Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 79 (2019) 626–631

www.elsevier.com/locate/procedia

12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy

Concept of a learning knowledge-based system for programming industrial robots

Alejandro Magaña Flores[a,]*, Philipp Bauer[a], Gunther Reinhart[a]

[a]*Institute for Machine Tools and Industrial Management, Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany*

* Corresponding author. Tel.: +49-89-289-15468; fax: +49-89-289-15555. *E-mail address:* alejandro.magana@iwb.mw.tum.de

Abstract

A major challenge for the use of industrial robots has been its programming, which requires expert knowledge. Offline programming tools, based on simulation models, are normally used to facilitate the robot programming. However, in many industrial applications the intervention of an operator is still necessary to correct the robot programs. This paper presents a concept based on a knowledge-based system (KBS) that integrates the capability to learn from manual adjustments conducted by an operator. Based on the learned data the KBS will be able to imitate the adjustments of the operator, enabling a full automation of the robot programming.

Peer-review under responsibility of the scientific committee of the 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering.

Keywords: Industrial robot programming; Knowledge-based system, Learning system, Knowledge processing, Solution space model, Robot pose model

1. Introduction

The trend towards mass customized products requires a high level of flexibility and automation along all processes in the production chain. These requirements are one of the biggest challenges that the developers of manufacturing equipment are being confronted by. The inherent flexibility and automation possibilities of robotic systems seem to be good qualities to defy these challenges. The use of industrial robots has been successfully demonstrated in several production processes. Even SMEs have already started to automatize some of their production processes using industrial robots. This trend can also be seen in current forecasts for the worldwide industrial robot supply, which predicts a global increase of almost 80% percent by 2020 compared to 2016 [1].

The use of robotic systems promises to bring a major flexibility in the automation, a quality of such systems that has not yet been properly exploited. The classical applications for industrial robots are repetitive production processes, where they work in a known environment and the programmed path never changes [2,3]. This contradicts the inherent flexibility of robotic systems. Furthermore, it reflects the impediment for their use in high flexible production processes due to the associated programming efforts. This can be confirmed by looking at the total costs of a robotic system that show one third being designated for its programming [4].

Industrial robots are commonly programmed using the following approaches: online and offline programming. Online methods are particularly suitable for the creation of simple programs or to correct programs. In this case, the operator moves the robot manually to the working pose, normally using a teach-in device. For this reason online programming is associated with a high expenditure of time for teaching-in the robot poses. To reduce the teach-in time other approaches have been developed e.g. the use of augmented reality [5]. However, the challenge of using operator knowledge remains unaddressed since the manual adjustment are not being modelled, making it difficult to use the applied knowledge in similar problems. In the field of humanoid robotics the use of programming by demonstration techniques, where an operator physically leads the robot, teaching it complete trajectories was demonstrated by [6]. Furthermore other approaches have been investigated for programming robots using different input and information and devices [7,8]. Nevertheless, most of the approaches

2212-8271
Peer-review under responsibility of the scientific committee of the 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering.
10.1016/j.procir.2019.02.076

concentrate on the modelling of the robot trajectories rather than on the modelling of the robot poses, which is more relevant in some industrial applications.

On the other hand, offline programming approaches using simulation models have proven to be more suitable for generating more complex robot programs. However, the simulation requires an extensive model of the robotic system and the production process. The norm DIN EN ISO 10218-2 describes an industrial robotic system by its elementary components: an industrial robot, an end effector and any other objects which form part of the robot cell [9]. In order to model such complex systems some works have opted to use knowledge-based systems (KBS). One of major advantages of knowledge-based systems is the separation of the modelling process into single expert domains (robotic system, production process, environment), allowing a high modularity of the whole system model [10]. Some works have demonstrated the high flexibility of such systems and developed knowledge-based frameworks for programming robots to autonomously execute pick and place tasks [11,12]. Further works in the field of industrial robotic systems have as well proven to be successful for programming welding tasks with knowledge-based models [13,14].

