Old-Fashion Rejuvenated - Software Handover Yesterday and Today!

Mira Kajko-Mattsson
KTH Royal Institute of Technology
SWEDEN
mekm2@kth.se

Agenda

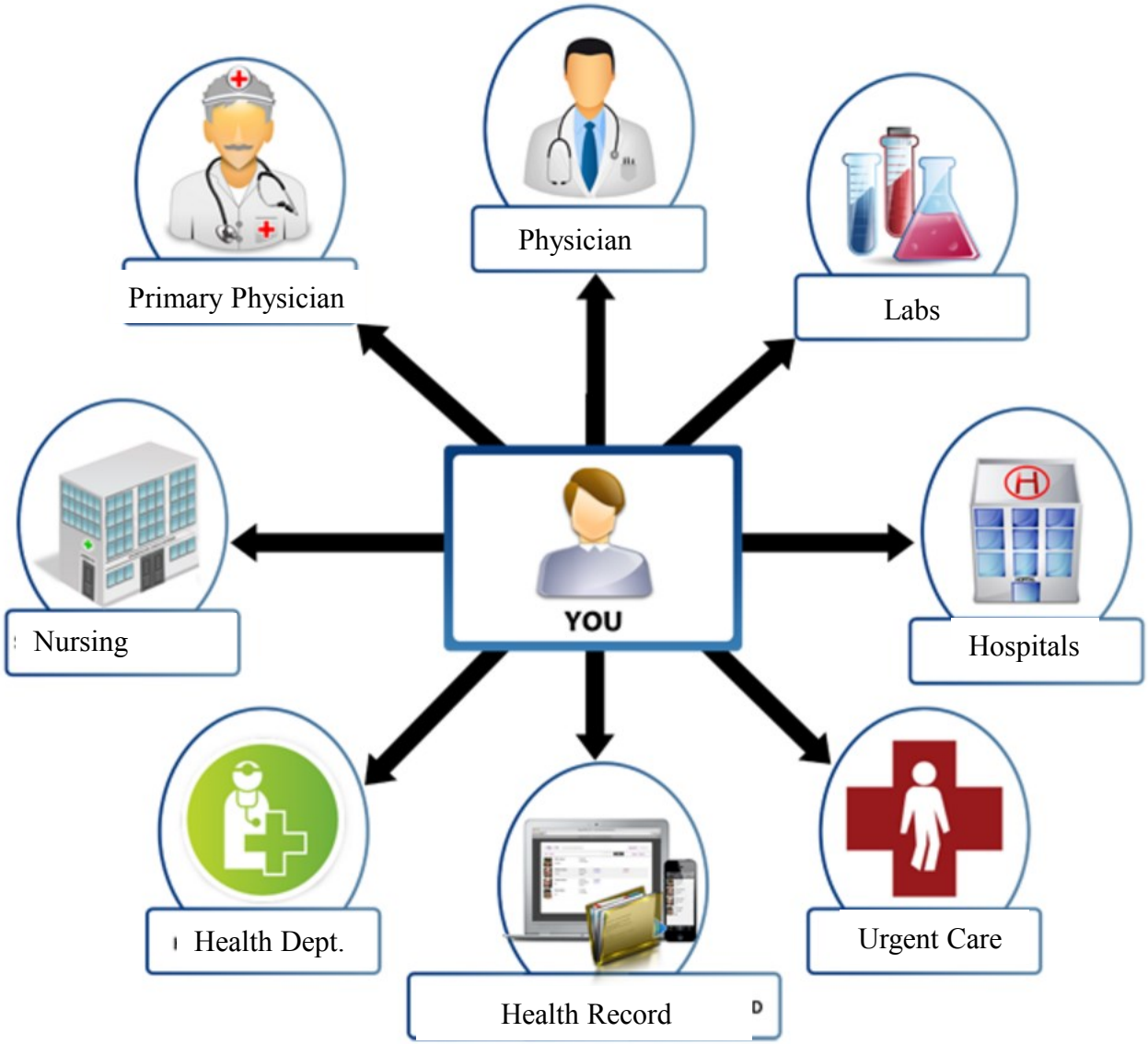
- Handover vs transition
- Problems
- Contexts
- EM^3: Handover Framework
- Future

This material is the result of collaboration with Dr. Salman Ahmad Khan, Lahore University, Pakistan.

Presidential transition

- The incoming president takes over the responsibilities of the outgoing president.
- New government personnel gets designated, trained, and prepared to take over the country in as smooth way as possible.
- If the presidential transition fails, then the country may get exposed to various internal and external threats.

Healthcare transition



Another example of a transition

Manufacturer

Post-sale support

- The manufacturer should not only transfer the support responsibilities but also:
  - knowledge about the cars
  - software and hardware needed for servicing the cars
  - provide replaceable components
  - documentation and operational instructions.

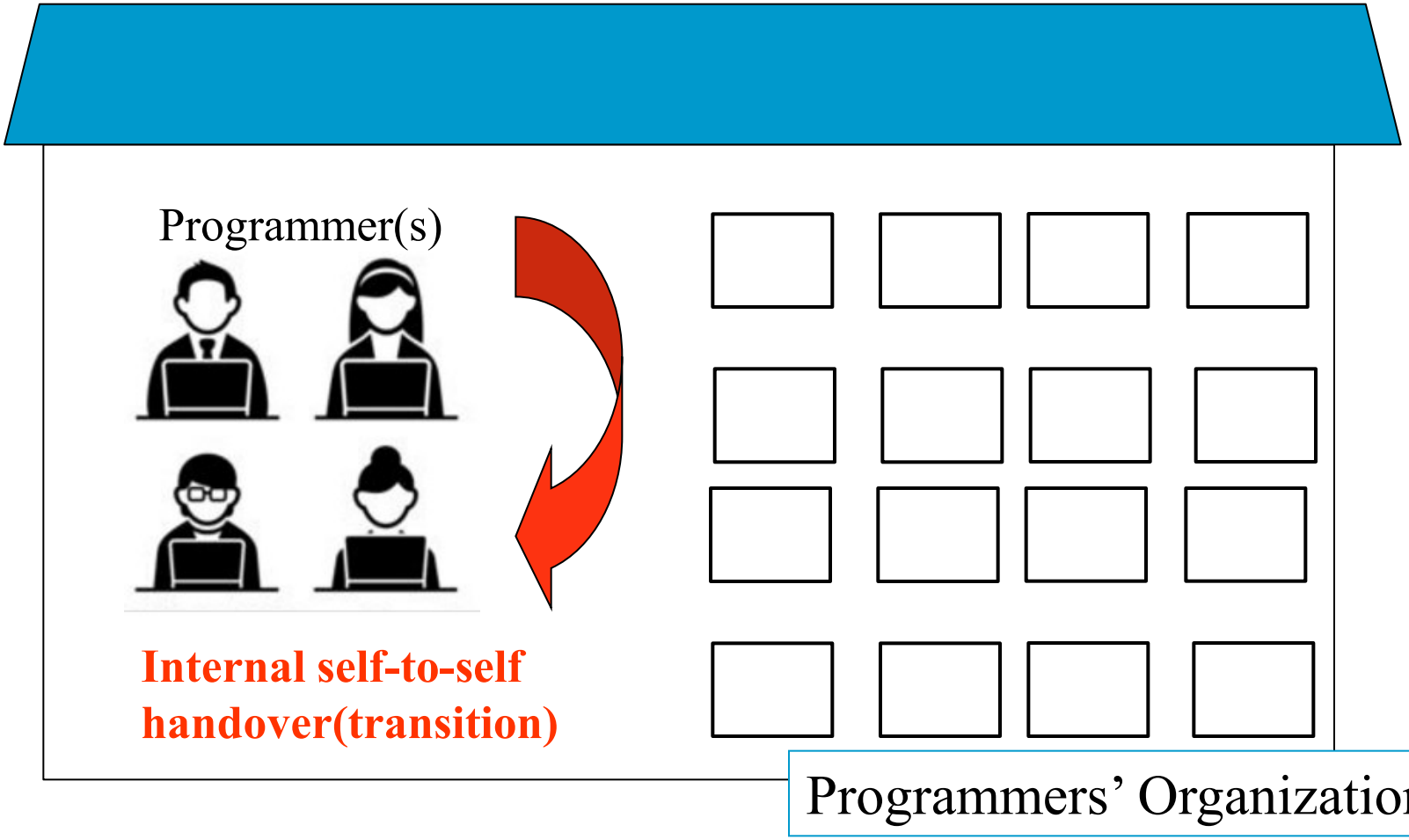
So what is software handover?

