

Evidence-Based Usability Engineering

Eduard Metzker¹, Harald Reiterer²

¹*DaimlerChrysler Research and Technology Centre, Software Technology Lab, HCI Research Group, P.O.Box 2360, D-89011 Ulm, Germany, eduard.metzker@daimlerchrysler.com*

²*University of Konstanz, Department of Computer and Information Science, P.O.Box D 73, D-78457 Konstanz, Germany, harald.reiterer@uni-konstanz.de*

Abstract: In this paper we propose an approach for the systematic integration of human-centered design (HCD) methods throughout the whole software development process called evidence-based usability engineering. We derive our approach from six claims that emphasize shortcomings and urgent requirements of current HCD practice. Instead of clinging to a fixed process model of the usability engineering process our approach advocates a paradigm of situated decision making. It enables software development teams to select optimal sets of HCD methods that match the engineering task at hand based on the cumulated experience of the software development organization. Qualitative and quantitative feedback on the efficiency of HCD methods is collected and integrated across project boundaries. Our approach is connected to existing process improvement frameworks such as UMM (Usability Maturity Model) via a meta-model. We present a first prototype of a novel kind of computer-aided usability engineering environment called ProUSE, that supports our evidence-based usability engineering approach.

Keywords: usability maturity, process improvement, human-centred design methods, computer-aided usability engineering environment

1. INTRODUCTION

The relevance of usability as a quality factor is continually increasing for software engineering organizations: usability and user acceptance are about to become the ultimate measurement for the quality of today's telematics applications, e-commerce web sites, mobile services and tomorrow's proactive assistance technology. Taking these circumstances into account human-centered design (HCD) methods for developing interactive systems are changing from a last minute add-on to a crucial part of the software engineering lifecycle.

It is well accepted both among software practitioners and in the human-computer interaction research community that structured approaches are required to build interactive systems with high usability. On the other hand specific knowledge about exactly how to most efficiently and smoothly integrate HCD methods into established software development processes is still missing [1]. While

approaches such as the usability maturity model (UMM) [2] or the DATech-Model [3] provide means to assess an organization's capability to perform HCD processes they lack guidance on how to actually implement process improvement in HCD. It often remains unclear to users of HCD methods if and why certain tools and methods are better suited in a certain development context than others [4]. We need strategies and tools that support engineering organizations in evaluating and selecting an optimal HCD method for a given development context and perform systematic process improvement in HCD. Little research has been done on integrating methods and tools of HCD in the development process and gathering knowledge about HCD activities in a form that can capture relationships between specific development contexts, applicable methods and tools and their impact on the engineering process [5].

In this paper we propose six claims that point out shortcomings and requirements of current HCD practice. First we take a look at existing HCD process models, trying to identify the organizational obstacles that hamper the establishment of these models in mainstream software development processes. Next we present results of a survey which examined how exactly HCD methods are applied in real projects and derive implications for tools to support process improvement in HCD. Finally we present a first prototype of an evidence-based computer-aided usability engineering environment (CAUSE) that we propose to address the shortcomings and requirements we have come across.

2. EXISTING HCD PROCESS MODELS: THEORY AND PRACTICE

A review on existing literature and case studies of some of the HCD approaches most applied revealed a number of organizational obstacles encountered in establishing HCD methods in mainstream software development processes [6]. We summarized these problems in claim 1-3.

Claim 1 *Existing HCD process models are decoupled from the overall software development process.*

Claim 2 *Most of the HCD approaches do not cover a strategy how to perform HCD methods and process models depending on the usability maturity of the software development organization.*

Claim 3 *Integrating UE Methods into mainstream software development process must be understood as an organizational learning task.*

To learn more about the way that HCD methods are actually used in software development organizations we conducted a study with software developers who are working on design of interactive systems in a variety of domainsⁱ [7]. The study

ⁱ DaimlerChrysler Aerospace (DASA) Ulm, Sony Fellbach, Grundig Fuerth and DaimlerChrysler Sindelfingen (all sites are located in Germany)

revealed that the organizations examined are practicing highly diverse individual development processes, however none of the UE development models proposed by [1, 8-10] are exactly used. Furthermore, the persons who are entrusted with the ergonomic analysis and evaluation of interactive systems are primarily the developers of the products. External usability or human factors experts or a separate in-house usability department are seldom available. Furthermore, few of the participants were familiar with methods like *user profile analysis* or *cognitive walkthroughs* which are regarded as fundamental from a usability engineer's point of view.

