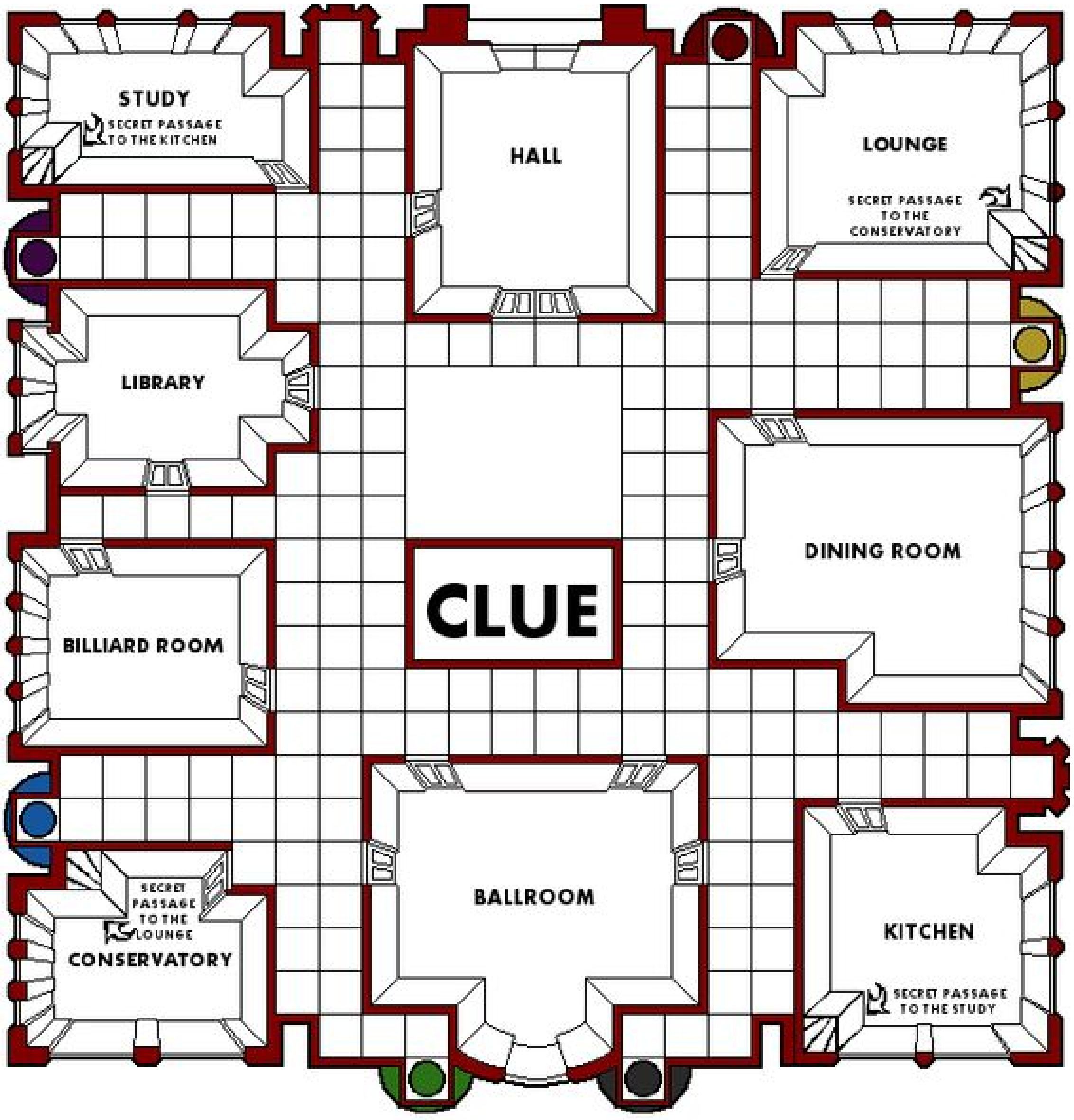

STUDY
SECRET PASSAGE TO THE KITCHEN
HALL
LOUNGE
SECRET PASSAGE TO THE CONSERVATORY
LIBRARY
DINING ROOM
BILLIARD ROOM
CLUE
BALLROOM
KITCHEN
SECRET PASSAGE TO THE STUDY
SECRET PASSAGE TO THE LOUNGE
CONSERVATORY

Figure 1 - "CLUE" game board

To organize the game, there are 3 types of cards:

- Cards with different characters participating in the game.
- Cards with different murder weapons.
- Cards with different places.