- ????

Research Uniqueness

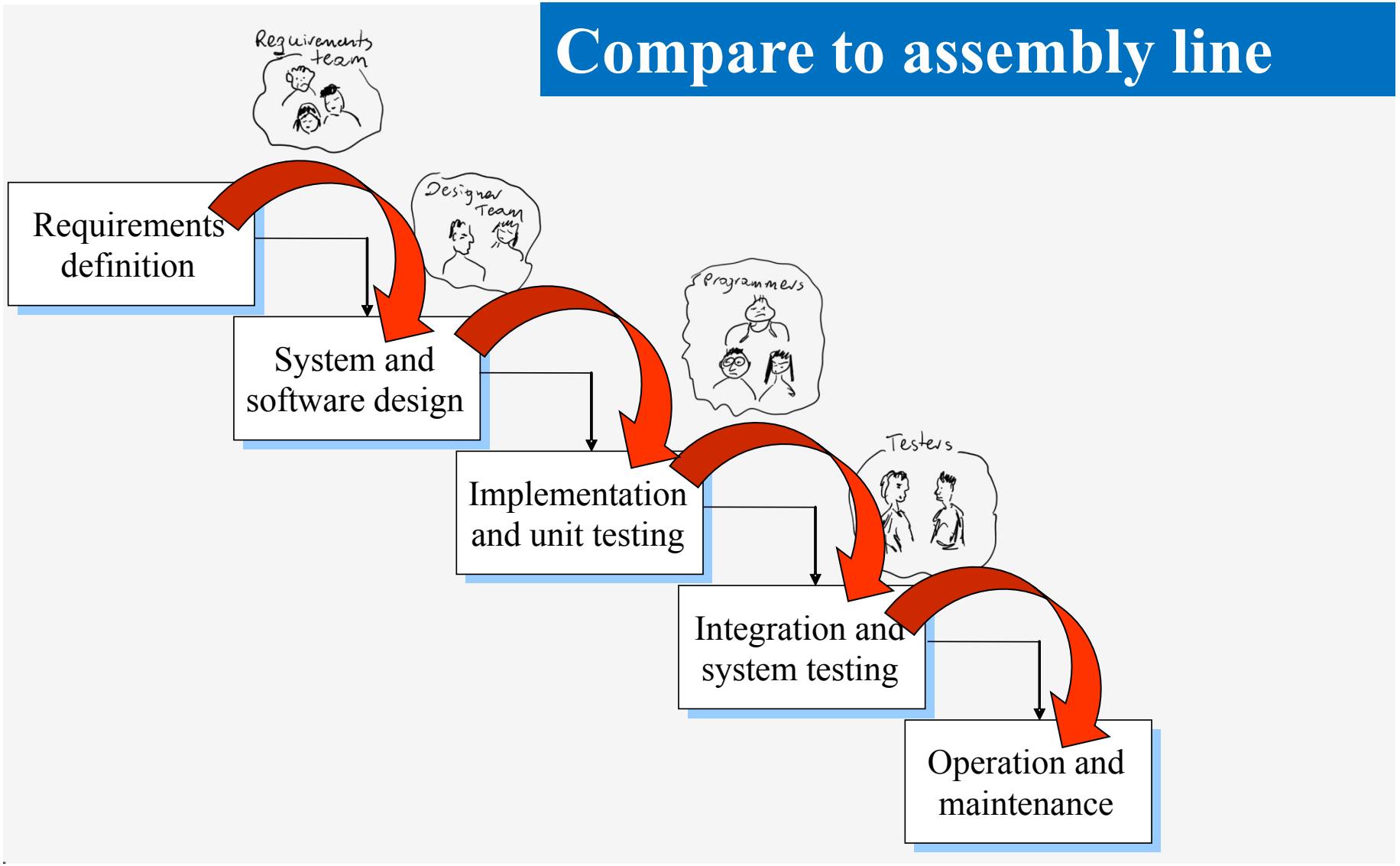
- Presently, very little research is being done on software handover.
- The main sources available
  - Thomas Pigoski's publications
  - Vollman's publications
  - ISO/IEC 14764 standard
  - ITIL – Information Technology Infrastructure Library
  - EM^3: Handover Framework