The UE methods that are considered to be reasonable to apply by the developers are often not used for the following interrelated reasons:

- There is no time allocated for UE activities: they are neither integrated in the development process nor in the project schedule.
- Knowledge needed for the performance of UE tasks is not available within the development team.
- The effort for the application of the UE tasks is estimated to be too high because they are regarded as time consuming.

From the analysis of the interviews we derived high level requirements for a software tool to support the improvement of UE processes. These high level requirements are summarized in claims 4-6.

Claim 4 *Support flexible UE process models*

Claim 5 *Support evolutionary development and reuse of UE experience*

Claim 6 *Provide means to trace the application context of UE knowledge*

3. THE EVIDENCE-BASED USABILITY ENGINEERING APPROACH

To address the shortcomings and to meet the requirements described in our claims we advocate an evidence-based approach to the improvement of HCD processes.

We define evidence-based usability engineering as follows:

- *Evidence.*

Something, such as a fact, sign, or object that gives proof or reasons to believe or agree with something.

- *Usability engineering*

(1) The application of systematic, disciplined, quantifiable methods to the development of interactive software systems to achieve a high quality in use;

(2) The study of approaches as in (1).

- *Evidence-based usability engineering.*

An approach for establishing HCD methods in mainstream software development processes, that uses

- (1) qualitative and quantitative feedback on the efficiency of HCD methods collected in projects and integrates this data across project boundaries for controlling and improving the quality and productivity of HCD activities.
- (2) concepts of organizational learning for integrating HCD knowledge, gathered as in (1), into software development processes.

The essence of the evidence-based approach is that we do not cling to a fixed process model of the usability engineering process, but instead follow a paradigm of situated decision making. In this approach HCD methods are selected for a given engineering task, e.g. *contextual task analysis*, based on the available evidence that they will match to the development context at hand.

After performing a method it should be evaluated if the method was useful for the development context or if it must be modified or discarded. The modification of the method should be recorded and stored for later reuse. Once a certain body of HCD knowledge is accumulated in that way, we have sound evidence for selecting an optimal set of HCD best practices for given development contexts. By continuously applying this procedure of conducting, evaluating and adapting HCD methods in a number of projects, an organization gradually adapts a set of HCD base practices to a wide variety of development contexts. This directly contributes to the general idea of software maturity models such as UMM [2]. According to these models organizations are highly ranked on a maturity scale, if they are capable to tailor a set of base practices according to a set of constraints such as available resources or project characteristics to solve a defined engineering problem.

So far we argue that the evidence-based approach requires four ingredients:

- A process meta-model, which guides the selection of HCD methods for a given development context and their integration in an overall software development process in a flexible, decision-oriented manner. The model must as well provide a strategy for evaluating, refining and capturing experiences for new development contexts thus promoting continuous process improvement and organizational learning in HCD.
- A semi-formal notation for structuring knowledge relating to HCD activities allowing to save it in an experience base. The experience base allows to keep track of documented best practices and their application context even if the underlying context factors such as processes, technologies, domains and quality standards are still evolving, A model is needed to formally relate the best practices of the experience base to a development context.
- A CAUSE for managing the experience base and allowing to predict optimal sets of HCD methods based on the available evidence and the experience of the development organization.
- An organizational concept for deploying, maintaining and evolving the experience base.

3.1 The Evidence-Based Meta-Model

Our evidence-based model comprises a set of organizational tasks to support the introduction, establishment and continuous improvement of HCD methods throughout the whole software engineering lifecycle. It helps to manage and tailor the HCD base practices defined in usability maturity models such as UMM [2] and the related methods practiced by the development organization according to specific constraints of the respective project and the experiences collected by the organization. The meta-model is based on our findings of experience-based improvement of user interface development processes [6]. We refined the model to the extent that we now use a formal model to relate a defined development context to a set of best practices. So the selection of an optimal set of HCD methods is guided by a model of the development context and the evaluation results of the HCD methods in real projects within the organization. In more detail the model consists of the following four logical steps:

- Step 1: Define, revise and extend the reference model

In the first step the reference model for the organization's HCD approach must be defined. If no reference model exists in the organization, one of the existing frameworks such as the process descriptions of UMM [2] or the DATech Prüfbaustein [3] could be used as a starting point for a reference model.