At the beginning of the game a card of each type is chosen randomly, being these three cards the ones that will tell us who, how and where the crime was committed (these cards will be hidden for all players).
During the course of the game, the characters or players, move in the squares depending on the number they roll with the dice. In case they reach a room, they will have to make an accusation in that place (i.e., they must say a character, and a murder weapon that possibly committed the crime in that place). The person who gets all three things right (Guilty character, murder weapon and place of the crime) wins.

Now, taking into account these characteristics of the original game, we will proceed to explain our game.

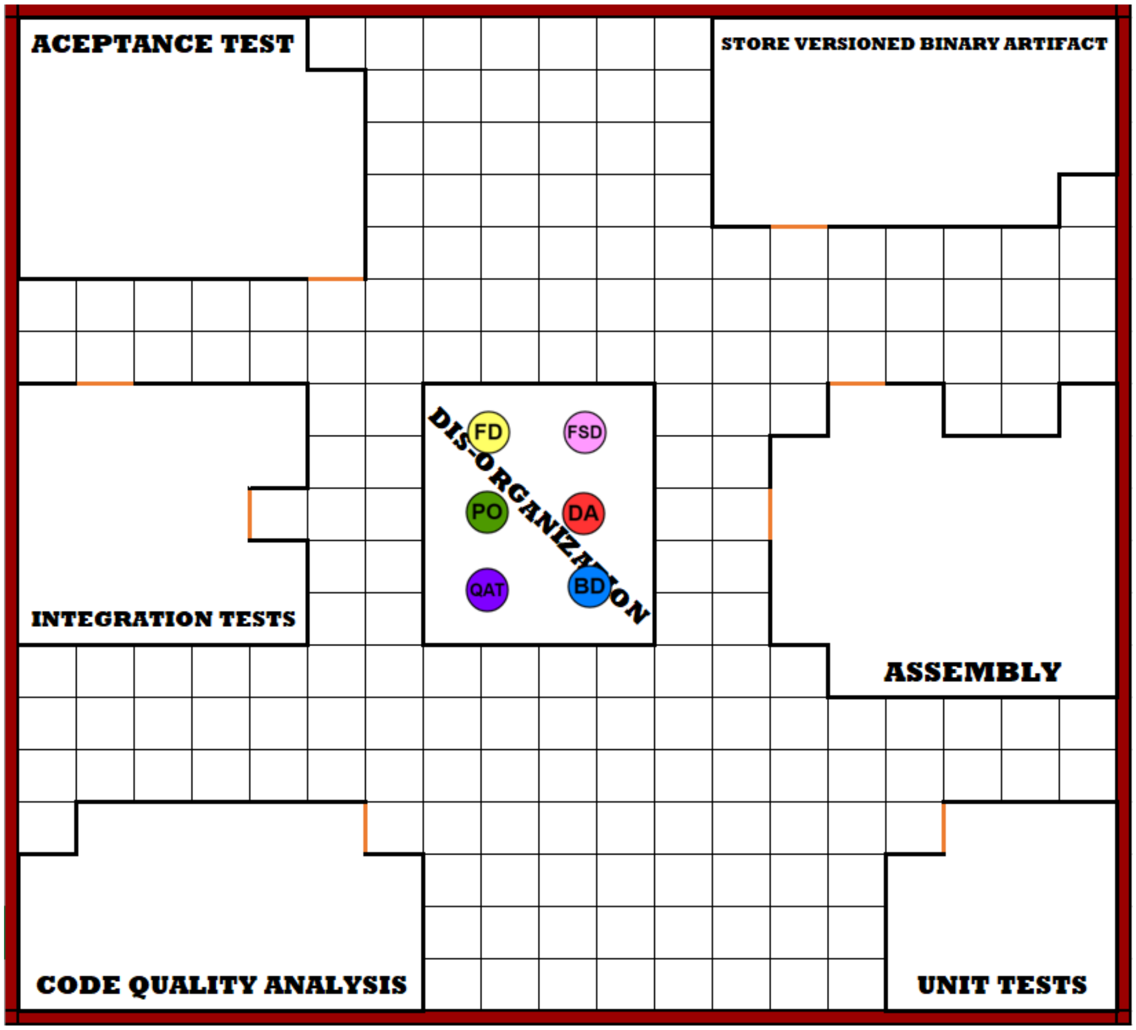


Figure 2 - Adapted board with locations being stages of continuous delivery

Taking into account the theme "roles of software teams in continuous delivery" our metaphor in the game will be the following (See Figure 2):

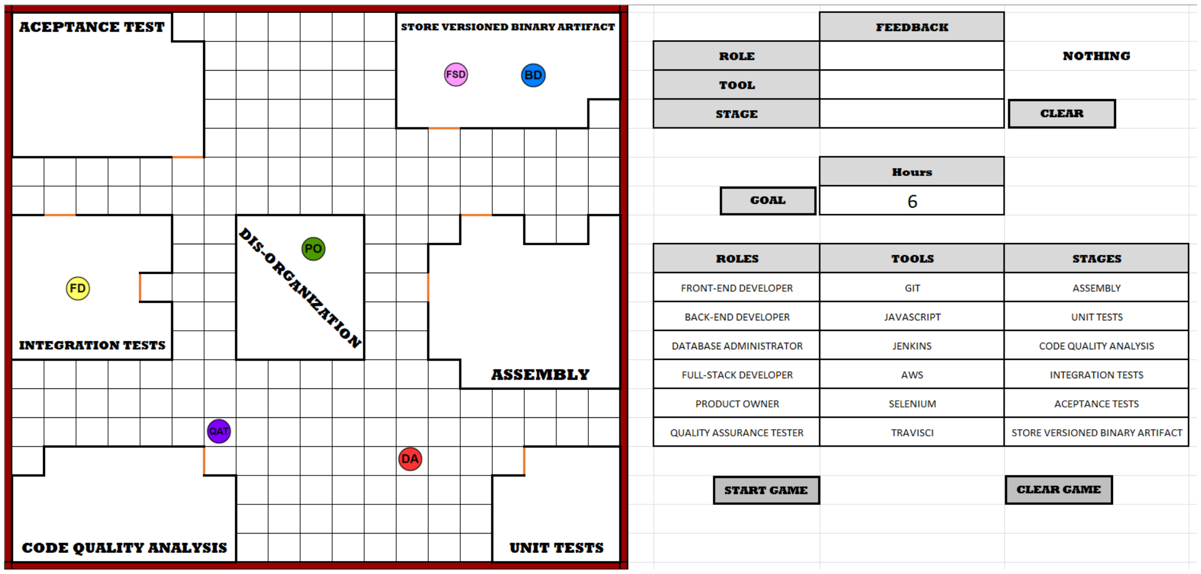
- The players will be the different roles involved in that software project:
    1. Front-End Developer.
    2. Back-End Developer.
    3. Database Administrator.
    4. Quality Assurance Tester.
    5. Full-Stack Developer.
    6. Product Owner.
- The board, instead of being a physical location, will be a software project that is under development in a growing company and is very disorganized.
- The different rooms will be stages of continuous delivery.
- For the murder analogy, in our case, what arises is a bug or problem in the software project.
- Instead of murder weapon, in our case it will be a tool that was used in the development of the project and is preventing the continuity of this (eg: Git, Jenkins, JUnit, PostgreSQL, etc).
- Instead of the place where the murder is committed, in our case it will be a stage of continuous delivery where the bug or problem occurred. (Since it is a rather disorganized company, it has difficulties to implement continuous delivery, this causes that there are tools guilty of the bug in stages where at first they would have nothing to do, but as said before, it is a disorganized company and therefore does not know this very well).
- The guilty character will be a role within the company that was largely to blame for this problem.
- For the mobility of players(roles) instead of using "**dice**" to get the number of **squares** to move, better to use a "**goal**" to get the number of **hours** to work.
- Other analogies:

