MCI MANAGEMENT CENTER INNSBRUCK

SYSTEMS PLANNING

System Modeling

Class content

MCI MANAGEMENT CENTER INNSBRUCK

What to expect from this class?

- An **introduction to system modeling** and its purpose
- A definition of **context models** and their goals
- **Use case** and **sequence diagrams**
- A distinction between **structural models and behavioral models**
- **Activity diagrams** and **state diagrams**
- A definition of **model-driven engineering**

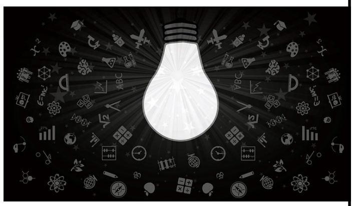
Source: https://pixabay.com/en/checklist-check-list-marker-2077020/ by TeroVesalainen |CC0 Creative Commons

Class learning goals

MCI® MANAGEMENT CENTER INNSBRUCK

What should be your learning outcome?

- Understand the **purpose of system modeling**
- Understand the **difference** between a **static and a dynamic modeling** perspective
- Know a number of **diagram types** and how to create them
- Understand the **idea of model-driven engineering**



Source: https://pixabay.com/en/learning-hint-school-subject-3245792/ by harishs | CC0 Creative Commons

System Modeling

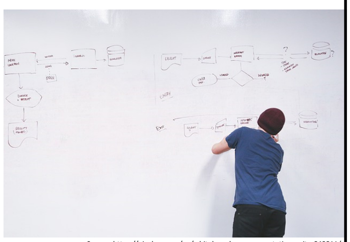
MCI® MANAGEMENT CENTER INNSBRUCK

What is system modeling?

- Developing **abstract models of a system**
- Different **views and perspectives**
- Uses a graphical notation
  Usually: **Unified Modeling Language (UML)**

Models are used:
- During the **requirements engineering** process
- During the **design** process
- And for **documentation** purposes

Note: The most important aspect of a system model is that it leaves out detail!

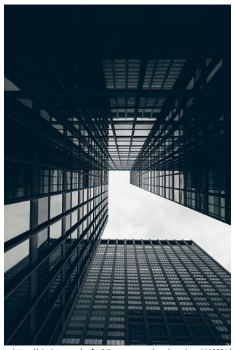


Source: https://pixabay.com/en/whiteboard-man-presentation-write-849811/ by StartupStockPhotos | CC0 Creative Commons

System Modeling

Different models for different perspectives

- **External perspective**
  Models the context or environment of the system
- **Interaction perspective**
  Models the interactions between a system and its environment or between the components of a system
- **Structural perspective**
  Models the organization of a system or the data that is processed by the system
- **Behavioral perspective**
  Models dynamic behavior of the system and how it responds to events

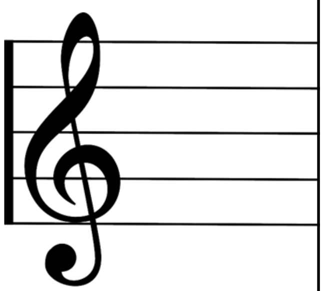


Source: https://pixabay.com/en/buildings-perspective-city-urban-1149851/
by Free-Photos |CC0 Creative Commons

System Modeling

Use of graphical notations

- **Facilitate discussion**
  Models may be incomplete. Goal: Stimulate discussion amongst software engineers
- **Document existing system**
  Do not need to be complete but need to be correct and accurate!
- **Detailed system description for implementation**
  Used for model-based development, hence models need to be complete and correct. Here you need to be careful with the notation.



Source: https://pixabay.com/en/treble-clef-staff-notation-plain-304441/
by Clker-Free-Vector-Images |CC0 Creative Commons

Initial steps

- First one needs to decide on the **system boundaries**
- Decide **what functionality should be included** e.g. should a patient information system focus exclusively on collecting information about consultations or should it also collect personal patient information. **What are the system boundaries?**
- The definition of **system boundaries** may be defined by **non-technical factors** e.g. the people you talk to, the resources that can/must be spent and the analysis, etc.
- Once the boundaries are clear, the next step is to **define the context using a context model**

Source: https://pixabay.com/en/country-road-road-traffic-tar-599457/ by markusspiske | CC0 Creative Commons