After performing one or more projects, the reference model, should be revised based on the experience collected in the projects.

- Step 2: Select suitable HCD base practices and the related methods and integrate them into the software development process practiced

In this step base practices are selected from the reference model to be integrated in the overall software development process. However, further important factors have to be considered, e.g. the type of system to be developed and project constraints like budget and schedules. This information must be mapped into a context model and guides the selection of appropriate HCD methods for the selected base practices. The HCD methods which have been selected for the improvement of the development process have to be integrated in the model of the practiced software development lifecycle and the project plan and complement the overall engineering process used in the project.

- Step 3: Support effective performance of the defined HCD methods

Generally, at this step in the model resources have already been allocated for HCD activities, e.g., a usability engineer was nominated, who is responsible for coordinating and supporting the execution of the various HCD activities of the new process. However, the efficiency and impact of the proposed HCD methods must be increased by providing the development team with best practices, tools and reusable deliverables of past projects (e.g. templates for usability test questionnaires, results of conceptual task analysis or user interface mockups) which facilitate effective performance of the selected HCD methods. This set of information should be easily accessible for all participants of HCD activities.

- Step 4: Collect and disseminate best practices and artifacts concerning HCD tasks

The HCD methods applied in a project should be rated by the project team members. These ratings form the rationale for following projects to adopt or reject HCD methods. Methods poorly rated by project team members in a project have a low likelihood of being reused in later projects.

During the execution of HCD activities, artifacts with a high value for reuse are generated by the participants of HCD activities, for example, templates for usability tests, reusable code fragments, or an experience on how to most efficiently conduct a user profile analysis for assistance systems. Observations like these comprise HCD experience that have to be captured and organized in best practices along with the development context in which they apply to allow for easy reuse in the same or subsequent projects.

3.2 USEPACKs and the HCD Experience Base

To capture and evolve HCD knowledge for reuse and process improvement, we need a semi-formal notation for structuring process knowledge relating to HCD methods. For this purpose we have developed the concepts of USEPACKs (Usability Engineering Experience Package) and a context model (see section 4.3). While USEPACKs are used to capture HCD methods the context model is used to formally relate these methods to a development context.

A USEPACK encapsulates best practices on how to most effectively perform certain HCD activities and includes the related artifacts like documents, code fragments or templates that facilitate the compliance with the best practice described. A USEPACK is structured into four logical sections:

1. The *core information* permits authors to describe the main message of a USEPACK. It is organized according to the pyramid principle for structuring information [11]. The information first presented to the reader has a low level of complexity, allowing the reader to quickly decide if the USEPACK is worth further exploration. With further reading, the degree of complexity rises, introducing the reader to the experience described. The core information section includes the fields *title*, *keywords*, *abstract*, *description* and *comments*.
2. The *context situation* describes the development context related to the experience in question. The context situation is generated by using the context model, allowing the authors and readers of USEPACKs to utilize a shared vocabulary for contextualizing and accessing USEPACKs.
3. A set of *artifacts and links*. Artifacts such as checklists for user profile analysis or templates for usability test questionnaires, facilitate the efficient compliance with the best practice. They represent an added value to the readers of a USEPACK. Artifacts allow readers to regain time spent on exploring the package by using the supplied artifacts to simplify their work.

Links are pointing to related resources that should be considered by the reader of a USEPACK.

4. The *USEPACK rating scheme* allows engineering teams to assess the quality of a USEPACK along a defined set of dimensions. The ratings collected will be evaluated and form the rationale for selecting optimal USEPACKs for future projects.