  Accusation → Feedback

  Turn → Work Session

  Win → Deploy System
- Unlike the original game where there are cards for each player, in our case, being a centralized board and with all the information visible/hidden for everyone, all

possible roles, tools and continuous delivery stages are left in plain sight for all players, in addition, as the turns pass (work sessions) alternatives can be discarded (See Figure 3).



	FEEDBACK
ROLE	
TOOL	
STAGE	

	Hours
GOAL	6

ROLES	TOOLS	STAGES
FRONT-END DEVELOPER	GIT	ASSEMBLY
BACK-END DEVELOPER	JAVASCRIPT	UNIT TESTS
DATABASE ADMINISTRATOR	JENKINS	CODE QUALITY ANALYSIS
FULL-STACK DEVELOPER	AWS	INTEGRATION TESTS
PRODUCT OWNER	SELENIUM	ACEPTANCE TESTS
QUALITY ASSURANCE TESTER	TRAVISCI	STORE VERSIONED BINARY ARTIFACT

Figure 3 - All the information available for the players

- Tools:
  - Git
  - Selenium
  - Jenkins
  - AWS
  - TravisCI
  - JavaScript

The game is developed by work sessions, where each character or player (role) will move to a stage of continuous delivery(with hours of an specific goal), and there, will give feedback on the project (It is common for this to happen in this disorganized company because when a problem appears, everyone starts to blame each other), that is, he will say a role and a tool, and the stage will be the one where he is at that moment. The player who discovers the tool, role, and stage responsible for the bug or problem in the project, will have the honor of deploying it and being recognized for it.

Original game

TECHNICAL TEMPLATE	
I. GENERAL INFORMATION	
Technical Name	CLUE
Game Objective	The main objective is to find the culprit of a murder, along with the murder weapon and the place where the crime was committed.
Number of Players	3 to 6

II. MATERIALS

Name	Quantity	Description
Board	1	The board is composed of rooms and boxes. The idea is that the user reaches the rooms or a red question mark. If he lands on the red question mark, he drags a card that has an effect.
Red question mark cards	13	These cards are taken by the user when he lands on a square with the same symbol. They are usually cards that have effects, extra moves, go somewhere, or an extra turn.
Homicide Weapon Card	7	A card is drawn randomly, which will be our murder weapon. The rest of the cards will be given one by one to each player.
Killer card	7	A card is drawn randomly, which will be our murder weapon. The rest of the cards will be given one by one to each player.
Homicide Scene card	7	A card is drawn randomly, which will be our murder weapon. The rest of the cards will be given one by one to each player.
Dices	2	The dice tell us the number of squares the user has to go through.
Inference sheet	6	These are the annotations that the detectives will make, to keep all the information
Character pawns	6	These are the pieces that will be moving on the board.

III. GAME RULES

N°	Description
1	Players may move their pieces on all sides, except diagonally.
2	When a player arrives in a room, he must accuse someone, with a murder weapon in that room. For example, he must say, "In the garage, the purple pawn did the murder with the rope".
3	A player who has been accused, must show one of the cards he has, as long as within the accusation, he has in his possession the weapon, place or person. Otherwise, if he has none, he must say "I refrain from answering this question". Example following rule two, the purple pawn has in his possession the rope weapon and the garage, he decides to show the other user the garage.
4	If a player thinks he knows who the murderer was, the weapon and the place. He must go to the center of the board and must see the cards. If he does not lose
5	The players will not be able to share notes or spy on other users' notes.
6	Your cards must be kept secret. It should only be shown if it is in the case of rule 3.

IV. CRITERIA FOR WINNER SELECTION

Only the player who reaches the center of the board and guesses the murderer, murder weapon and crime scene wins.