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

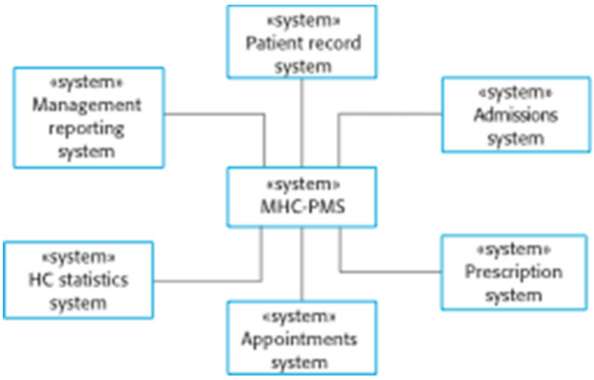
Context Models

- Show the **environment**
- **Do NOT show** the types of **relationships** between the systems
- Systems might **produce data, consume data, or share data**
- Systems might be **connected directly** or **not connected at all**
- Systems may be physically **co-located** or placed in **separate buildings**

Source: https://pixabay.com/en/ideas-new-creative-innovation-man-3064921/ by mohamed_hassan | CC0 Creative Commons

System Modeling

Context Model: Example Patient Information System

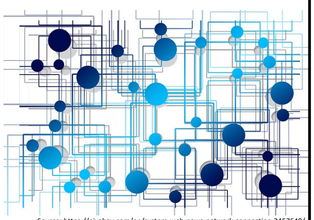


Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

Interaction Models

- All systems involve interaction
- Ranges from **user interactions to interactions between system components**
- Helps **highlighting potential communication problems**
- **Two common approaches**
  1) Use Case modeling to model interactions between a system and external actors;
  2) Sequence diagrams model interactions between system components (and actors)



Source: https://pixabay.com/en/system-web-news-network-connection-2457648/ by geralt |CC0 Creative Commons

Use Case Modeling

- Developed by Jacobson et al., 1993
- Incorporated into the **first release of UML**
- **Simple scenario** that describes what a user expects from a system
- Represents a **discrete task** that involves external interaction with a system
- Use case is shown as an ellipse
- Actors are represented as **stick figures**
- **Fairly simple overview** of the interaction

Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

Use Case Modeling

- Add more detail to understand what is involved
- **Simple textual description, structured description in a table, or sequence diagram**

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the MHC-PMS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

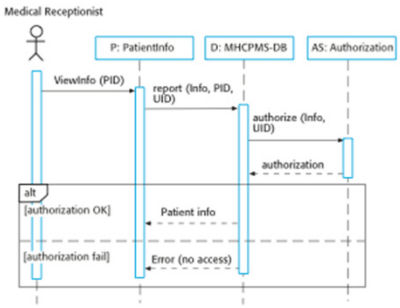
Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

MCI® MANAGEMENT CENTER INNSBRUCK

Sequence Diagrams

- Model **interactions between actors and objects**
- UML offers a **rich syntax** for sequence diagrams
- Shows a **sequence of interactions during a particular use case**



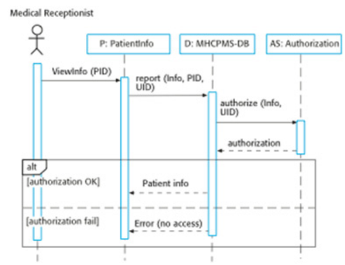
Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

MCI® MANAGEMENT CENTER INNSBRUCK

Sequence Diagrams

- **Actors** involved are listed **along the top** of the diagram
- **Interactions** are indicated by **annotated arrows**
- The **rectangle** on the dotted line indicates the **lifeline of the object concerned**
- The sequence is read from **top to bottom**
- **Arrows** indicate **calls to the objects, their parameters** and the **return values**

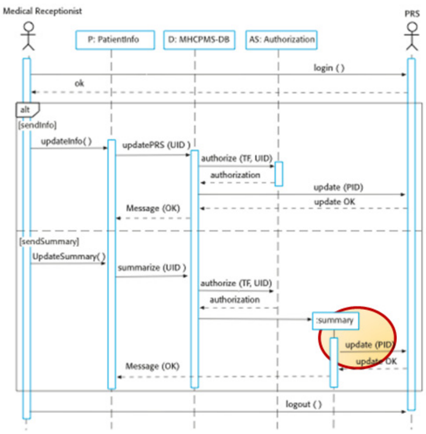


Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

MANAGEMENT CENTER INNSBRUCK

Sequence Diagrams



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

- **Two additional features**
  1) Direct **communication between** the **actors**
  2) Creation of **objects** as part of the sequence

- In sequence diagrams one can **focus on the important/relevant interactions**
- Not every interaction needs to be included!