3.3 The Context Model

The context model serves as a template to construct the *context situation* for USEPACKs – a semi-formal description of the development context in which the experience described by a USEPACK can be applied. It is organized in a hierarchical structure, divided into sections which contain groups of *context factors*. On the one hand, authors of USEPACKs can use the context model to easily construct a description of the context in which the information of a USEPACK can be applied by selecting appropriate context factors from the model. On the other hand, readers of USEPACKs can use the context model for retrieval of USEPACKs by specifying a context situation which reflects the development context for which they need support. Currently a context model containing the following four sections is used:

- The *process context* section provides context factors to describe to which base practices of the UMM reference model a best practice is related.
- The *project context* section provides context factors to describe project constraints like the size of the development team, budget or project duration which are related to the experience cited.
- The *domain context* section provides context factors to describe elements of the domain related to the experience described. Top-level context factors of this section specify domains in terms like ‘home entertainment systems’ or ‘car driver assistance systems’, which can be subsequently refined to capture more detailed domain attributes.
- The *technology context* section provides context factors to describe features of technologies related to the experience such as ‘gesture recognition’ or ‘speech input’.

3.4 A Computer-Aided Usability Engineering Environment

To increase the impact of the evidence-based usability engineering approach tool support is needed. In the BMBFⁱⁱ lead project EMBASSIⁱⁱⁱ we currently develop a prototypical CAUSE called ProUSE (Process centred Usability Engineering

ⁱⁱ German Ministry of Education and Research

ⁱⁱⁱ Electronic Multimedia Operating and Service Assistance

Environment)^{iv}. ProUSE consists of an HCD experience base and three logical components for planning, selecting, applying, evolving and assessing HCD methods and reusable artifacts.

The experience base is seeded with an initial set of best practices in form of USEPACKs. In our case we have adopted a variety of usability engineering methods from Nielsen and Mayhew [1, 8] but in general any HCD approach should be appropriate. The REUSE (Repository for Usability Engineering Experience) [7] component is used to capture, manage and evolve best practices related to HCD activities. It assists in documenting best practices using the USEPACK concept and relating them to a formal development context using a context model and storing them in the experience base.

SATUP (Setup Assistant for Usability Engineering Processes) is used to plan HCD activities for a software development project. Given all available context information on the process (e.g. which HCD base practices should be performed), project (e.g. duration, budget, team size), domain and technology context factors, SATUP will propose optimal HCD methods and reusable artifacts based on the accumulated experience of the development organization stored in the experience base. If there are alternative USEPACKs available for a defined context situation, SATUP will compare the available ratings for all USEPACKs using a fuzzy multi-criteria decision making approach and propose the optimal USEPACK.

Once an optimal HCD process was planned, CUES (Cooperative Usability Engineering Workspace) can be used by a distributed development team to access, perform and assess the HCD methods selected. The ProUSE prototype is based on Java technologies and integrated via a web portal concept which makes the modules available through intranet and web browser.

4. DEPLOYMENT OF PROUSE IN AN INDUSTRIAL SETTING

Before the ProUSE environment is deployed in a development organization for the first time, the systems experience base should be seeded as described in section 3.4. After this first step, the experience base contains a set of USEPACKs that describe methods for each base practice of the reference model used. These USEPACKs will usually have a poor context situation. Imagine now that a large software development organization is developing assistive, interactive home entertainment components and has recognized that usability aspects will play a major role in a new project. In that project an intelligent avatar for assisting users in programming their video recorders should be created. The organization decides that

^{iv} Special thanks go to Michael Plichta, Ingo Gruell, Gerhard Reuss and Christoph Ballhause from the DaimlerChrysler Research Center Ulm for the great job they did in implementing the ProUSE system.

the overall development process should be supplemented with usability engineering activities. One usability engineer is available for the project but the rest of the development team is inexperienced in usability engineering.

When the project is planned, the usability engineer logs in to the ProUSE system and starts the SATUP component. The usability engineer uses SATUP to create a new project and roughly describes the projects usability aspects in textual form. After this first rough textual description, the usability engineer uses the context model to map the projects usability constraints to the context factors of the context model thereby creating a context situation for the project. After the usability engineer modelled the development context, the project is accessible to the development team.