Game

TECHNICAL TEMPLATE	
I. GENERAL INFORMATION	
Technical Name	DIS-ORGANIZATION
Game Objective	Find the problem or bug that prevents the deployment of the software project, taking into account that the company does not have a culture of continuous delivery.
Number of Players	3 to 6
II. EDUCATIONAL COMPONENT	
Thematic Name	Roles of software teams in continuous delivery
Purpose	Educational
Instructional Objectives	- Identify the roles and different stages involved in the adoption of continuous delivery. - Show the problem of implementing continuous delivery culture in small or disorganized companies.
Basic Concepts of the Thematic	Roles in software teams, continuous delivery, continuous integration

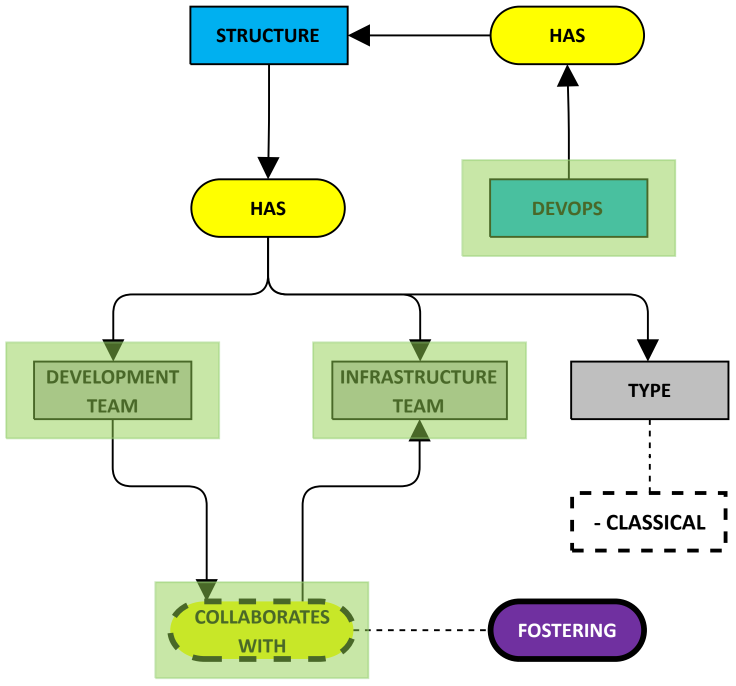
II. MATERIALS		
Name	**Quantity**	**Description**
Software Project	1	It is composed of 6 phases of continuous delivery and hours of workrtime.
Roles	6	These are the roles played by some of the company's employees, who will work depending on the hours of the goal.
Goal	1	The goal shows the number of hours invested.
Stage	6	These are the stages of continuous delivery
Tools	6	The company has 6 tools. Because of the disorganization, any role uses any of these tools

III. GAME RULES	
N°	**Description**
1	The game starts when a bug or problem arises in the development of the software project.
2	In each work section a role will set a goal, which will have a number of hours from 1 to 12.
3	Depending on the number of hours, this role will reach one of the stages of the continuous delivery. Being in this stage you will be able to give feedback on which role and with which tool, the bug took place in that stage.

IV. CRITERIA FOR WINNER SELECTION
If the roles's feedback is accurate (role, tool and stage) related to the bug or problem, then this role will have the honor of deploying the software project and be recognized for it.

Fragment #1

The classical DevOps structure focuses on collaboration among developers and the infrastructure team.

