Einbeziehung von Benutzer-Absichten in pervasive Dienstumgebungen

Zusammenfassung der Masterarbeit

How to take the user on his word
Expressing and interpreting user intention
in a pervasive service environment

eingereicht im Juni 2005 am
Institute National des Sciences Appliqués de Lyon

von
cand. inform.
Pascal Bihler

Zur Bewertung der Gesamtleistung als
Diplomarbeit
am
Institut für Telematik
Fakultät für Informatik
Universität Karlsruhe (TH)

Betreuung Lyon:
Prof. Dr. Ing. Lionel Brunie
Dr. Vasile-Marian Scuturici

Betreuung Karlsruhe:
Prof. Dr. rer. nat. Wilfried Juling
Dr. Ing. Martin Gaedke
Dipl. Inf. Martin Nussbaumer

11. September 2005

Ich versichere, diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Karlsruhe, den 11. September 2005

Zusammenfassung

Die Einführung von pervasivem Computing in das tägliche Leben bedeutet nicht nur für die Anwender, sondern auch für die Anwendungsdesigner eine große Herausforderung. Bisher wohlbekannte und präzise defininerte Schnittstellen wie „Tastatur“ und „Maus “ machen Platz für andere, intuitivere Interaktionsarten. In einer pervasiven Dienstumgebung ist es Aufgabe der Middleware, die Absicht eines Benutzers zu erkennen und zu modellieren sowie Mehrdeutigkeiten bei der so gegebenen Definition einer pervasiven Aktion aufzulösen. Diese Arbeit präsentiert die Anfragesprache für Pervasive Dienst-Aktionen (PsaQL). PsaQL bietet eine Formalisierung, um die Absicht eines Benutzers bei der Verwendung verketteter pervasiver Dienste auszudrücken. Diese Arbeit beschreibt eine Vorgehensweise, diese Benutzerabsicht in eine ausführbare Aktion zu übersetzen und präsentiert Algorithmen, die diese Übersetzung realisieren. Am Beispiel von PerSE, einer von unserer Forschungsgruppe in Lyon entwickelten Platform zur Verwaltung von pervasiven Diensten, werden Möglichkeiten der Implementierung dieses Prozesses aufgezeigt und Überlegungen zur Performanzmessung solcher Algorithmen gegeben, gefolgt von konkreten Ergebnissen einer solchen Messung an einem Prototyp.

Abstract

The introduction of pervasive computing environments in everyday life will not just be a big step for users, but also for application designers. The well defined interaction interfaces “keyboard” and “mouse” will make place for other, more intuitive ways of interaction. It is the challenge for a pervasive system middleware to capture and model the user’s intention in a smart way and to solve ambiguousness in the user’s expression of a pervasive action. This thesis introduces the Pervasive Service Action Query Language (PsaQL). PsaQL formalize the description of a user intention using composed pervasive services. The thesis describes a way of translating the user intention into an executable action and propose algorithms performing this translation. Considerations to implement this process are given within the scope of PerSE, a pervasive service environment developed by our research group in Lyon, together with general evaluation metrics for such algorithms and prototype benchmark results.

Inhaltsverzeichnis

1 Einführung

1.1 Problematik

Der menschliche Lebens- und Arbeitsraum wird in Zukunft von ubiquitärem bzw. pervasivem Computing geprägt sein: In unsere Umgebung sind elektronische Elemente eingebettet, mit Rechenkapazität und Kommunikationsmöglichkeiten ausgestattet. Diese kleinen „Computer“ werden aber nicht mehr als eigenständige, „unnatürliche“ Rechner wahrgenommen, sondern als natürliche Bestandteile einer intelligenten Umgebung, wie heutzutage schon zum Beispiel die Armbanduhr auch keinen „Fremdkörper“ mehr darstellt.

Wird ein Rechner als solcher aber nicht mehr wie heute wahrgenommen, kann auch die Art und Weise, in der mit diesen Geräten umgegangen wird, nicht mehr die gleiche bleiben. An die Stelle von Tastatur und Maus treten andere, intuitivere Interaktionsschnittstellen wie zum Beispiel Sprach- oder Gestenerkennung. Die Entwicklungsrichtung ist klar definiert: Nicht mehr der Anwender muss sich in die Arbeitsweise eines Rechners einarbeiten, sondern das elektronische System sollte in der Lage sein, sich dem Benutzer anzupassen und auf dessen Wünsche und Bedürfnisse einzugehen. In unserer Vision von pervasivem Computing ist die Schnittstelle zwischen dem Menschen und dem Rechner so natürlich und intuitiv gestaltet, dass auch ungeübte Anwender ohne große Einarbeitung von der intelligenten Umgebung profitieren können. Diese Umgebung definiert sich nicht durch sich selbst, sondern dadurch, dass sie das Leben des Anwenders vereinfacht.