System Modeling

MANAGEMENT CENTER INNSBRUCK

Structural Models

Display the **organization of a system**

- **Static Models**
  Show the structure of the system design

- **Dynamic Models**
  Show the organization of the system when it is executed

Source: https://pixabay.com/en/blue-cube-design-modern-cube-shape-2137334/ by PIRO4D | CC0 Creative Commons

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

Class Diagrams

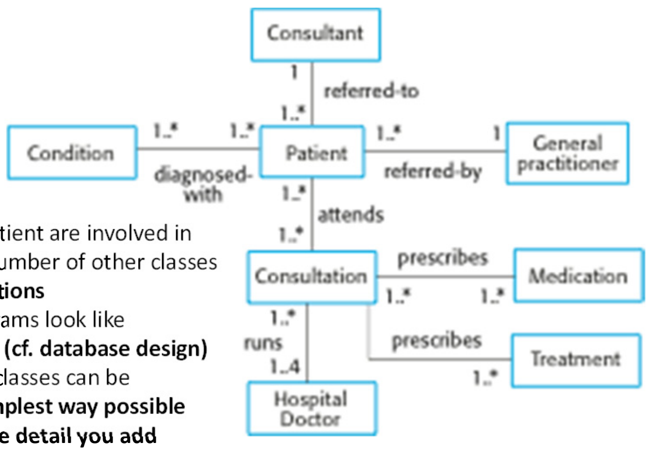
- Used when developing **object-oriented systems**
- Shows the classes in a **system and their associations**
- Early on in the development process, classes represent **real-world objects**
- Later you will need additional **implementation classes**
- Class diagrams can be expressed at different **levels of detail**
- The simplest way **writes class names in boxes** and **associates them through lines**

Source: https://pixabay.com/en/school-back-to-school-schulbeginn-1665537/ by Gellinger | CC0 Creative Commons

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

Class Diagrams



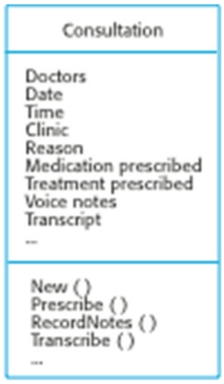
- Objects of the class Patient are involved in relationships with a number of other classes
- **You can name associations**
- At this level class diagrams look like **semantic data models (cf. database design)**
- To **show associations** classes can be **represented in the simplest way possible**
- To define them in **more detail you add attributes and operations**

Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

Class Diagrams

- **The name of the object** class is in the top section
- **The class attributes** are in the middle section
- **The operations** (called methods in Java) are in the lower section

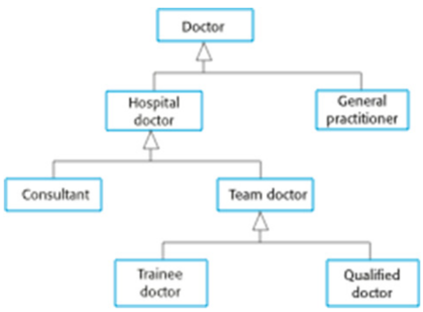
E.g. doctors record **Voice notes** that are transcribed later **(Transcribe method)**. To prescribe medication, the doctor involved must use the **Prescribe** method to generate an electronic **Prescription**.



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

Generalization

- Everyday technique that we use **to manage complexity**
- We put **entities in more general classes** (e.g. animals, cars, houses, etc.)
- Classes have **common characteristics**
- When modeling systems we **check whether** there is **scope for generalization**
- This is **good design practice** as it **reduces the number of classes that need to be changed if change occurs**
- **UML** has a specific type of **association to denote generalization**



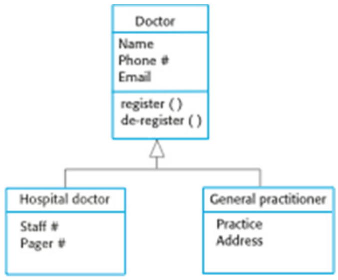
Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

Generalization

- The generalization is shown as an **arrow head** pointing up to the more general class
- Attributes and operations associated with higher-level classes are also **associated with the lower-level classes**
- Lower-level classes **add more specific attributes and operations**



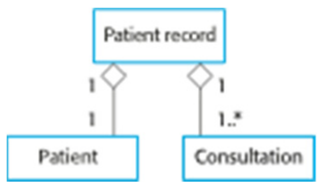
Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

Aggregation

- Objects in the real world are often **composed of different parts**
- This is illustrated by a **type of association called aggregation**
- One object **(the whole) is composed of** other objects **(the parts)**
- We show this by putting a **diamond shape** next to the class that represents the whole