In ancient times

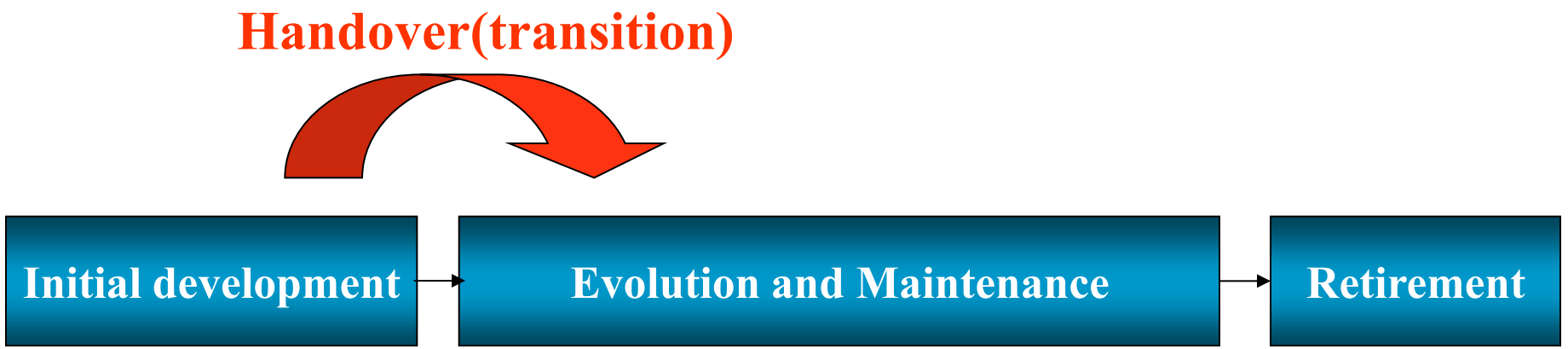


From 60s, 70s till now

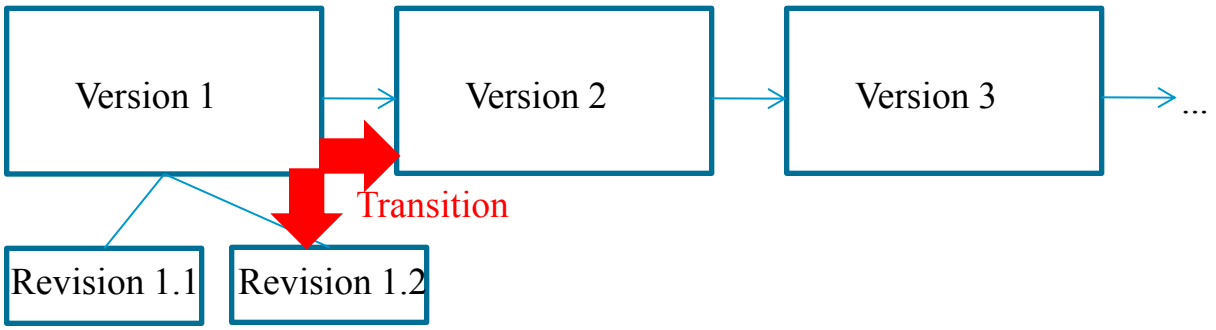
Compare to assembly line



Handover within lifecycle

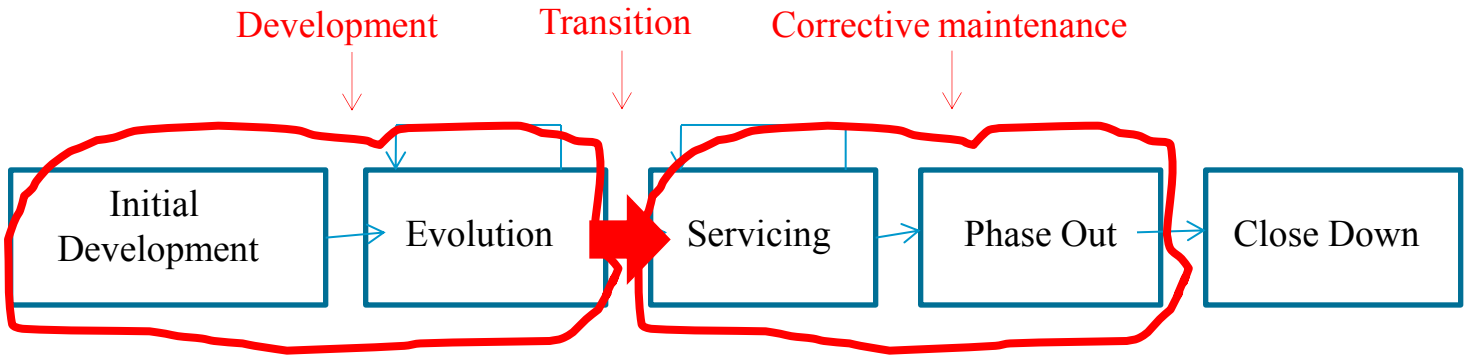


Handover within lifecycle



Transition Enhancive maintenance

Handover within lifecycle

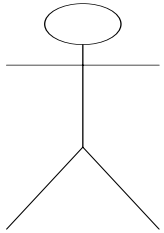


Development

Tra

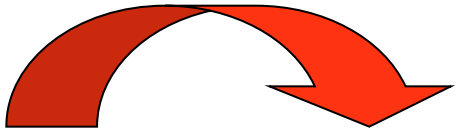
Handover

Customer



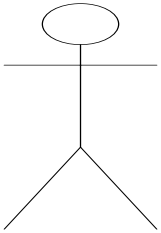
Acquires the system, and wants it built quickly and inexpensively.

Handover(transition)



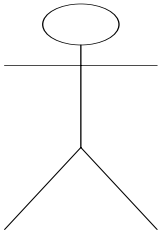
Initial development → Evolution and Maintenance → Retirement

Developer



Must develop the system, wants to deliver it on time and within budget.

Maintainer



Must live with the results of the development, and must maintain it for some extended period (far longer than the development).

Handover

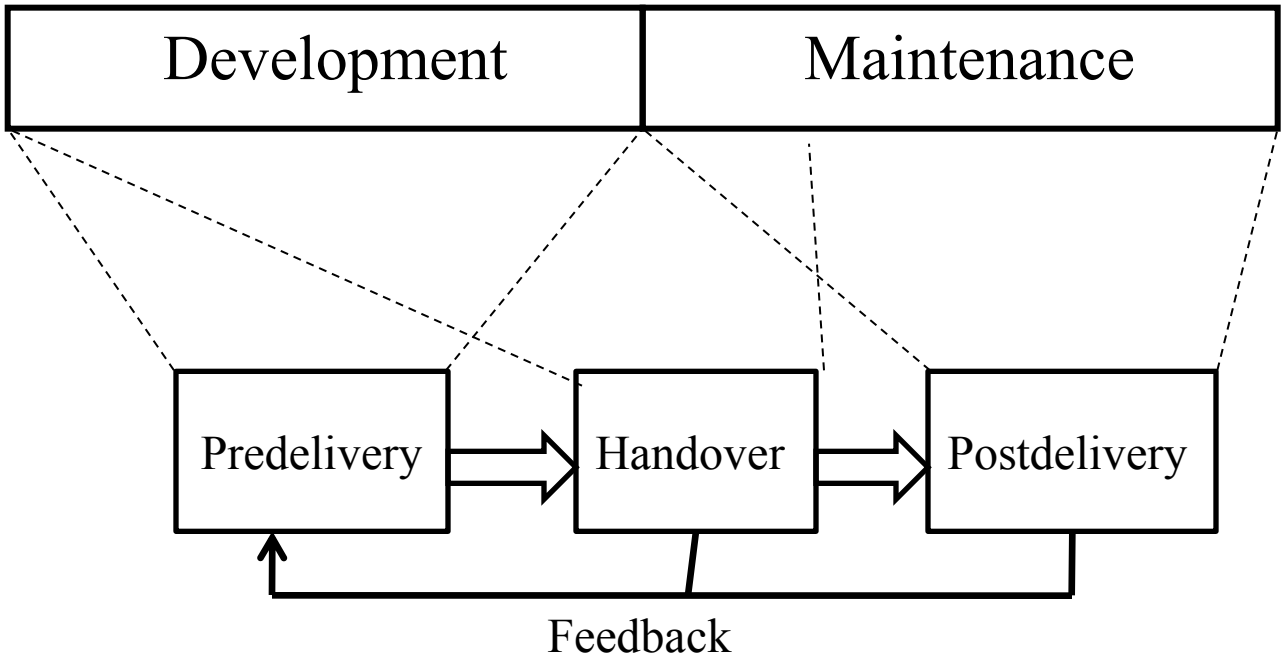
- Handover(transition) is a controlled and coordinated activity, during which the responsibilities of a software system are transferred from the team/organization performing software development to the team/organization performing post deployment software maintenance and support.

- Who conducts predelivery maintenance?