Während heutzutage Rechner die meisten Aktionen als Reaktion auf einen Befehl des Benutzers, also „reaktiv“, ausführen, so werden zukünftige Systeme mehr und mehr „proaktiv“ arbeiten müssen (vgl. Abb. 1).[1] Nach unserem Verständnis arbeitet ein Rechensystem „proaktiv“, wenn es „versteht“, was die Absicht eines Benutzers ist, wenn es Aktionsentscheidungen auf der Basis früher ausgeführter Aktionen fällt, wenn es dem Anwender zum richtigen Zeitpunkt sinnvolle Aktionen anbietet und diese dann auch ausführt.

Hiermit sind auch schon die beiden Möglichkeiten genannt, die Absicht eines Benutzers zu erkennen: zum einen auf Basis einer expliziten Aktion des Anwenders (beispielsweise einer gesprochenen Aufforderung), zum anderen durch die Auswertung von Kontext-Informationen und den Vergleich dieser mit der Ausführungs-Geschichte des Systems. Ein solches System kann zum Beispiel eine pervasive Dienstumgebung sein. Dabei bieten in die Umwelt eingebettete, elektronische Geräte so genannte „Dienste“ *(Services)* an, die miteinander verknüpft werden und so eine sinnvolle Aktion zur Verfügung stellen können. Die Absicht eines Anwenders zu erfassen und zu interpretieren sollte jedoch nicht die Aufgabe des Dienst-Entwicklers sein, sondern vielmehr ist es wünschenswert, diese Funktionalität schon in die Middleware des Systems einzubetten.

Am LIRIS[2] in Lyon sind wir dabei, eine solche Middleware, PerSE (Pervasive Service Environment), zu entwickeln. Diese Arbeit legt die Grundlagen dafür, die Absicht des Anwenders beim Ausführen von Aktionen in PerSE zu berücksichtigen und definiert Formalismen, um solche Aktionen als Verknüpfung von Diensten auszudrücken.

[1] Natürlich ist es wichtig, dem Anwender nicht das Gefühl zu geben, er würde von der Technik beherrscht.

[2] Laboratoire d'InfoRmatique en Images et Systèmes d'information, Informatik-Labor für Bildverarbeitung und Informations-Systeme

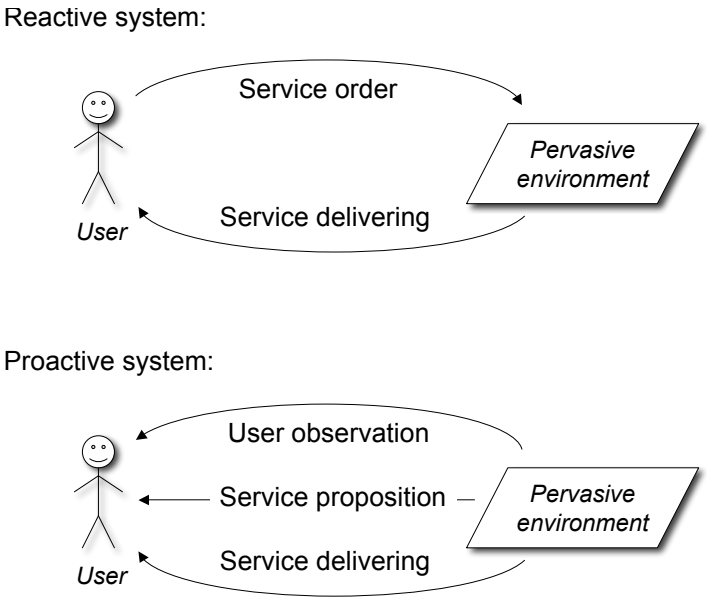

Abbildung 1: Vergleich von reaktiven und proaktiven Systemen: reaktive Systeme warten auf einen spezifischen Befehl, um eine Aktion auszulösen, während proaktive Systeme versuchen, auf Basis von Kontext-Informationen selbstständig sinnvolle Aktionen zu ermitteln.