Members of the development team can now access the new project via CUES. CUES creates a workspace where the USEPACKs relevant for the defined context situation can be accessed by the development team to support their usability engineering activities. CUES provides the development team with a tailored view on

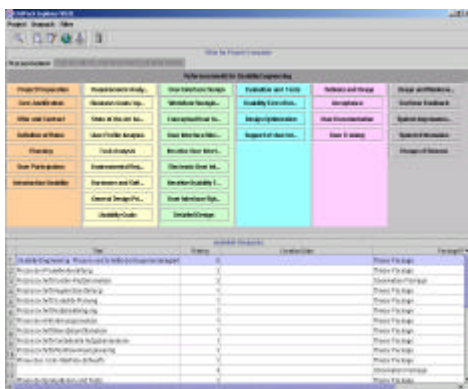


Figure 1 : Browsing project related experiences with CUES

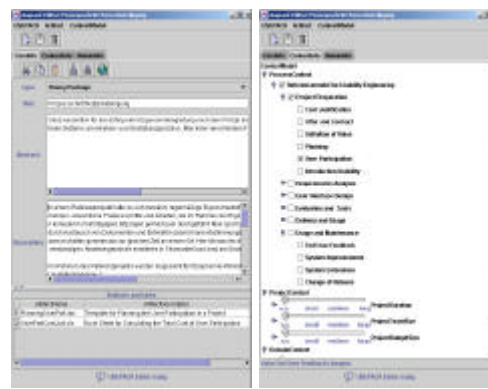


Figure 2 : Capturing experiences with REUSE

the experience base, shielding them from irrelevant information (Figure 1).

If a certain method was not useful to the development team, the team members can assign a poor rating to the respective USEPACK. This will reduce the likelihood for the USEPACK to be selected in subsequent projects with the same context situation. On the other hand team members can assign high ratings to USEPACKs which had a positive impact on their work thus increasing the likelihood that the respective USEPACK will be reused in subsequent projects. By applying the best practices described in a USEPACK members of the development team will make valuable experiences. These experiences can be captured as extensions or comments of an existing USEPACK or can be captured in a new USEPACK.

Assume that the development team used the USEPACK “Contextual Task Analysis” to perform the contextual task activity. Though they found the general

guideline useful, they soon discovered that the contextual task analysis for designing an intelligent avatar is fundamentally different from the general method they found in the USEPACK. To capture the modifications to the general method they create a new USEPACK ‘Contextual Task Analysis for Intelligent Assistants’ using REUSE. They enter their experiences in the USEPACK and attach documents and templates that can be reused when performing a contextual task analysis for an intelligent assistant. Figure 2 shows how a USEPACK is created using REUSE.

On the left side you see how the core information and reusable artefacts of a USEPACK are captured by using the USEPACK template. On the right side you see how the context situation for a USEPACK is captured by using the context model. The user can easily relate the experience described in a USEPACK to a context situation by selecting appropriate context factors from the context model.

The next time a project team has to design an intelligent assistant, it will have access to the improved contextual task analysis method and the reusable artifacts. These improvement activities will prevent subsequent development teams to encounter similar problems and will enable them to perform the task more efficiently.

How the tool support provided by ProUSE is related to our evidence-based usability engineering approach is depicted in Figure 3. On the left side you see the steps of the evidence based meat-model as described in section 3.1. On the right side you see which component of ProUSE supports the respective step.

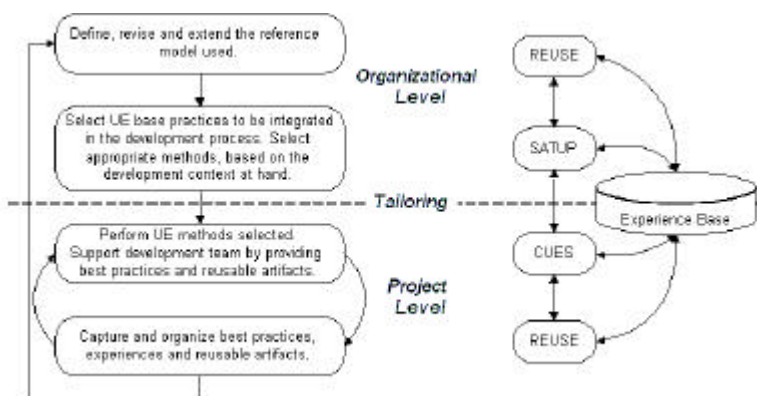


Figure 3 : Tool Support for Evidence-Based Usability Engineering

5. DISCUSSION

Evidence-based usability engineering provides a multidisciplinary approach for integrating usability engineering activities into software development processes. In fact the approach has its foundations in the intersection of three disciplines: usability engineering, software process improvement and organizational learning. This interdisciplinarity is directly caused by our practical goal: to support the introduction,

establishment and continuous improvement of usability engineering activities in software engineering processes. We want to integrate usability engineering activities into software development processes to improve the engineering process for interactive software systems. As a tool to achieve process improvement we use concepts of organizational learning.