Robot programming can be time consuming and requires expert knowledge of the robot and the production process. Some of the revised approaches have already dealt with this problem to ease the programming effort of robotic systems by using simulation models to automatically generate robot programs, composed of robot poses and paths. However, the validity of the calculated robot poses and paths is directly affected by the quality of the simulation models [3]. The implemented models may fail to calculate valid robot poses and paths, particularly in production processes, where some effects influencing the process have not been entirely investigated and modelled. In these cases, the inaccuracy of the model must be manually corrected by the operator for the affected robot poses and paths. Since the used models for the simulation cannot be updated, the operator may be applying similar changes to all affected robot poses and paths. Depending on the production process and robotic system, this task may result being very time exhaustive. To address this problem, our work introduces a concept that combines the explicit modelling of the system with the integration of implicit knowledge from the operator. The gathered knowledge is then used for updating the system model. Our proposed approach bases on a learning knowledge-based system (KBS), which comprises three fundamentals steps: integrating, modelling and processing of knowledge.

2. Concept overview

Based on the revised approaches for the programming of industrial robots, our work proposes a learning KBS that additionally extends the system modelling to the integration of new information provided by the operator. The diagram, shown in Fig. 1, summarizes the workflow of the system and gives an overview of the main modules.

The first module deals with the collection of external information. Thus, the primary objective of the knowledge acquisition module (KAM) is to collect every possible information of the system as well as the feedback of the operator. The KAM operates as an information interface between the real system and the KBS. In the next step the gathered information of the KAM is integrated into the knowledge modelling module (KMM). The entire data model of the whole system builds the knowledge base (KB). The KB works just as a model and storage of the gathered information. The utilization of the information is handled by the knowledge processing module (KPM). The KPM uses the explicit model of the system and learning approaches to derive adjustments for the system.

3. Knowledge acquisition module

The structure of the KAM can be derived by looking at the workflow for the correction of simulated robot programs. The workflow can be mainly described by two steps: the operator identifies a problem (caused by an error/incomplete of the system model) and the operator corrects it by adjusting the poses and paths of the robot program and other system parameters. Thus, the knowledge acquisition module (KAM) is responsible for collecting any kind of information related with the cause of the problem and the derived adjustments to the system. To fulfill this task, the KAM must be capable to collect information coming from the operator (implicit knowledge) and registering the adjustments applied to the system (explicit information). The structure of the KAM is shown in Fig. 2.

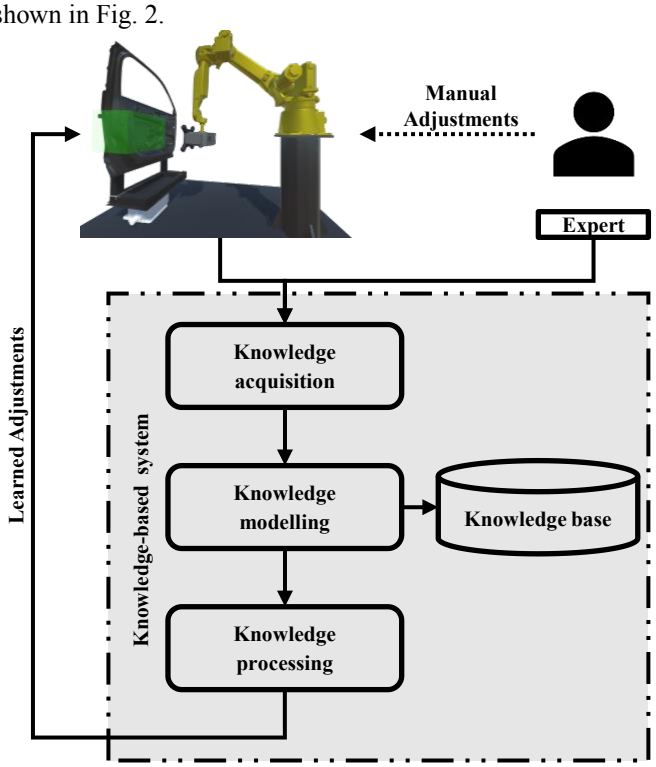

Fig. 1. Architecture of the knowledge-based system with integration of expert knowledge.