1.2 Erarbeitete Forschungsergebnisse

Diese Arbeit behandelt die Überführung einer Benutzer-Absicht in eine vollständige Aktion, die in einer pervasiven Dienstumgebung ausgeführt werden kann. Dabei werden vor allem folgende Beiträge erbracht:

- Einführung der Konzepte *teilweise definierte Aktion*, *Aktions-Graph*, und *vollständige Aktion*.
- Entwicklung von *PsaQL*, einer formalen Sprache zur Beschreibung einer Benutzer-Absicht.
- Entwurf von *Algorithmen*, die eine Benutzer-Absicht in eine ausführbare Aktion überführen.
- Vorschläge zur *Implementierung* dieser Transformations-Algorithmen.
- Überlegungen zu *Performanz-Messungen* und *Benchmark-Ergebnisse* eines Prototyps.

1.3 Aufbau der Arbeit

Zunächst werden Forschungsarbeiten genannt, die mit dieser Arbeit in Beziehung stehen, danach wird anhand von PerSE auf die Eigenheiten und Herausforderungen pervasiver Dienstumgebungen eingegangen. Der folgende Abschnitt behandelt die Einbeziehung von Benutzer-Absichten in pervasiven Dienstumgebungen. Dafür wird eine formale Sprache eingeführt, die die Absicht eines Benutzers zum Ausführen einer spezifischen Aktion durch das System ausdrückt. Die notwendigen Algorithmen zur Überführung der Benutzer-Absicht in eine ausführbare Aktion werden im nächsten Abschnitt vorgestellt, gefolgt von Überlegungen zur Performanz-Messung und beispielhaften Ergebnissen einer solchen Messung an einem Prototyp. Am Ende der Arbeit stehen einige Schlussbemerkungen und eine Sammlung offener Fragen, die sich an diese Arbeit anschließen.

2 Verwandte Forschungsarbeiten

Die Entwicklung pervasiver Anwendungen, die den Benutzer in die Anwendungslogik mit einbeziehen, wurde unter anderem in [SG03] und [SPP+03] untersucht. Besonders die von El Kathib et al. [EKvBS04] entwickelte Plattform ist im Rahmen unserer Arbeit von Interesse, da sie ähnlich wie PerSE versucht, die Zufriedenheit des Anwenders als Qualitätsmaßstab der Anwendung zu verwenden. Bei El Kathib geht es um die Anpassung von Multimedia-Datenströme mit Hilfe von aneinander gereihten Codeumsetzern, während PerSE versucht, eine solche Dienst-Verknüpfung nicht nur auf Multimedia-Daten beschränkt, sondern generischer zu realisieren. Beide Ansätze nehmen zur Problemlösung Graphenrepräsentationen zu Hilfe, aber El Kathib präsentiert in seiner Arbeit keine formale Sprache, die zur abstrakten Beschreibung einer Benutzereingabe verwendet werden kann. Des weiteren wurde die in seiner Arbeit vorgestellte Plattform bisher hauptsächlich für relative starre Netzwerkstrukturen, wie sie zum Beispiel im Internet gegeben sind, optimiert. Da jedoch Mechanismen zur Fehlertoleranz vorgesehen sind, scheint ein Einsatz der Plattform auch in pervasiven Umgebungen möglich.

Andere benutzerorientierte pervasive Plattformen sind zum Beispiel Gaia [RHC+02] und Aura [GSSS02]. In Gaia steht dem Entwickler eine spezielle Programmiersprache zur Verfügung, mit der dieser benutzerorientierte Aktionen modellieren kann, indem er im System angebotene Dienste sinnvoll kombiniert. Aura geht einen Schritt weiter und versucht, sämtliche direkte Interaktionen zwischen dem Benutzer und dem System zu vermeiden, weswegen auch diese Arbeit keine Modellierung für die Benutzerabsicht einführt.