Handover Problems

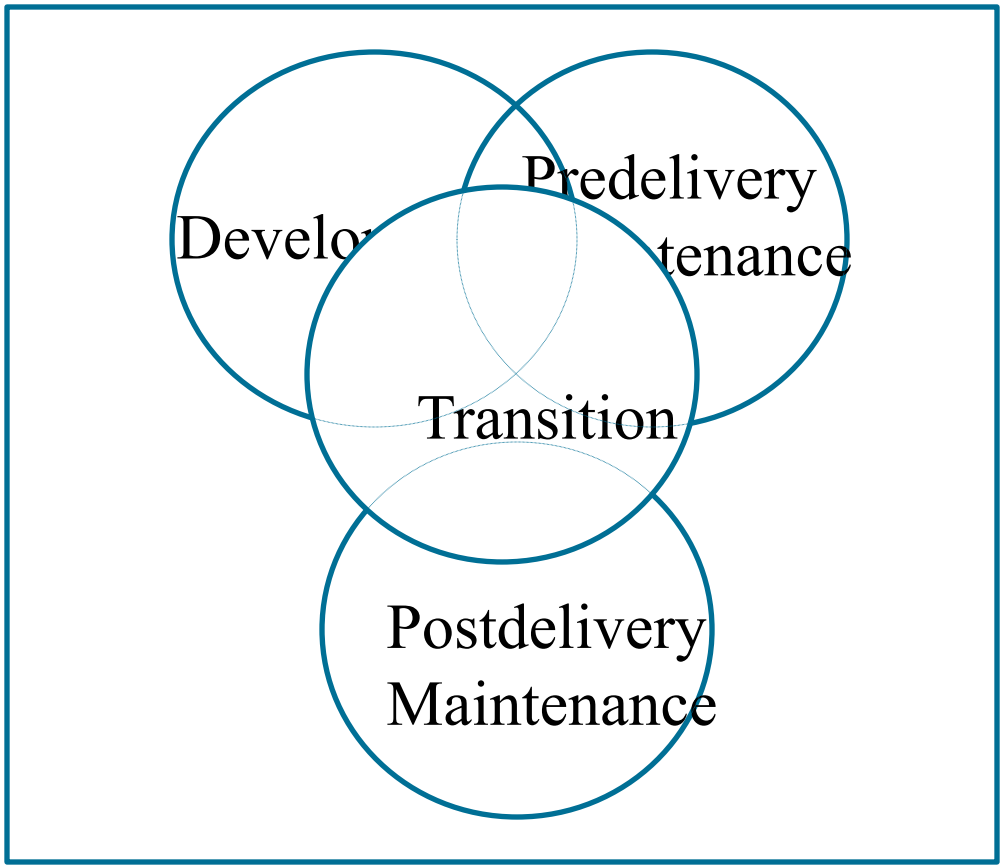
- Insufficient system knowledge
  - Both maintainers and support technicians
- Lack of domain knowledge
- Insufficient communication
- Inadequate or insufficient documentation
- Difficulties in tracking changes and in estimating ripple effect
- ….

Another way of illustrating software lifecycle



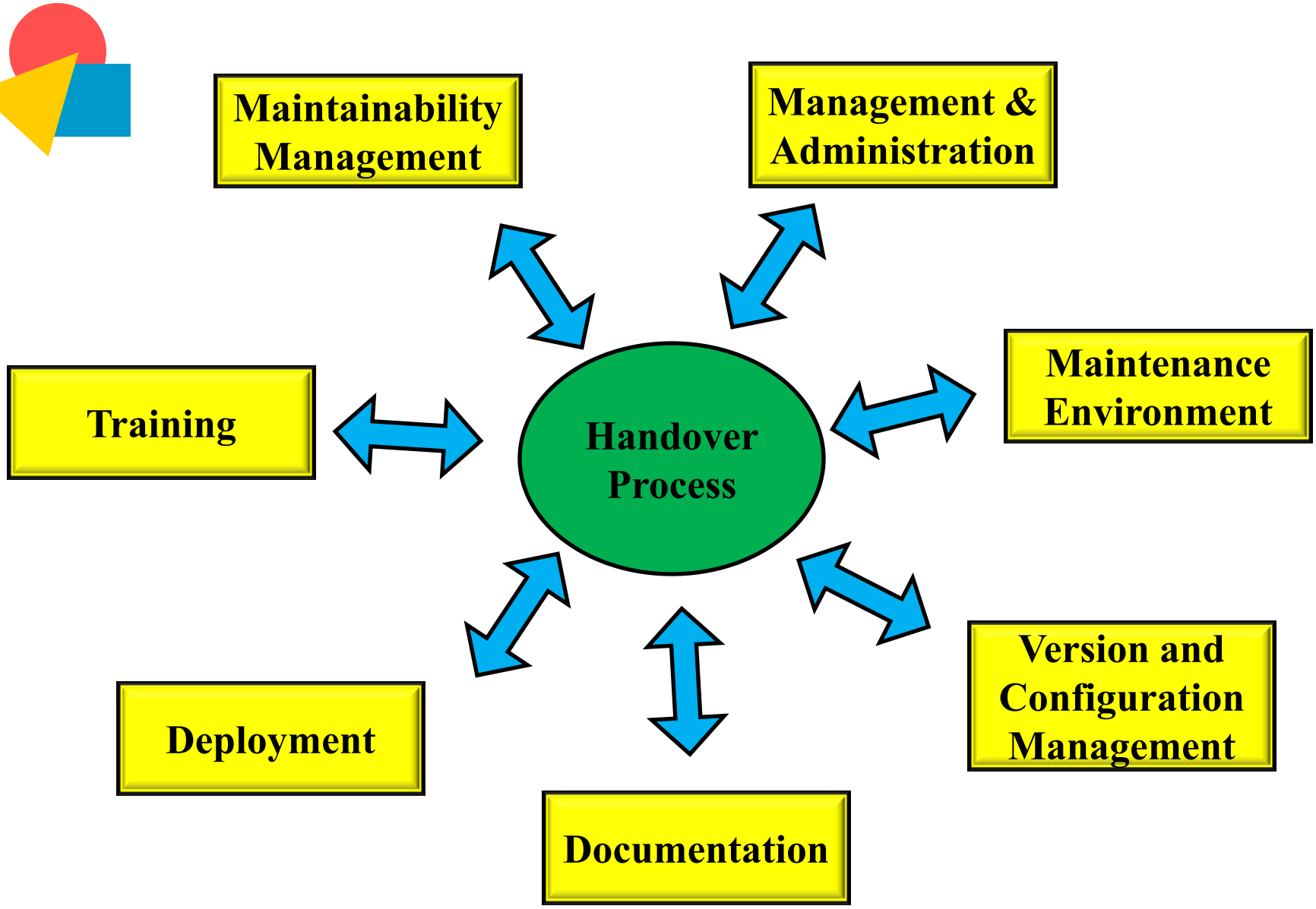
So, where do we place software handover?