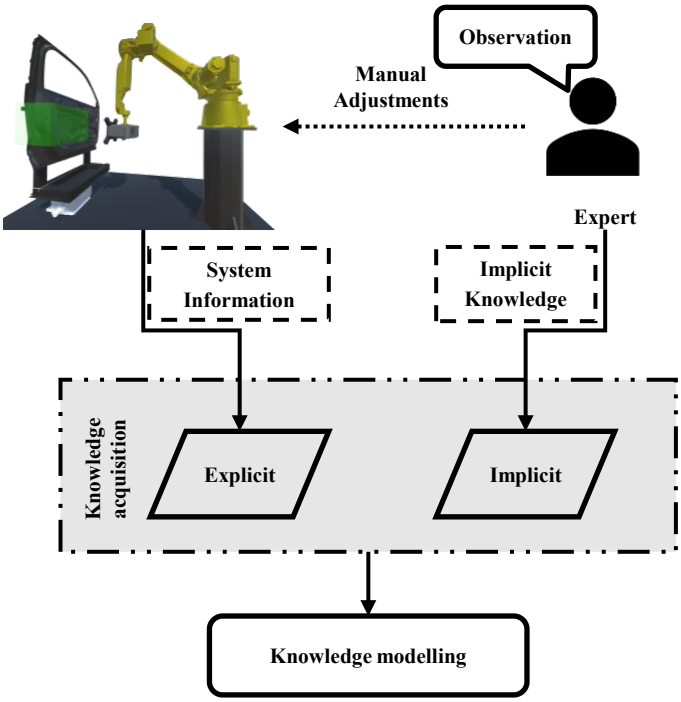

Fig. 2. Architecture model of the knowledge acquisition module.

3.1. Acquisition of implicit knowledge

This submodule deals with the acquisition of implicit knowledge provided by the operator. Therefore, this module is responsible for converting implicit knowledge (i.e., the operator input/feedback describing a situation) into explicit information (machine-readable form) that can be processed by the KBS. In the most primitive form this can be achieved by the operator filling up an established protocol. To facilitate the information collection and to not disturb the work of the operator, technologies for gesture or voice recognition can be hereby used to acquire the necessary feedback.

3.2. Acquisition of explicit information

The second submodule is responsible for registering the adjustments applied to the robotic system. These adjustments are usually more accessible and can be directly associated with explicit information, e.g. joint angles or velocities of the robot, process parameters, inter alia.

4. Knowledge modelling module

The KMM comprises the explicit modelling of the robotic system. As a first step, the KMM is responsible for the processing of the gathered information by the KAM and for its integration in the KB. Additionally, the KMM provides the local models of the poses and paths to the KPM for their further interpretation. An architecture overview of the KMM is shown in Fig. 3.

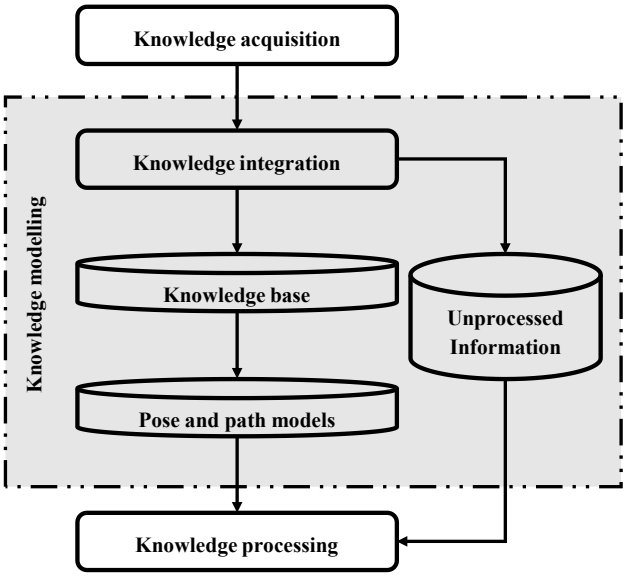

Fig. 3. Architecture of the knowledge modelling module.

4.1. Knowledge integration

The interpretation of the acquired information provided by the KAM is processed using rules such as description logics (DL) or first-order logic (FOL) commonly used for classification and interpretation of information. This module enables the acquired data to be explicitly related to the KB. To guarantee its integration in the KB, the rules have to previously established by the whole personnel involving the design, the development and usage of the system, e.g. robot and process experts as well as operators. Logic rules may not always suffice to interpret and classify all the gathered information. Thus, this module must preserve this information and pass it to the KPM for further processing.