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

Behavioral Models

- Models of the **dynamic behavior of the system** as it is executing
- They show what is happening when a system **responds to a stimulus**

- **Two types of stimuli**
  Data - data arrives and is processed
  Events - some event triggers system processing

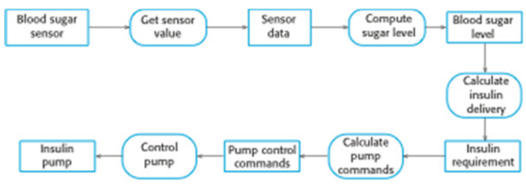
Source: https://pixabay.com/en/security-insecurity-vulnerability-856168/ by johnhain | CC0 Creative Commons

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

Data-driven modeling

- Data-driven models show the **sequence of actions involved** in processing input data and generating an associated output
- They can show **the entire sequence of actions**
- They were among the **first graphical software models**
  e.g. DeMarco, 1978: data-flow diagrams (DFD)
- DFDs show how **data moves through the system**
- **UML 2.0** introduced a similar concept **i.e. activity diagrams**



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

MCI® MANAGEMENT CENTER INNSBRUCK

Activity Diagrams

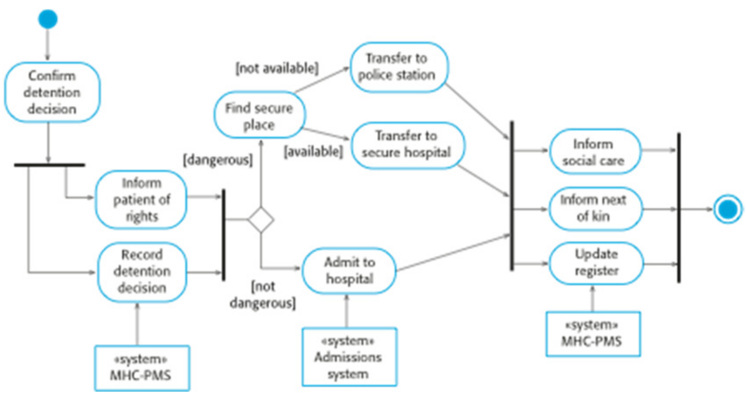
- Show the **activities that make up a system process**
- **Start** of a process is indicated by a **filled circle**
- **End** of a process is indicated by a **filled circle inside another circle**
- **Rectangles** with round corners represent **activities**
- **(External) objects** use the UML **stereotype feature**
- **Arrows** represent the **flow of work**
- **Solid bar** is used to indicate activity **coordination**

Source: https://pixabay.com/en/tricycle-red-childhood-toy-fun-691587/ by Free-Photos |CC0 Creative Commons

System Modeling

MCI® MANAGEMENT CENTER INNSBRUCK

Activity Diagram: Example Patient Information System



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

Event-driven modeling

- Event-driven models show how a system responds to **external and internal events**
- **Assumptions**
  1) a system has a **finite number of states**
  2) stimuli causes **transition from one state to the other**

- UML offers **state diagrams**
- State diagrams show the **system states and events that cause transition**
- State diagrams **do NOT show the flow of data**

Source: https://pixabay.com/en/start-ready-competition-track-1414148/
by MemoryCatcher |CC0 Creative Commons

System Modeling

MCI MANAGEMENT CENTER INNSBRUCK

Event-driven modeling: Example Microwave

Handles
- Switch to select full or half power
- Numeric keypad
- Start/stop button
- Alphanumeric display

Actions
- Select the power level
- Input cooking time
- Press start

Source: https://pixabay.com/en/microwave-oven-appliance-kitchen-29056/
by Clker-Free-Vector-Images |CC0 Creative Commons

Constraints
- Oven should not operate when the door is open
- On completion a buzzer is sounded

System Modeling

MANAGEMENT CENTER INNSBRUCK

Event-driven modeling: Example Microwave

- States are represented by **round rectangles**
- **Arrows represent stimuli** that force transition
- **Start and end** states use **filled circles**

