

# POKA-YOKE

---

RAKESH AMBRE – 04319401

L. SUNDARAVALLI – 04419801

# Presentation plan

---

- Introduction
- What is Mistake Proofing?
- Classifications by Dr. Shigeo Shingo
- ZQC Vs. SPC
- Case report – Push buttons
- Classifications of Poka-yoke devices
- Everyday Examples
- Characteristics of good Poka-yoke devices
- Poka-yoke in software
- Conclusions

# Introduction

---

- Pronounced as POH-kah YOH-kay
- Poka-yoke is the Japanese word for mistake-proofing
- Discrimination between mistakes and defects
- Mechanism that prevents a mistake from being made which is obvious at a glance.
- Examples:        Push buttons  
                          Welding nuts into panels

**TO ERR IS HUMAN NATURE**

**BUT**

---

**INTELLIGENCE IS ALSO HUMAN  
NATURE**

**SO**

**NEVER STOP ,FINDING WAYS OF  
NOT MAKING MISTAKES**

# POKA YOKE (Mistake Proofing)

---

## CAUSES

- DISTRACTION
- CONFUSION
- LACK OF PRACTICE
- UNCERTAINTY

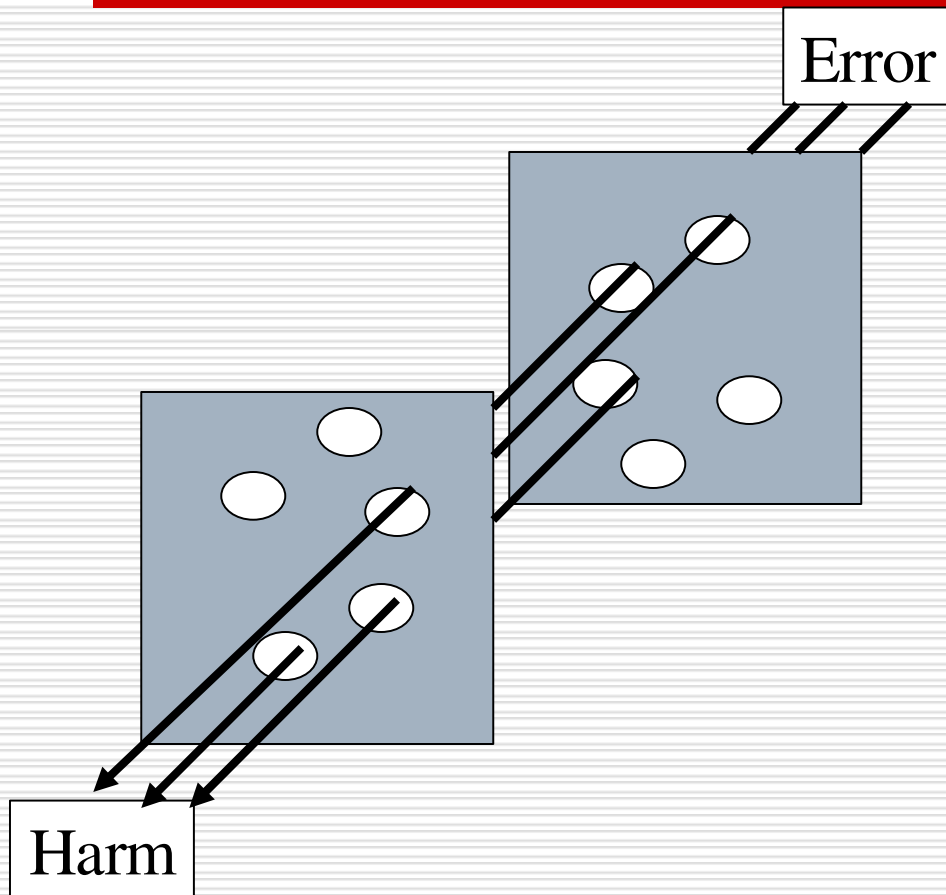
**INATTENTION  
ERROR**



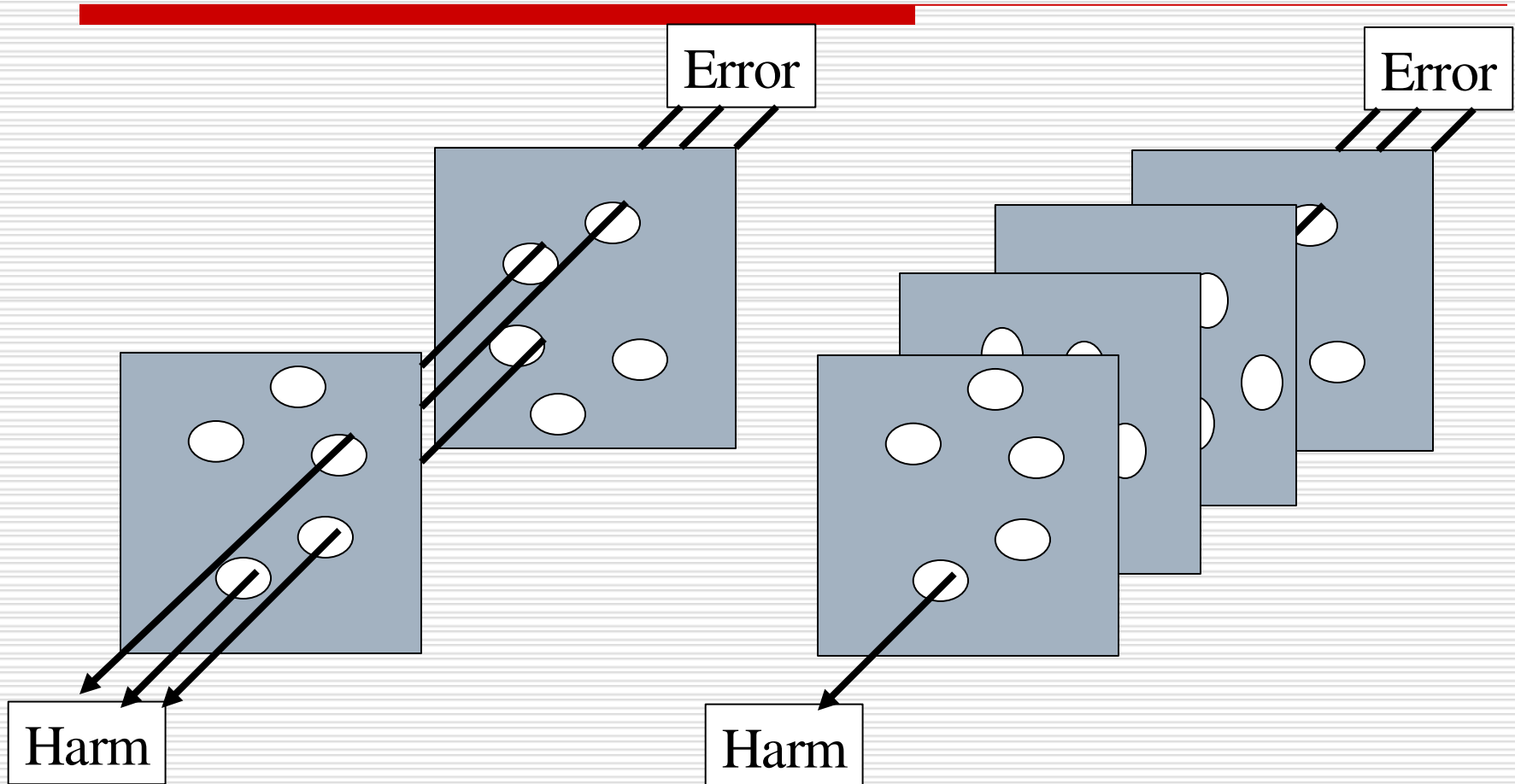
## EFFECTS

- DEFECTS (QUALITY)
- INSUFFICIENT NUMBER (QUANTITY)
- ACCIDENTS (SAFETY)
- WORK DELAY

# No System of Barriers is Perfect...



# No System of Barriers is Perfect... ...But improvements can be made



# WHAT IS MISTAKE PROOFING

---

A SYSTEM For Organizing Work that

- Excludes any risk of error even by the beginner
- Allows everyone to operate without mistake
- Prevents error that are about to occur

# Classifications by Dr. Shigeo Shingo

---

- Judgment inspection
  - Inspection in quality
- Informative inspection
  - Uses data gained from inspection to control the process and prevent defects. Examples: Traditional SPC and repeated checks and self-checks in ZQC
- Source inspection
  - Before the fact at source, before reaching customer or Forcing function

# ZQC Vs. SPC

---

- "...a look at SPC methods as they are actually applied shows that feedback and corrective action - the crucial aspects of informative inspections - are too slow to be fully effective."
- PokaYoke techniques to correct mistakes + Source inspection to prevent mistakes = ZQC

# Continued...

---