Der Ansatz, unabhängige und verteilte Dienste miteinander zu verknüpfen und entlang dieses Pfades komplexe Anpassungsfunktionen zu realisieren, wurde erstmalig im Ninja Environment [GWvB+01] realisiert, Buchholz et al. ergänzten dies um eine Optimierung der Anpassungswege [BB03]. M. Vallée präsentierte darauf aufbauend ein System, das, ähnlich wie PerSE, eine dynamische Dienstverknüpfung zur Lösung von Problemen in einer intelligenten Umgebung verwendet: Ausgehend von einem *abstrakten Plan* wird in einem Verknüpfungsalgorithmus unter der Zuhilfenahme von Kontextinformationen eine konkrete Lösung, ein *detaillierter Plan* berechnet [VRV05]. Die Arbeit behandelt besonders die Bereiche Dienstbeschreibung und den Verknüpfungsalgorithmus, führt jedoch keine formale Beschreibung der Benutzerabsichten in der intelligenten Umgebung ein. Der Ansatz, auf einer Bibliothek vordefinierter abstrakter Pläne aufzubauen, ähnelt zwar unserer Idee, die Ausführungsgeschichte bei der Berechnung konkreter Aktionen in Betracht zu ziehen, er ist aber rein statisch realisiert.

C. Popien präsentiert eine formale Sprache, um Dienstanfragen in Rechnernetzen zu beschreiben. Dabei wird besonders darauf Wert gelegt, den gewünschten Dienst anhand von semantischen Informationen zu ermitteln, eine Verknüpfung mehrerer Dienste ist aber darin nicht vorgesehen [PM94].

Unsere Arbeit wurde stark von den Entwicklungen im Bereich der Semantic Web Services beeinflusst, wie sie beispielsweise in [MSZ01] beschrieben werden. Um zum Beispiel gültige und flexible Aktionsgraphen zu erzeugen, wird in einer späteren Entwicklungsstufe unserer Plattform OWL-S [MvH04] zum Einsatz kommen. Semantische Verknüpfung von Web Services wurden unter anderem von [SvdAB+03], [SdF03], [SPAS03], [VR04] und [Bra05] eingeführt. Die allgemeine Abbildung von Anfragen auf Web-Ressourcen unter Zuhilfenahme einer Beschreibung ihrer Semantik zeigt [NSDM03].

Eine wichtige Eigenschaft pervasiver Umgebungen wird die Einbeziehung von Kontextinfor-

mationen sein [MYA+05]. PerSE verwendet diese Kontextinformationen, um aus allen möglichen Dienst-Kombinationen die sinnvollste zu ermitteln. Frühere Ansätze, die kontextbasierte Anpassung in pervasiven Umgebungen verwenden, sind [RC03], [Mos03] und [VR04]. Die Bedeutung der Begriffe „Kontext“ und „Anpassungsfähigkeit“ im Bereich pervasiver Anwendungen wurde im Rahmen dieser Arbeit detaillierter untersucht (vgl. `http://liris.wh4f.de/adaptation.pdf`). Obwohl „context-awarness“ gerade im Bereich des pervasiven Computings intensiv erforscht wird, kristallisierte sich bislang noch kein optimales Verfahren zur Implementierung derselben heraus.

3 PerSE: Eine pervasive Dienstumgebung

In diesem Abschnitt sollen die wichtigsten Eigenschaften pervasiver Dienstumgebungen zunächst informell eingeführt werden. Eine formale Definitionen der Schlüsselkonzepte findet sich in den darauf folgenden Abschnitten der Arbeit.

Pervasive Dienstumgebungen ermöglichen die sinnvolle Zusammenarbeit unabhängiger Dienste, die auf pervasiven Geräten ausgeführt werden, um eine komplexe Aktion zu realisieren. Solche Dienste können beispielsweise ein Dateisystem, ein Übersetzungsprogramm oder eine Projektion mit Hilfe eines Beamers sein. Die Middleware der Dienstumgebung fügt die vorhandenen Dienste zusammen, diese Verknüpfung nennen wir *vollständige Aktion*.

PerSE (Pervasive Software Environment) ist eine konkrete Implementierung einer solchen Dienstumgebung. Dabei werden die Dienste selbst von Basisanwendungen, so genannten *PerseBases* verwaltet, die untereinander logisch verbunden sind. Die Basisanwendungen sind auch dafür verantwortlich, auf der Grundlage einer erkannten Benutzer-Absicht eine *vollständige Aktion* zu berechnen. Diese vollständigen Aktionen werden in PerSE als zusammenhängende Graphen modelliert. Zur Berechnung wird von einer Modellierung der Benutzer-Absicht, einer *teilweise definierten Aktion*, ausgegangen, wobei bekannte Informationen zu Kontext und Ausführungsgeschichte mit einbezogen werden. Dieser Vorgang ist in Abb. 2 skizziert.