4.2. Knowledge base

To achieve a high modular and flexible modelling of the robotic system our work uses a KB. The KB is designed based on analytical models (mathematical, probabilistic model and functions), relation models (DL and FOL) and data models (CAD, system parameters) of the single components of a robotic systems. The KB is modelled using the semantic web ontology language (OWL), as introduced by [11,12]. Fig. 4 shows an overview of the semantic model for a robotic system including individual semantic models for an industrial robot, end-effector and workpiece.

Furthermore our concept extends the modelling of the production process and environment in comparison with some of the revised works. To assure that the applied adjustments to the system applied by the operator can be preserved our proposed KB focuses on the modelling of the poses and paths of the robot.

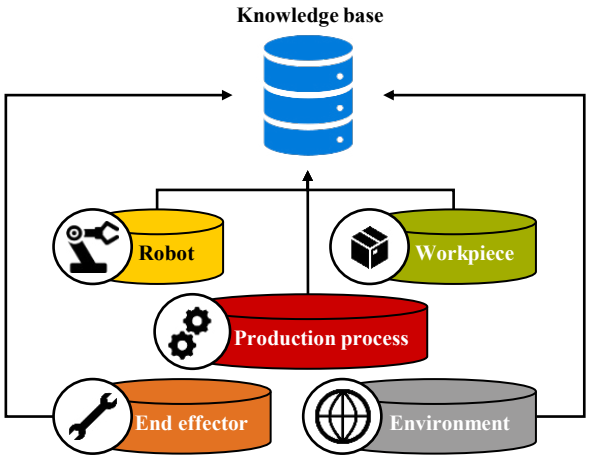

Fig. 4. Base architecture of the knowledge base for a robotic system.

4.3. Pose and path modelling

The robot program, defined by poses and paths generated during the offline-programming, normally has to be adjusted by the end user during the commissioning in the online programming phase. A robot pose p is normally given by the spatial description of the end effector (position and orientation) $p=[x \quad y \quad z \quad \alpha \quad \beta \quad \gamma]$ or by the robot joint angles. The path is described by dynamics parameters normally given by the velocity and acceleration of the end effector or joint angles.

The conventional models of poses and paths do not allow to integrate any other related information of the system. Therefore, an explicit and extensive model of each pose and path is needed to integrate locally changes made by the end user. By the introduction of the state k for $k \in \{0,\ldots,n-1\} \wedge k \in \square$ with n being the total number of poses, the robot program can be space discretized and each pose and path can be modelled individually. The discretization of the path is commonly used in path planning problems for automatically finding collision free paths [15]. The extension of these models for relating other information to the poses and paths were introduced in [16] for the calculation of optimal path planning using prediction models. Our work also proposes a state model and extends the pose modelling by a solution space model for each pose and path between two poses. The solution space for a pose p_k is given by a local set P_k and a function F_p for calculating a valid pose depending on a local model of the robotic system r_k.

$$P_k = \{p_k \mid p_k \in F_p(r_k)\} \tag{1}$$

A valid path u_k is defined between two poses p_{k-1} and p_k. Using a similar notation as for the pose, the solution space for a path is given by a set U_k and a function F_u which depends on the local robotic system model.

$$U_k = \{u_k \mid u_k \in F_u(r_k)\} \tag{2}$$

The discrete robotic system model is given by the function S depending on local models of the individual components: robot t_k, end effector e_k, workpiece w_k, production process o_k and environment z_k.

$$r_k = S(t_k, e_k, w_k, o_k, z_k) \tag{3}$$

The sets (1) and (2) can be broken into more detailed subsets separating the solution space model into valid and invalid space solution models. The valid space solution model is described by the function V. The counterpart, the invalid space solution space, is given by a function N. The complete local pose and path solution space models are given at the state k:

$$P_k = \{p_k \mid p_k \in V_{k,p}(r_k) \wedge p_k \notin N_{k,p}(r_k)\}, \tag{4}$$

$$U_k = \{u_k \mid u_k \in V_{u,k}(r_k) \wedge u_k \notin N_{u,k}(r_k)\}. \tag{5}$$

Furthermore, to track historical changes of the system model, each individual pose and path model is given a timestamp z: $p_{k,z}$ and $u_{k,z}$. The integration of a timestamp facilitates the data analysis to track the history of the solution space models.