- ZQC is not as effective as SPC for defects that result from variance in measurement data
- ZQC is a special case of SPC for defects that result from variance in attribute data.
- ZQC's source inspection can be used effectively to eliminate mistakes and in conjunction with SPC to eliminate the recurrence of special causes.

# Case Report – Push buttons

---

- ❑ In the old method, a worker began by taking two springs out of a large parts box and then assembled a switch.
- ❑ In the new approach, a small dish is placed in front of the parts box and the worker's first task is to take two springs out of the box and place them on the dish. Then the worker assembles the switch. If any spring remains on the dish, then the worker knows that he or she has forgotten to insert it.

# PokaYoke devices

---

- A *prevention* device engineers the process so that it is impossible to make a mistake at all.
  - A 3.5 inch computer diskette
- A *detection* device signals the user when a mistake has been made, so that the user can quickly correct the problem
  - Small dish for push buttons



# Everyday Examples



# Which dial turns on the burner?

---



Stove A



Stove B

---

# Other examples

---

- Prevention devices
  - Microwaves
  - Washing machines
  - Central locking in modern cars
  - Key locks in mobiles
- Detection devices
  - Refrigerators
  - Messaging in most soft wares
  - Beeps in automobiles if key is left in ignition

# Characteristics of PokaYoke devices

---

- Simple and cheap
- Part of the process, permitting 100% inspection
- Placed close to where the mistakes occur, providing quick feedback.
- Designed to stop a particular mistake
- A detection device cannot provide a complete error proof solution
- Necessary and not a sufficient solution

# Common Mistake-proofing Devices

---

- Guide Pins
- Blinking lights and alarms
- Limit switches
- Proximity switches
- Counters
- Checklists

# PokaYoke in software

---

- Prevention devices
  - Development of programming languages
- Detection devices
  - Software testing

# Conclusions:

---

- Think simple
- Think specific
- Think attributes
- Think early
- Think responsive
- Think re-use



THANK YOU