Handover (transition) overlaps with…

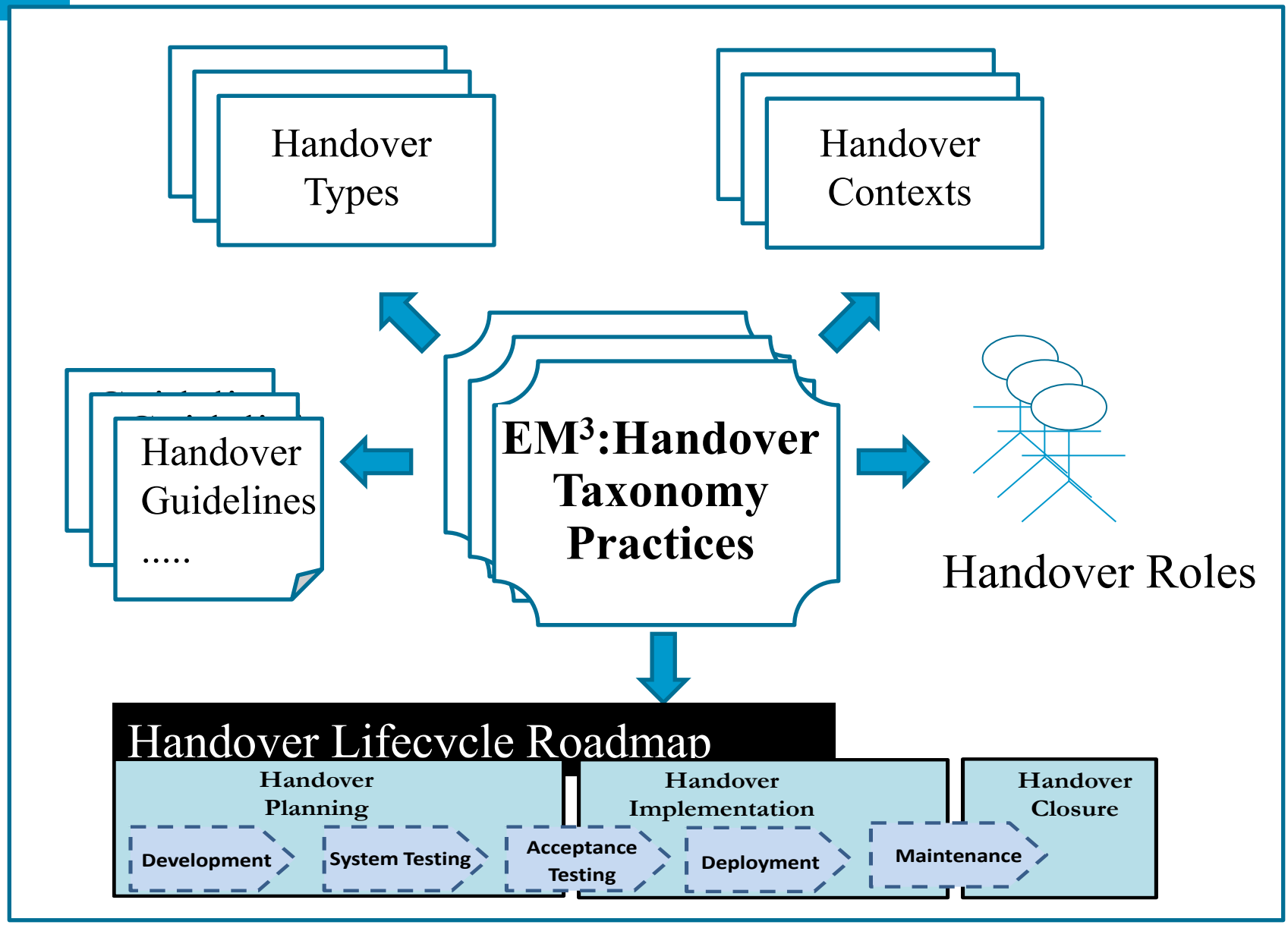


Predelivery activities

- Maintainability Planning
- Maintenance Planning
- Contract
- Education
- Evaluation of the development process

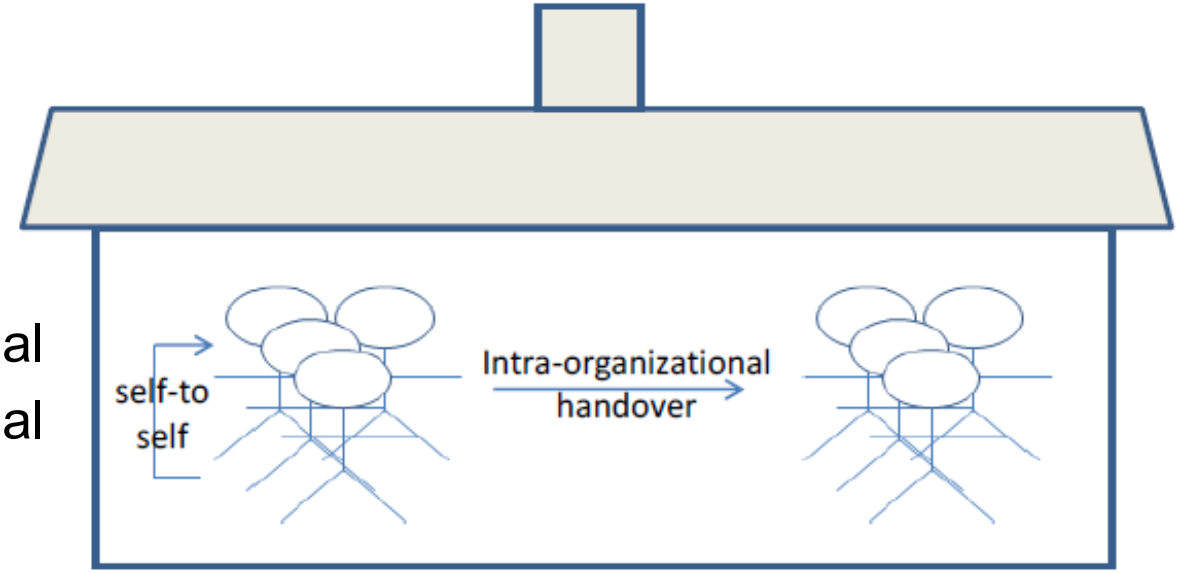

Maintainability Management
Management & Administration
Training
Handover Process
Maintenance Environment
Deployment
Version and Configuration Management
Documentation