With the introduction of individual solution space models for every pose and path, the complete robotic system can be extensively locally modelled. This model allows changes, applied by the end user, to be explicitly related to a specific component of the robotic system. Fig. 5 shows an illustrative example for the valid and invalid solution space model for a robot-based measurement system. In this case the solution space model gives all possible poses where the robot may be positioned to measure a feature in the door. The negative space solution model shows invalid tested poses that could have been corrected by the end user due to an inaccurate or incomplete modelling of influencing factors. The modelling refers to components used for the calculation of poses and path such as: light-reflection, CAD models, robot singularities, sensor parameters, inter alia.

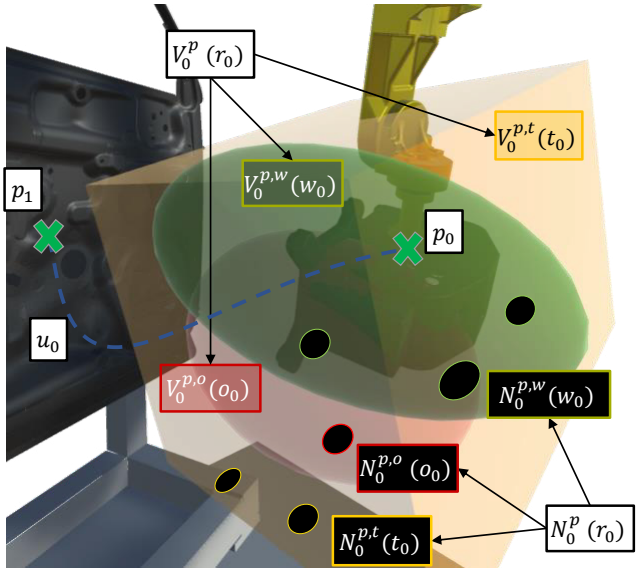

Fig. 5. Exemplary solution space model of the pose p_0.

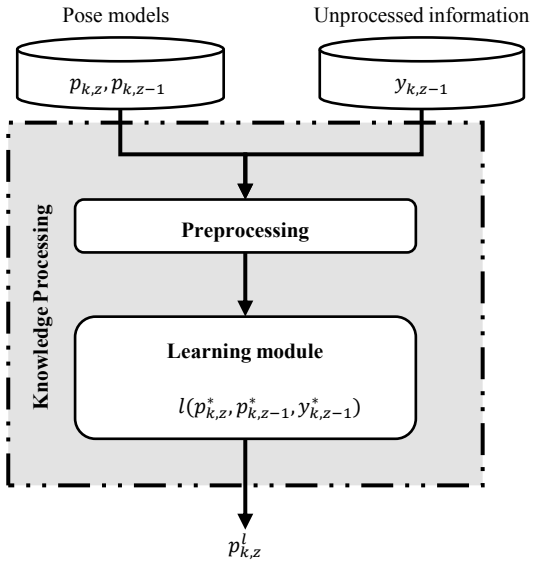

Fig. 6. Workflow of the knowledge processing module.

5. Knowledge processing module

This module is responsible for the processing of the knowledge. The KPM is subdivided into two submodules. The preprocessing module is responsible for filtering the information that will be provided to the learning module The second module, the learning module trains a model based on the given data with the goal to reproduce the same adjustments made to the system by the end user to correct a pose or path.

The architecture of this module is presented in Fig. 6 and illustrates exemplary the workflow for the correction of a pose p_k. First of all, the historical pose models from the KMM (before $p_{k,z-1}$ and after $p_{k,z}$ a correction) are loaded into the preprocessing submodule. Furthermore, the processing module is responsible for supplying any further information $y_{k,z-1}$ that may not have been integrated in the KB, for example additional feedback of the user indicating if a correction was successful or not. This data can be provided by the knowledge integration submodule of the KMM. Once all the information is available in the preprocessing module, the information can be passed to the learning module. The data given to the learning module should not be limited to the use of one local model. A more general learning model can be trained by considering other correction models applied for other poses and paths. In the learning module a model $l(p^*_{k,z-1}, p^*_{k,z}, y^*_{k,z-1})$ keeps getting trained until the learned correction $p^l_{k,z}$ can be applied on the same or similar problems. The training of the model can be implemented using statistical, heuristic or analytical approaches, depending on the required learning task and provided pose and path models.