Ein Beispiel (siehe Abb. 3) verdeutlicht den Prozess: *Eine Person betritt einen Vortragsraum und möchte dort eine Präsentation seiner Urlaubseindrücke vorführen. Heutzutage muss er umständlich Dateien kopieren bzw. zunächst sicherstellen, dass die notwendigen Programme auf dem Präsentationsrechner installiert sind und er über ausreichende Zugriffsrechte auf diesem Rechner verfügt. In einer* PerSE*-gestützten Umgebung sind Beamer und Steuerrechner unsichtbar im Raum verborgen, über Funk nimmt jedoch das Notebook Kontakt mit der intelligenten Einrichtung auf. Drückt der Anwender nun seinen Wunsch aus, indem er zum Beispiel sagt: „Zeige die Präsentation mit dem Sonnenuntergang auf dem Beamer hier“, so interpretiert die Basisanwendung auf seinem Rechner diesen Wunsch als* teilweise definierte Aktion*: Diese enthält einen Dienst, der Präsentationen ausliefern kann (das* Dateisystem*) mit dem Attribut* Sonnenuntergang*, sowie einen lokalen Dienst namens* Beamer*. Mit Hilfe des Wissens um die lokal verfügbaren Dienste, den vorhandenen Kontextinformationen und der Ausführungs-Geschichte konstruiert die PerseBase nun einen Graphen aller möglichen und sinnvollen Dienstkombinationen, die der teilweise definierten Aktion genügen. Aus diesen wählt ein Algorithmus (vgl. Abschnitt 5.1) eine „beste Lösung“ aus und schließlich wird die gewünschte Aktion ausgeführt, die Präsentation also angezeigt.*

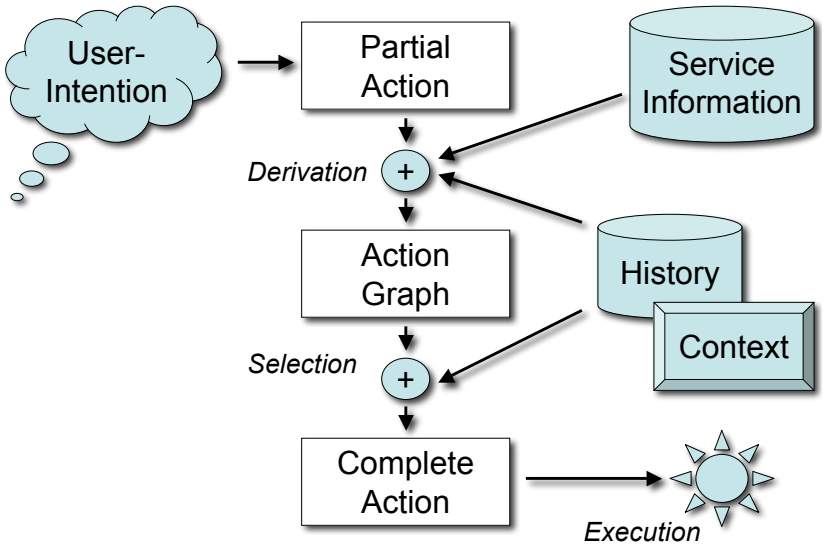

Abbildung 2: Eine Benutzer-Absicht *(user intention)* wird in eine vollständige Aktion *(complete action)* überführt, indem aus allen möglichen und passenden Dienst-Kombinationen, dem Aktions-Graphen *(action graph)*, die optimale Lösung unter Zuhilfenahme von Kontext-Informationen und Ausführungs-Geschichte ausgewählt wird.