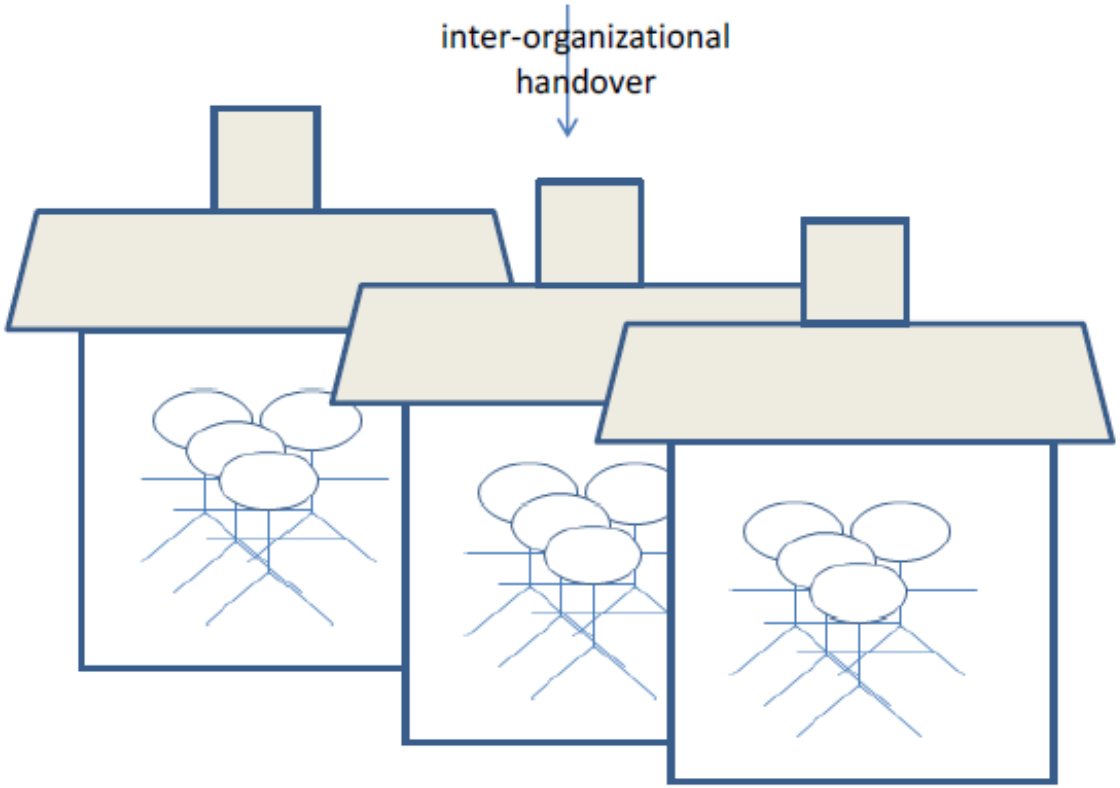
EM³: Handover Framework



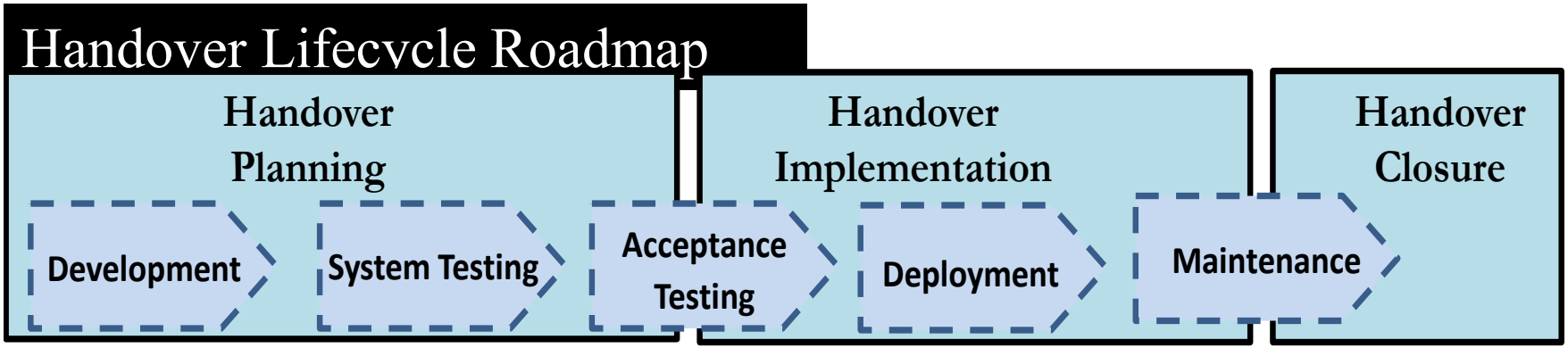
Handover types

- self-to-self
- intra-organizational
- inter-organizational

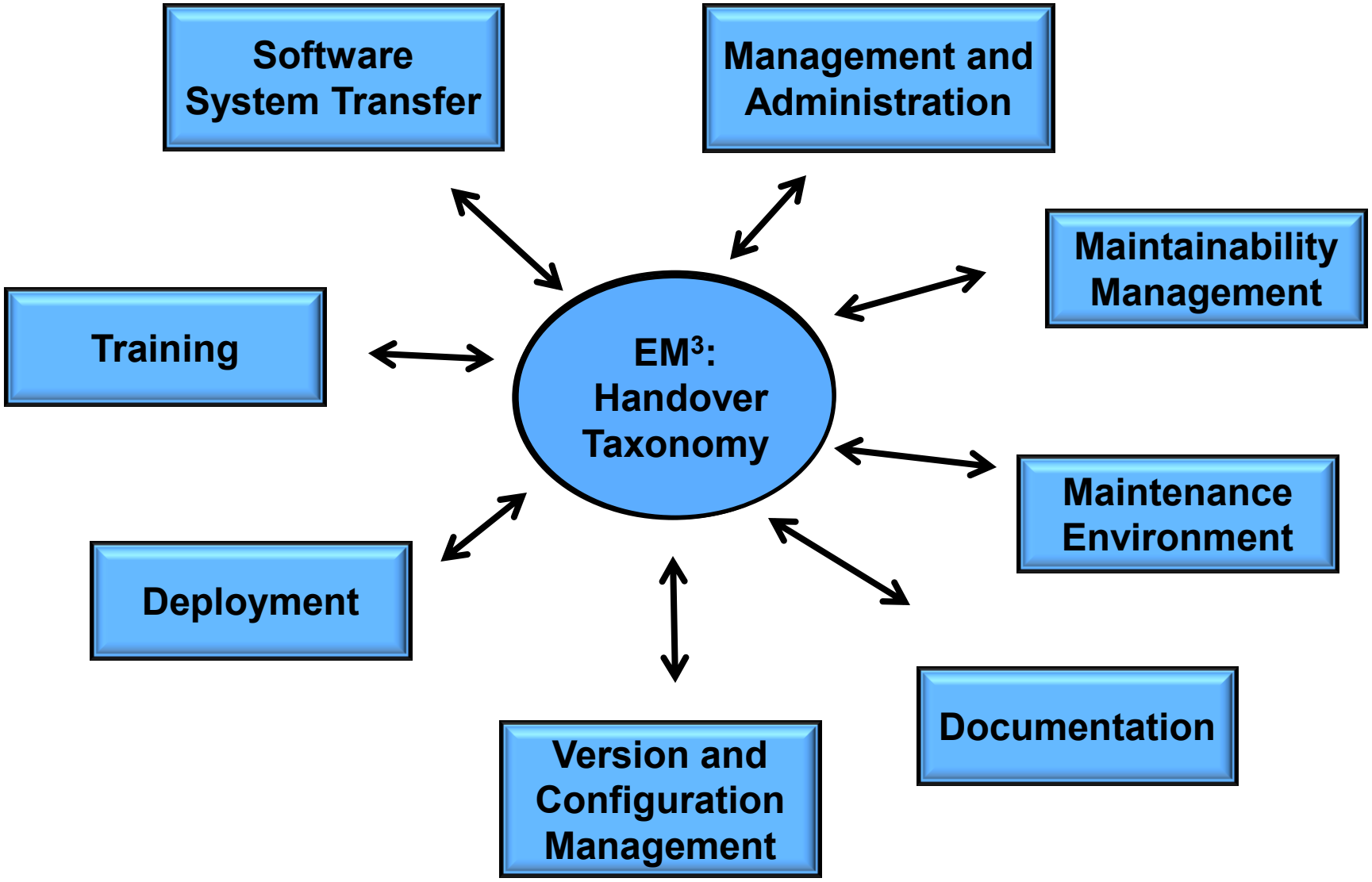




EM³: Handover Framework



EM³: Handover Practices



Software Handover

Documentation

D 1. Establish a system documentation repository.
D2. Define services to be provided by the system documentation repository.
D 3. Subject system documentation repository to SCM.
D 4. Establish standards for documentation development.

Deployment

DP 1. Develop installation procedures.
DP 2. Install.
DP 3. Plan future releases.
DP 3. 1. Plan updates of future releases.
DP 3.2. Determine the distribution structure.
DP 3.3. Determine forms of deploying software.
DP 3.4. Determine the structure of release notes.

Maintenance Environment

ME 1. Determine hardware/software suite needs.
ME 2. Install hardware/software suite.
ME 3. Assess current hardware/software suite, if any.
ME 4. Remedy the deficiencies in the hardware/software suite, if any.
ME 5. Determine/assess maintenance support suite.
ME 6. Supplement maintenance support suite with new tools.
ME 7. Install support software.
ME 8. Install software baseline.
ME 9. Install data.
ME 10. Transfer modification requests from development to maintenance.
ME 11. Place modification requests in a *Modification Request* repository.