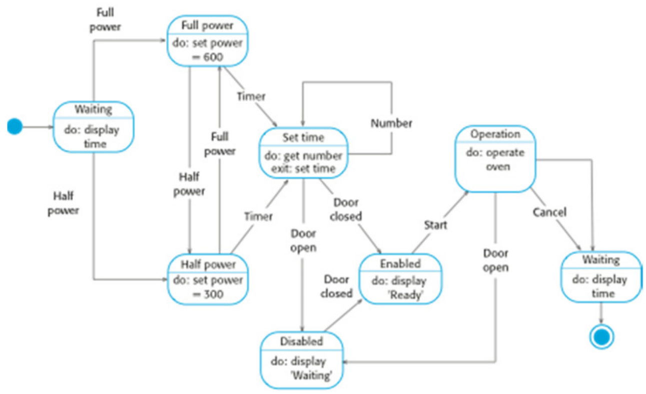
- For system specification **more detail is necessary**
- For this you can use a **tabular description**
- **Problem: Number** of states **increases rapidly**
- Use '**superstates'** to encapsulate a number of separate states

Source: https://pixabay.com/en/appliances-refrigerator-microwave-993782/ by hannahlouise123 |CC0 Creative Commons

System Modeling

MANAGEMENT CENTER INNSBRUCK

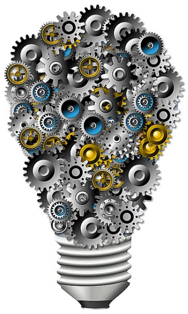
Event-driven modeling: Example Microwave



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

Model-driven Engineering

- **Models** rather than programs are the output of the **development process**
- **Programs** are then **generated automatically** from the models
- It has been used since 2001 but is **still at an early stage**
- You need to construct graphical models whose **semantics are well defined**
- This is possible through executable UML (**xUM**L) **cf. Mellor & Balcer, 2002**



Source: https://pixabay.com/en/gears-bulb-innovation-technology-1443730/
by Maklay62 |CC0 Creative Commons

Model-driven Engineering: Pro

- Allows engineers to **think about systems at a high level of abstraction**
- **Reduces** likelihood of **errors**
- **Speeds up** design and implementation process
- Creation of **reusable, platform independent application models**
- Implementations can be generated for different platforms from the same model (i.e. it is only necessary to write a '**translator'** for that platform)

Source: https://pixabay.com/en/pro-approve-affirm-thumbs-up-green-1402154/
by johnhain |CC0 Creative Commons

System Modeling

Model-driven Engineering: Cons

- Models are a good facilitator for discussions. However, the **implementation may not follow the abstraction models provide**
- **Platform independence is only important for large long-lifetime systems** where other platforms may become obsolete during a system's lifetime. However, for these classes of systems the **implementation is not the major problem. They rather require integration and testing!**

Source: https://pixabay.com/en/pro-approve-affirm-thumbs-up-green-1402154/ by johnhain | CC0 Creative Commons

Summary

What you should have taken away from this class:

- A **model is an abstract view of a system** that ignores some system details. Complementary system models can be developed to show the **system's context, interactions, structure, and behavior.**
- **Context models** show how a system that is being modeled is **positioned in an environment with other systems and processes**.
- **Use case diagrams** and **sequence diagrams** are used to describe the **interactions between the system being designed and users/other systems**.
- **Structural models** show **the organization and architecture of a system**. **Class diagrams** are used to define the **static structure of classes** in a system and **their associations.**
- **Behavioral models** are used to describe the **dynamic behavior of an executing system**. This can be modeled from the **perspective of the data** processed by the system **or by the events** that stimulate responses from a system.
- **Activity diagrams** may be used to model the **processing of data**, where each activity represents one process step.
- **State diagrams** are used to model a **system's behavior in response to internal or external events**.
- **Model-driven engineering** is an approach to software development in which a system is represented as a set of models that can be **automatically transformed to executable code**.

References

- Ambler, S. W., & Jeffries, R. (2002). *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. New York, NY, USA: John Wiley & Sons.
- Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide*. Boston, MA, USA: Addison-Wesley.
- DeMarco, T. (1978). *Structured Analysis and System Specification*. New York, NY, USA: Yourdon Press.
- Jacobson, I., Christerson, M., Jonsson, P., & Overgaard, G. (1993). *Object-Oriented Software Engineering*. Wokingham, UK: Addison-Wesley.
- Kent, S. (2002). Model-driven Engineering. *Proceedings of International Conference on Integrated Formal Methods*, (pp. 286-98).
- Mellor, S. J., & Balcer, M. J. (2002). *Executable UML*. Boston, MA, USA: Addison-Wesley.
- Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The Unified Modeling Language Reference*. Reading, MA, USA: Addison-Wesley.
- Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *The Unified Modeling Language Reference Manual 2nd Ed*. Boston, MA, USA: Addison-Wesley.
- Schmidt, D. (2006). Model Driven Engineering. *IEEE Computer 39(2)*, 25-31.

Thank you for your attention!

mentoring the motivated.