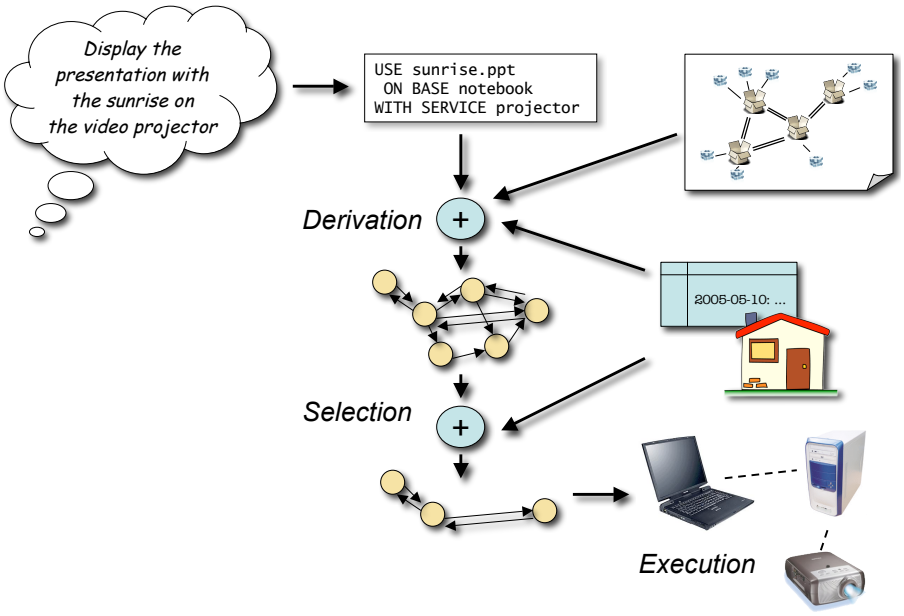

Abbildung 3: Auf einem im Raum eingebetteten Beamer wird eine Präsentation dargestellt.

4 Modellieren von Benutzer-Absichten - PsaQL

Möchte man ein Computersystem nutzen, so weiß man zumeist recht gut, was man machen will, selbst wenn man nicht genau weiß, wie. Es scheint recht klar zu sein, was das System zu tun hat, aber normalerweise ist die vom Benutzer gegenebene Intention nicht „vollständig“ im Sinne einer elektronisch ausführbaren Aktion. Ein darauf angepasster Algorithmus muss aufgrund der spärlich gegebenen Informationen in der Benutzer-Absicht eine vollständige Aktion berechnen, die in dieser Situation am besten der gegebenen, teilweise definierten Aktion entspricht.

Um diese teilweise definierte Aktion systemunabhängig darstellen zu können, haben wir in dieser Arbeit die formale Sprache *PsaQL* (Pervasive Service Action Query Language) entwickelt. Wie SQL in relationellen Datenbanksystemen als Anfragesprache verwendet wird (auch wenn das für den Anwender nicht immer sichtbar ist), so erfüllt PsaQL diese Aufgabe in pervasiven Dienstumgebungen. Ein Ausdruck in PsaQL hat so auch eine gewisse Ähnlichkeit mit einem SQL-Ausdruck. Für das im vorigen Abschnitt eingeführte Beispiel könnte dies sein:[3]

```
USE sunrise.ppt ON BASE notebook WITH SERVICE projector
```

Die Definition von PsaQL in BNF-Notation ist wie folgt gegeben:

```
<partial_action> ::= USE <action_part> [<ext_action>]
<ext_action> ::= WITH <action_part> [<ext_action>]

<action_part> ::= <attr_constr_def>[ FOR <service_constr_def>]
                                    [ ON <base_constr_def>] |
                  <service_constr_def>[ ON <base_constr_def>] |
                  <base_constr_def>

<base_constr_def> ::= BASE <base_constraint>[ AS <name>]
<base_constraint> ::= <name> | LIKE "<partial_name>"

<service_constr_def> ::= SERVICE <service_constraint>[ AS <name>]
<service_constraint> ::=  <name> | LIKE  <partial_name>

<attr_constr_def> ::= (<name> | LIKE <partial_name>)[ AS <name>]

<name>  ::= {<‘a’-‘z’, ‘A’-‘Z’, ‘0’-‘9’,‘_’>}
<partial_name>  ::= {<‘a’-‘z’, ‘A’-‘Z’, ‘0’-‘9’,‘^’,‘$’,
                            ‘(’,‘)’,‘[’,‘]’,‘.’,‘+’,‘*’,‘?’, ...>}
```

Mit PsaQL kann eine Aktion in sehr unterschiedlicher Granularität beschrieben werden. Für die Darstellung einer vollständigen Aktion haben wir aus Implementationsgründen und um PsaQL nicht zu überladen ein XML-Format definiert. Mit dieser Beschreibung ist es dann möglich, vollständige Aktionen zwischen verschiedenen Programm-Modulen oder unterschiedlichen Systemen auszutauschen. Ein Prototyp, bei dem auf Basis einer teilweise definierten Aktion eine vollständige Aktion berechnet wird, ist unter der Internetadresse `http://liris.wh4f.de/perse` verfügbar.