It has often been claimed that there is a gap between the need for structured approaches in HCD on the one side and creativity in the process on the other side. The evidence-based usability engineering approaches tries to balance these needs. It does not force development organizations to run a second usability engineering lifecycle in parallel to the actual development lifecycle nor does it force engineering organizations to discard their established processes and replace them by a completely usability centered approach. Following our approach usability engineering activities are not linked together in a fixed process model. Instead they are arranged in a construction-set-like reference model linked by a context model. The reference model and its base practices describe what should be done. The best practices together with their context situation describe how it should be done in a defined context situation. Engineering teams can select, evaluate, refine and extend base practices to meet their needs. By continuously evolving base practices to contextualized best practices (USEPACKs) engineering organizations compile a set of optimized methods which they can dynamically link into their development process.

Recently the idea of process maturity was introduced to the field of usability engineering [2, 3]. The rationale behind usability maturity models is that there are direct dependencies between an organization's capability to tailor its HCD processes according to various constraints, the quality of the development process performed and the quality in use of the interactive software system. Our approach is geared to support software engineering organizations in reaching high usability maturity levels. At a low usability maturity development organizations can use the experience base as a repository for base practices which are found in the initial set of USEPACKs (seed of the experience base) and which enable the engineering teams to perform those base practices. By applying the base practices in various projects, the development organization is able to derive USEPACKs for each base practice which capture project constraints such as 'size of development team' or 'available budget'. Thus the organization develops a set of more contextualized methods for each base practice, which will enable the organization to control its usability engineering activities.

6. ACKNOWLEDGEMENTS

Special thanks go to Michael Plichta, Ingo Gruell, Gerhard Reuss and Christoph Ballhause from the DaimlerChrysler Research Center Ulm for the great job they did in implementing the ProUSE system.

This research was sponsored in part by the BMBF award #01 IL 904 B7 of the EMBASSI project. We would like to thank all participants of our studies.

7. REFERENCES

1. Mayhew, D.J., *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design*. 1999: Morgan Kaufman Publishers.
2. Earthy, J., *Human Centred Processes, their Maturity and their Improvement*. IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT'99), 1999, pp. 117-118, see also: http://info.lboro.ac.uk/research/husat/eusc/guide/tr_ump_c.doc.
3. DATech Frankfurt/Main, *DATech Prüfbaustein: Qualität des Usability Engineering Prozesses*, . 2001, Deutsche Akkreditierungsstelle Technik e.V.
4. Welie, M.v. *Breaking Down Usability*. in IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT'99). 1999. Endinburgh, UK: IOS Press: pp. 613-620.
5. Henninger, S., *A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design*. *Interacting with Computers*, 2000. 12(3): pp. 225-243, .
6. Metzker, E. and M. Offergeld. *An Interdisciplinary Approach for Successfully Integrating Human-Centered Design Methods Into Development Processes Practiced by Industrial Software Development Organizations*. in IFIP TC2/TC13 WG2.7/WG13.4 Eighth IFIP Conference on Engineering for Human Computer Interaction (EHCI'01). 2001. Toronto, Canada: Springer: pp. 21-36.
7. Metzker, E. and M. Offergeld. *REUSE: Computer-Aided Improvement of Human-Centered Design Processes*. in *Mensch und Computer*, 1. Fachübergreifende Konferenz des German Chapter of the ACM, (MC2001). 2001. Bad Honnef, Germany: Teubner Verlag: pp. 375-384.
8. Nielsen, J., *Usability Engineering*. 1994: Morgan Kaufman Publishers.
9. Constantine, L.L. and L.A.D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. 1999: Addison-Wesley.
10. Beyer, H. and K. Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*. 1998: Morgan Kaufmann Publishers.
11. Minto, B., *The Pyramid Principle - Logic in Writing and Thinking*. 3 ed. 1987, London: Minto International Inc.