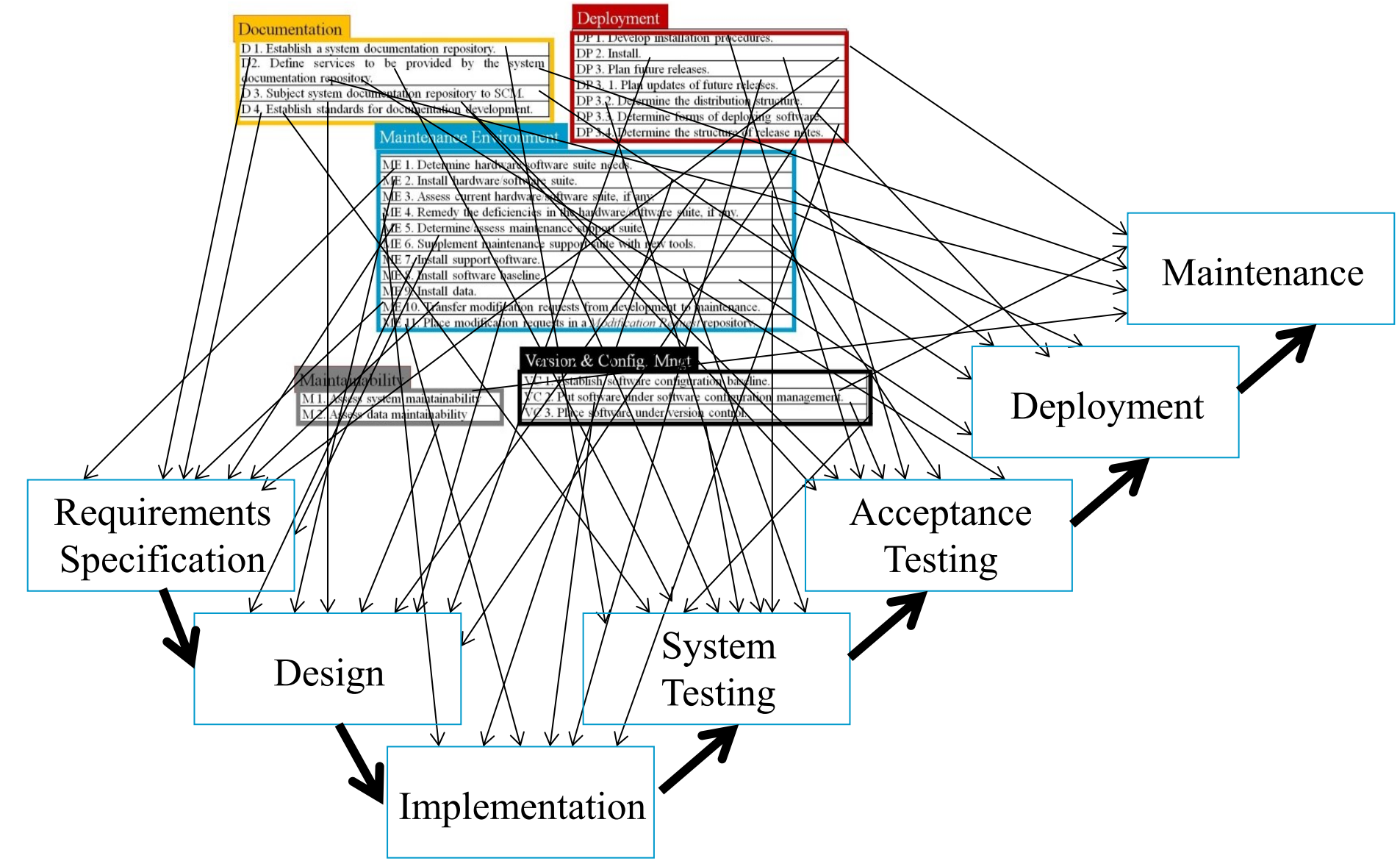
Maintainability

M 1. Assess system maintainability
M 2. Assess data maintainability

Version & Config. Mngt

VC 1. Establish software configuration baseline.
VC 2. Put software under software configuration management.
VC 3. Place software under version control.

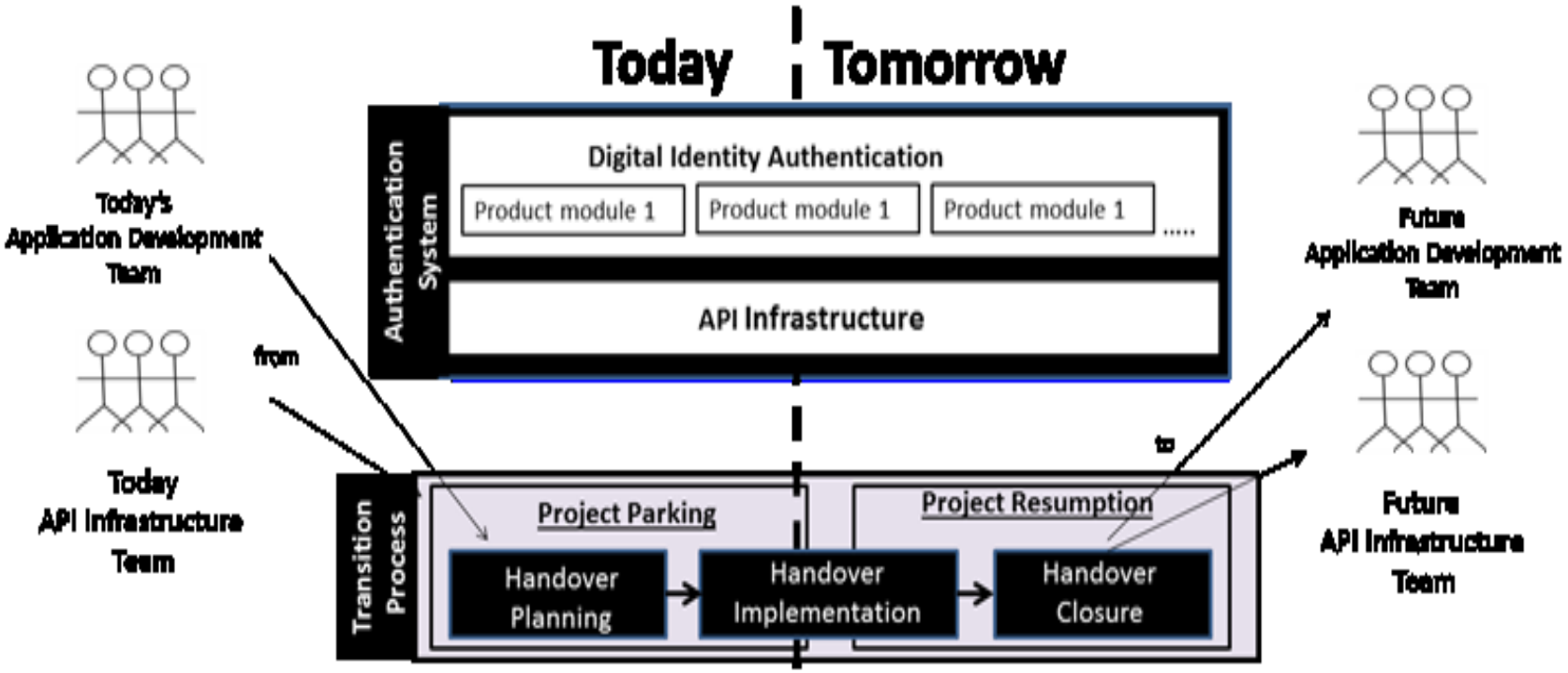
Software Handover



Future contexts

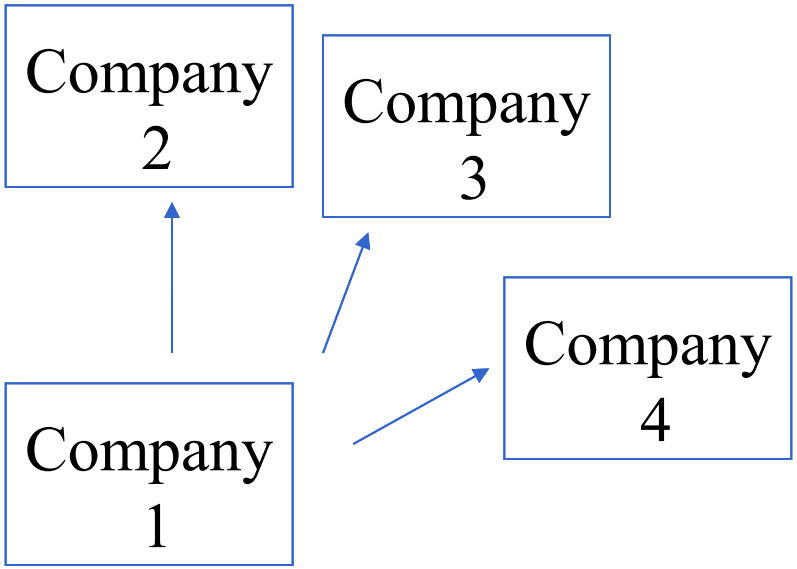
- Project parking
- Macro outsourcing
- Micro outsourcing