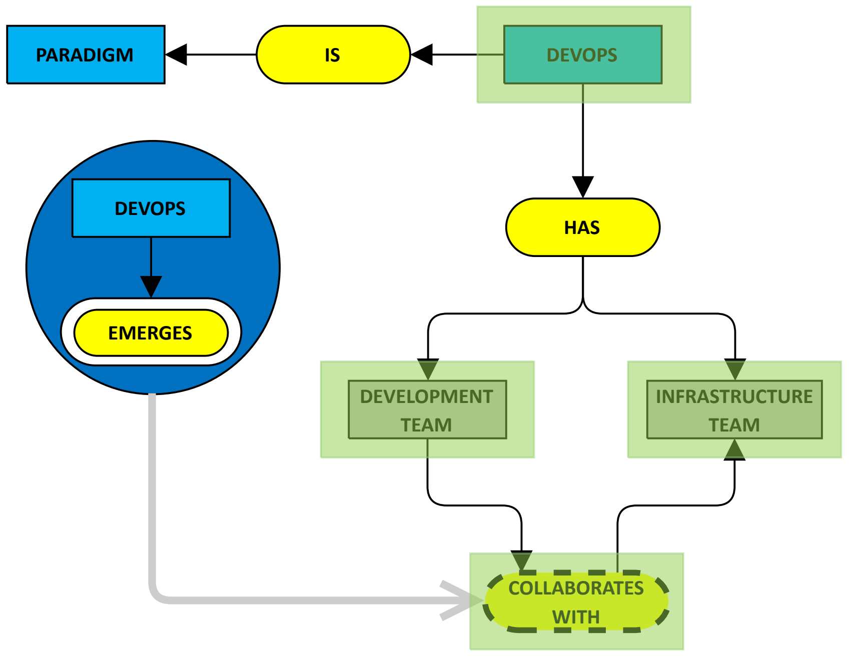


Reference

LEONARDO LEITE, GUSTAVO PINTO, FABIO KON, PAULO MEIRELLES, *The organization of software teams in the quest for continuous delivery: A grounded theory approach*, Information and Software Technology, ISSN 0950-5849, Volume 139, 106672, 2021.

Fragment #2

DevOps is an emerging paradigm that refer to a collaborative culture of development and operation teams.



Reference

SAIMA RAFI, MUHAMMAD AZEEM AKBAR, WU YU, AHMED ALSANAD, ABDU GUMAEI, MUHAMMAD UMER SARWAR, *Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis*, Applied Soft Computing, ISSN 1568-4946, Volume 116, 108377, 2022.

Fragment #1 Pre-conceptual Schema traceability table

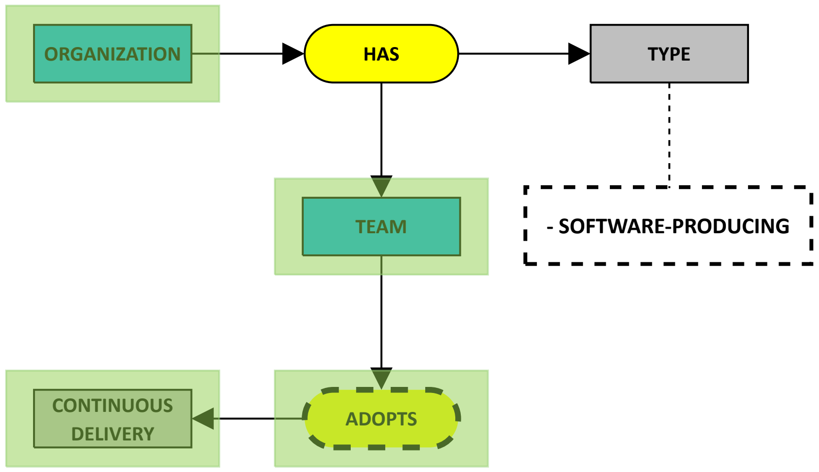
Original sound/Image/Text	Source	Location	Element	Kind of element	Observations
The classical DevOps structure focuses on collaboration among developers and the infrastructure team.	Text	The organization of software teams in the quest for continuous delivery: A grounded theory approach, page 7, paragraph 4	- DevOps has structure - Structure has development team - Structure has infrastructure team - Structure has type	Structural triad	- We inferred from the paragraph that DevOps has structure and both development team and infrastructure team are part of this structure. - We inferred from the context of the article that "developers" is related to "development team"
The classical DevOps structure focuses on collaboration among developers and the infrastructure team.	Text	The organization of software teams in the quest for continuous delivery: A grounded theory approach, page 7, paragraph 4	- Development team collaborates with infrastructure team	Dynamic triad	
The classical DevOps structure focuses on collaboration among developers and the infrastructure team.	Text	The organization of software teams in the quest for continuous delivery: A grounded theory approach, page 7, paragraph 4	- TYPE: -Classical	Note	- We inferred that classical is a attribute of structure and we modeled it like DevOps structure has a type and that type is classical
The classical DevOps structure focuses on collaboration among developers and the infrastructure team.	Text	The organization of software teams in the quest for continuous delivery: A grounded theory approach, page 7, paragraph 4	Fostering Development team collaborates with infrastructure team	Goal	- We inferred from the context of the article that "focuses on collaboration" is related to the need of the organization to "foster" this collaboration between teams