6. Conclusion

The fully automated programming of robotic systems remains an unsolved problem in many industrial applications, where the robot programs are calculated based on models of the robotic system, which are incorrect or incomplete and cannot be updated. Thus, the continuous intervention of end users is still required to compensate the inaccuracy of the system model.

This paper presents a concept to update the robotic system model with the integration of expert knowledge. Furthermore, the acquired knowledge is used to train a correction model that can be used to solve similar problems leaving out the intervention of the end user. To achieve this, our work proposes three elementary modules involving: the acquisition, the modelling and the processing of knowledge. The first module is responsible for the acquisition of implicit and explicit knowledge in the system. The second module deals with the modelling of the system. A KB is introduced to facilitate the integration of expert knowledge, while preserving a high modular and flexible model of the robotic system. Furthermore, a novel approach was introduced for modelling robot poses and paths to achieve a high modular and extensive local modelling of the robotic system. Finally, the knowledge processing module is responsible to train a model based on the acquired knowledge to imitate the user handling for adjusting the robotic system.

Acknowledgements

The presented concept is being developed and tested within the project ‘CyMePro’ (Cyber-physical measurement technology for 3D digitization in the networked production) to achieve a high autonomous grade for programming robotic systems for measurement tasks. The project is kindly being funded by the Bavarian Ministry of Economic Affair, Energy

and Technology and coordinated by the VDI / VDE-IT (funding code ESB036 / 001). Furthermore, we would like to thank the partners AUDI AG and ZEISS Optotechnik GmbH for their continuous work and support in the project.

References

[1] International Federation of Robotics. World Robotics Industrial Robots Re-port 2017. IFR Statistical Department. 2017.
[2] Magaña A, Reinhart G. Herstellerneutrale Programmierung von Robo-tern," WT Werkstattstechnik. 2017.
[3] Hägele M, Nilsson K, Pires JN. Industrial Robotics. Handbook of Robotics. Springer Verlag. 2008; p. 963–986.
[4] Boston Consulting Group. How a Takeoff in Advanced Robotics Will Pow-er the Next Productivity Surge. The Shifting Economics of Global Manufacturing. 2015.
[5] Vogl W. Eine interaktive räumliche Benutzerschnittstelle für die Program-mierung von Industrierobotern (Doctoral dissertation). Herbert Utz Verlag, 2009.
[6] Guenter F, Hersch M, Calinon S, Billard A. Reinforcement learning for imitating constrained reaching movements. Advanced Robotics. 2007;21: p. 1521–1544.
[7] Neto P, Pires JN, Moreira A. High-level programming for industrial robot-ics: using gestures, speech and force control. IEEE International Conference on Robotics and Automation. 2009.
[8] Dillmann R. Teaching and learning of robot tasks via observation of hu-man performance. Robotics and Autonomous Systems. 2004;47: p. 109–116.
[9] DIN EN ISO 10218-2. Robots and robotic devices -- Safety requirements for industrial robots -- Part 2. ISO/TC 299. 2012.
[10] Beierle C, Kern-Isberner G. Methoden wissensbasierter Systeme-Grundlagen. Springer Vieweg. 2008; p.11.
[11] Björkelund A, Malec J, Nilsson K, Nugues P. Knowledge and skill repre-sentations for robotized production. IFAC Proceedings. 2011;44: p. 8999–9004.
[12] Tenorth M, Beetz M. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. The International Journal of Robotics Research. 2013;32,: p. 566–590.
[13] Munzert U. Bahnplanungsalgorithmen für das robotergestützte Remote-Laserstrahlschweißen (Doctoral dissertation). Herbert Utz Verlag. 2009.
[14] Hartfuss C. Wissensbasierte Programmierung von Industrierobotern zum Schutzgasschweißen im Stahlhochbau (Doctoral dissertation). Springer. 1996.
[15] LaValle SM. Planning Algorithms. Cambridge University Press. 2006.
[16] Kalisiak M. Toward more efficient motion planning with differential con-straints (Doctoral dissertation). University of Toronto. 2008.