Project Parking



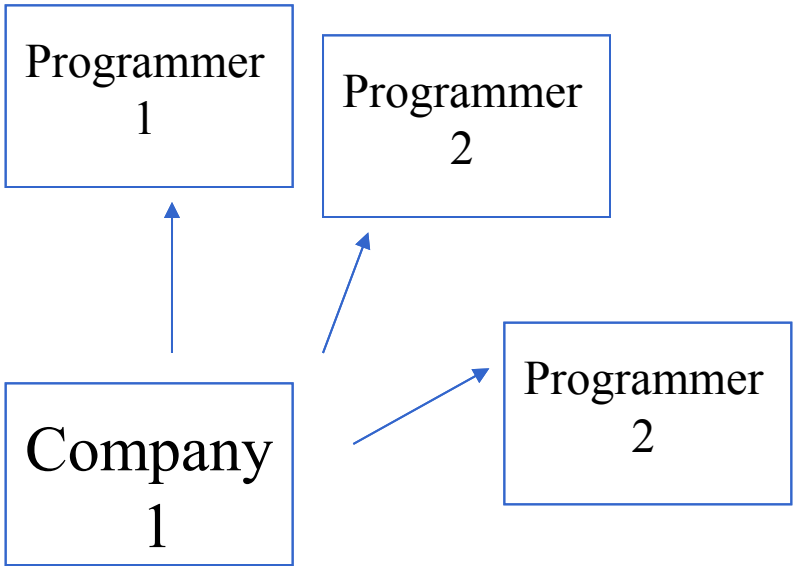
- Self-to-self? No?
- Self to someone unknown

Globalization needs thorough understanding of processes in traditional environments!



Transferring responsibilities to other companies requires a good transition process

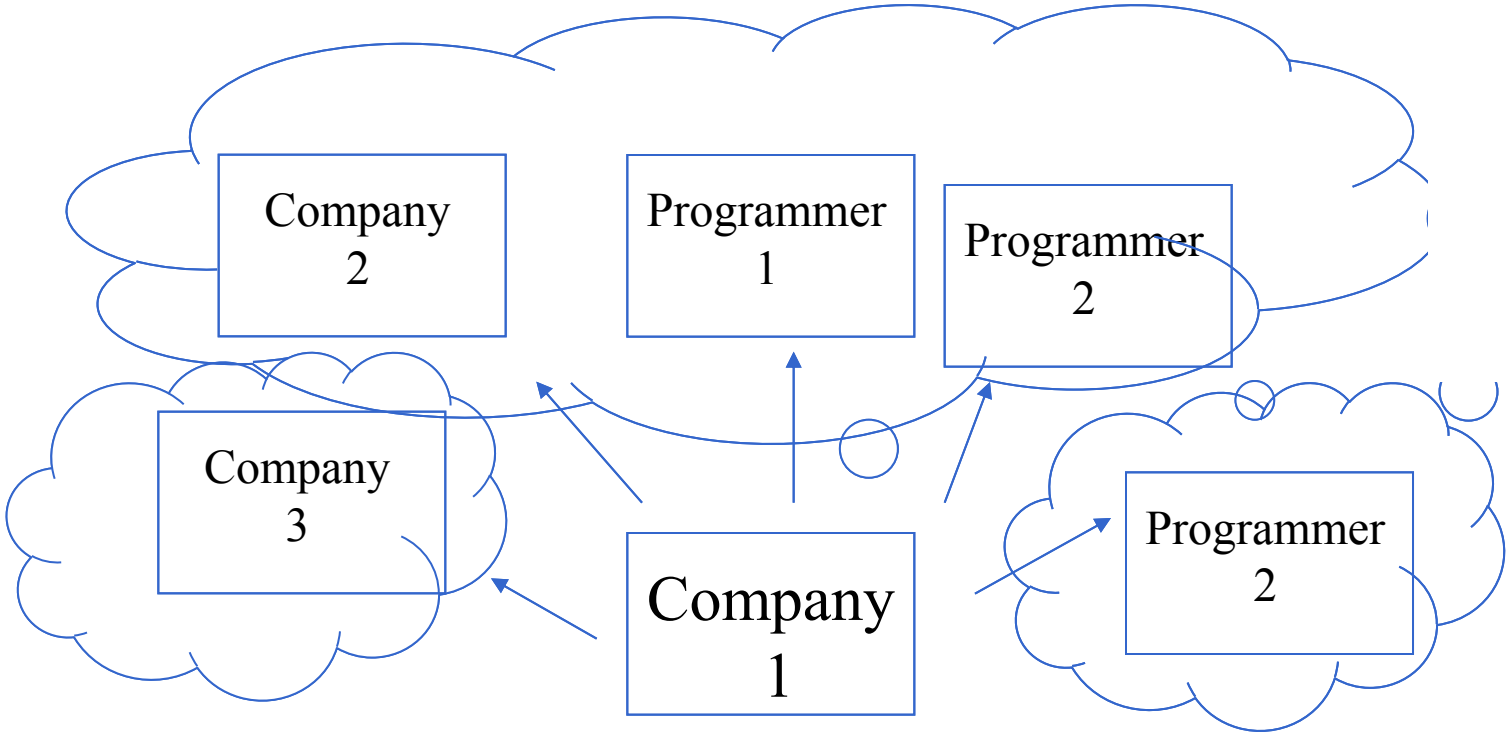
Micro cloud outsourcing



Your colleagues are not known in advance!

You only pay them for the job done!

Macro cloud outsourcing



Traditional outsourcing is not enough!
Companies need access to 100 professionals now and not in one month!

Research Uniqueness

- Presently, very little research is being done on software handover.
- The main sources available
  - Thomas Pigoski's publications
  - Vollman's publications
  - ISO/IEC 14764 standard
  - ITIL – Information Technology Infrastructure Library
  - EM^3: Handover Framework

We act like Bull Ferdinand. We sit under the trea and still smell our flowers

Research questions

- How should we define vision, strategy and tactical plans required for embarking on *Cloud Outsourcing?*
- How should we assess the outsourcing company's readiness to embark on *Cloud Outsourcing?*
- How should the organizations prepare themselves for *Cloud Outsourcing*?
- How should we assess the suitability of *eFrelancers* to do their work?
- How should we evaluate whether the outsourced tasks are suitable for cloud-mediated work?
- How should we monitor and control the outsourced *Cloud* tasks or projects?
- How should we manage distributed *Cloud* projects?
- How should we effectively communicate *Cloud* requirements to the *eFreelancers*? The success of the outsourced tasks strongly depends on how well the requirements are communicated.
- How should we evaluate the quality of services as provided by the *eFreelancers?*
- How should we provide feedback to the *eFreelancers* on their jobs?

Research questions

- What algorithms should we use for compensating the *eFreelancers*?
- What *"go and no-go"* decision criteria should we use when choosing *Macro Cloud* projects?
- How should we perceive cultural fit over the Internet and how should we deal with cultural differences?
- How should we define contingency and/or action plans to deal with situations when *eFreelancing* companies go out of business?
- What does *Cloud* lifecycle software process model look like?
- How should we adapt current risk management models for managing *Cloud Outsourcing* risks?
- How do we agree upon a common language so that we do not misunderstand each other?
- Do we need guidelines aiding us to focus on long-term or short-term partnerships in the clouds?
- What do the cloud contracts look like?
- How do we define cloud SLAs?
- How do we manage cloud emergency and crisis situations?
- How should we deal with the intellectual property rights, security and confidentiality aspects?
- How do we attract *eFreelancers* to work overtime in case of time pressure and tight schedules?