Fragment #2 Pre-conceptual Schema traceability table

Original sound/Image/Text	Source	Location	Element	Kind of element	Observations
DevOps is an emerging paradigm that refer to a collaborative culture of development and operation teams.	Text	Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis, page 1, paragraph 1	- DevOps has development team - DevOps has infrastructure team - DevOps is paradigm	Structural triad	- We can inferred from the context of the article that "operation team" is related to "infrastructure team"
DevOps is an emerging paradigm that refer to a collaborative culture of development and operation teams.	Text	Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis, page 1, paragraph 1	- Development team collaborates with infrastructure team	Dynamic triad	- We can interpret "collaborative culture" as the development team and operation team collaboration with each other
DevOps is an emerging paradigm that refer to a collaborative culture of development and operation teams.	Text	Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis, page 1, paragraph 1	DevOps emerges	Event	We modeled the part "DevOps is an emerging paradigm" as an event that iniciates the dynamic triad "Development team collaborates with infrastructure team" because when the DevOps culture emerged was with the idea of fostering the collaboration between both
DevOps is an emerging paradigm that refer to a collaborative culture of development and operation teams.	Text	Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis, page 1, paragraph 1	DevOps emerges, then development team collaborates with infrastructure team	Implication	We modeled the part "DevOps is an emerging paradigm" as an event that iniciates the dynamic triad "Development team collaborates with infrastructure team" because when the DevOps culture emerged was with the idea of fostering the collaboration between both

Fragment #3

Software-producing organizations have adopted continuous delivery.

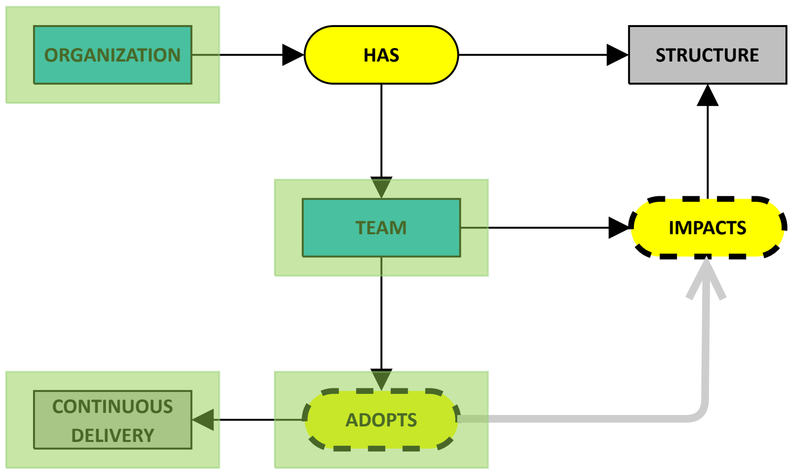


Reference

LEONARDO LEITE, GUSTAVO PINTO, FABIO KON, PAULO MEIRELLES, *The organization of software teams in the quest for continuous delivery: A grounded theory approach*, Information and Software Technology, ISSN 0950-5849, Volume 139, 106672, 2021.

Fragment #4

The adoption of continuous delivery has also an impact on the organizational structure.



Reference

LEONARDO LEITE, GUSTAVO PINTO, FABIO KON, PAULO MEIRELLES, Building a Theory of Software Teams Organization in a Continuous Delivery Context, *ICSE'20*, ISBN 978-1-4503-7122-3/20/05, ACM, Seoul, 296-297, 2020.

Fragment #3 Pre-conceptual Schema traceability table

Original sound/Image/Text	**Source**	**Location**	**Element**	**Kind of element**	**Observations**
Software-producing organizations have adopted continuous delivery.	Text	The organization of software teams in the quest for continuous delivery: A grounded theory approach, page 1, paragraph 1	- Organization has type - Organization has team	Structural triad	- We inferred from the context of the article that the organization has a team in charge to do some tasks
Software-producing organizations have adopted continuous delivery.	Text	The organization of software teams in the quest for continuous delivery: A grounded theory approach, page 1, paragraph 1	Team adopts continuous delivery	Dynamic triad	- We inferred from the context of the article that the organization itself is not the one who adopts "continuous delivery", in fact, the organization has several teams in charge of adopting it.
Software-producing organizations have adopted continuous delivery.	Text	The organization of software teams in the quest for continuous delivery: A grounded theory approach, page 1, paragraph 1	- TYPE: -Software-producing	Note	

Fragment #4 Pre-conceptual Schema traceability table

Original sound/Image/Text	**Source**	**Location**	**Element**	**Kind of element**	**Observations**
The adoption of continuous delivery has also an impact on the organizational structure	Text	Building a Theory of Software Teams Organization in a Continuous Delivery Context, page 1, paragraph 5	- Organization has structure - Organization has team	Structural triad	- We inferred from the context of the article that the organization has a team in charge to do some tasks
The adoption of continuous delivery has also an impact on the organizational structure	Text	Building a Theory of Software Teams Organization in a Continuous Delivery Context, page 1, paragraph 5	- Team impacts structure - Team adopts continuous delivery	Dynamic triad	- We inferred from the context of the article that the organization itself is not the one who adopts "continuous delivery", in fact, the organization has several teams in charge of adopting it.
The adoption of continuous delivery has also an impact on the organizational structure	Text	Building a Theory of Software Teams Organization in a Continuous Delivery Context, page 1, paragraph 5	team adopts continuous delivery, then team impacts structure	Implication	

References

- LEONARDO LEITE, GUSTAVO PINTO, FABIO KON, PAULO MEIRELLES, *The organization of software teams in the quest for continuous delivery: A grounded theory approach*, Information and Software Technology, ISSN 0950-5849, Volume 139, 106672, 2021.
- ISO/IEC/IEEE 24765-2010(E), Systems and software engineering — Vocabulary 2010.
- SAIMA RAFI, MUHAMMAD AZEEM AKBAR, WU YU, AHMED ALSANAD, ABDU GUMAEI, MUHAMMAD UMER SARWAR, *Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis*, Applied Soft Computing, ISSN 1568-4946, Volume 116, 108377, 2022.
- Yottabyte Software Powers IT Infrastructure for West Island College.
- Essence – Kernel and Language for Software Engineering Methods Version 1.2. SEMAT.
- LEONARDO LEITE, GUSTAVO PINTO, FABIO KON, PAULO MEIRELLES, Building a Theory of Software Teams Organization in a Continuous Delivery Context, *ICSE'20*, ISBN 978-1-4503-7122-3/20/05, ACM, Seoul, 296-297, 2020.
- ROSSEL, SANDER. “Continuous Integration, Delivery, and Deployment Foundations”, Akshata, Lobo, *Continuous Integration, Delivery, and Deployment*, ISBN 978-1-78728-661-0, Packt Publishing Ltd, Birmingham, 7-22, 2017.
- BROWN, SAMUEL. “Stages of Continuous Delivery Part 1: The Build.” *Oteemo*, 2 November 2017, https://oteemo.com/stages-continuous-delivery-part-1-build/. Accessed 12 April 2022.
- IEEE Standard Glossary of Software Engineering Terminology - IEEE Std 610.12-1990, 1-55937-067-X, New York: Institute of Electrical and Electronics Engineers, 1991