[3] In diesem Beispiel ist dem Anwender bereits bekannt, dass seine Präsentation in einer Datei namens „sunrise.ppt“ auf einem Rechner mit der Bezeichnung „notebook“ gespeichert ist und dass der Beamer auf den Namen „projector“ hört. Besitzt er dieses Vorwissen nicht, so kann die Anfrage mit „LIKE“-Anweisungen unscharf formuliert werden.

5 Von der Benutzer-Absicht zum zufriedenen Anwender

5.1 Übersetzen einer teilweise definierten Aktion in eine vollständige Aktion

In diesem Abschnitt wird der Algorithmus formalisiert, mit dem aus einer teilweise definierten Aktion eine vollständige Aktion berechnet werden kann (vgl. Abb. 2, Seite 9). Eine *teilweise definierte Aktion* ist dabei die formale Repräsentation einer Benutzer-Absicht, eine *vollständige Aktion* wird durch einen zusammenhängenden Dienstgraphen dargestellt, der die beste Kombination aus Diensten ist, die der erfassten Absicht des Benutzers entspricht. Der Übersetzungsprozess selbst besteht aus drei Schritten:

1. Übersetzen der Benutzereingabe (gegeben in PsaQL) in eine interne Repräsentation einer teilweise definierten Aktion.
2. Erweitern der teilweise definierten Aktion zu einem Aktionsgraphen. Dazu werden Informationen über vorhandene Dienste, den Kontext und die Ausführungsgeschichte sowie heuristische Strategien zu Hilfe genommen.
3. Auswählen der besten Lösung als vollständige Aktion.

Diese Vorgehensweise lässt sich mathematisch-formal wie folgt beschreiben:[4]

Definition 1 (Teilweise definierte Aktion) *Sei B die Menge aller Basen, S die Menge der möglichen Diensttypen und $S(b)$ die Menge der Dienste, die auf der Basis b zur Verfügung stehen, wobei $S(b) = S$, falls $b = \perp$.*[5] *Eine* teilweise definierte Aktion *p, die formale Beschreibung einer Benutzer-Absicht, kann durch eine Liste von Paaren modelliert werden:*

$$p = (e_1, \cdots, e_n) \mid e_i = (b_i, s_i) \text{ mit } b_i \in \{\perp\} \cup B; s_i \in \{\perp\} \cup S(b_i) \quad \forall i : (b_i \neq \perp) \vee (s_i \neq \perp) \tag{1}$$

Definition 2 (Dienstgraph) *Sei $E = \{\epsilon = (\beta, \sigma) \mid \beta \in B, \sigma \in S(\beta)\}$ die Menge der in einer pervasiven Dienstumgebung verfügbaren Dienste und $I(E) \subseteq E \times E$ die Menge der möglichen Dienst-Interaktionen in dieser Umgebung.*[6] *Dann können wir als Dienstgraph einen Teil $\mathcal{E}$ der pervasiven Dienstumgebung definieren durch*

$$G_{\mathcal{E}} = (\mathcal{E}, \mathcal{I});\ \mathcal{E} \subseteq E;\ \mathcal{I} \subseteq I(\mathcal{E}) \tag{2}$$

Die Menge Γ_E aller in E gültigen Dienstgraphen ist definiert durch:

$$\Gamma_E = \{(\mathcal{E}, \mathcal{I}) \mid (\mathcal{E} \subseteq E, \mathcal{I} \subseteq I(\mathcal{E}))\} \tag{3}$$

Definition 3 (Zusammenhängender Dienstgraph) *Ein Dienstgraph $g = (\mathcal{E}, \mathcal{I});\ \mathcal{I} \subseteq I(\mathcal{E})$ wird genau dann als* zusammenhängend *bezeichnet, wenn*

$$\forall \epsilon_0, \epsilon_n \in \mathcal{E},\ \epsilon_0 \neq \epsilon_n : (\epsilon_0, \epsilon_n) \in \mathcal{I} \vee (\exists \epsilon_1, \ldots, \epsilon_{n-1} : (\epsilon_i, \epsilon_{i+1}) \in \mathcal{I};\ (i = 0, 1, \ldots, n-1)) \tag{4}$$

[4] Zur Vereinfachung der Beschreibung werden die in PsaQL definierten Dienst-Attribute hier nicht behandelt. Sie lassen sich jedoch später einfach in das Modell einfügen.

[5] $\perp$ steht für „nicht definiert“.

[6] Die Frage der Interoperabilität zweier Dienste wird in dieser Arbeit nicht behandelt. Wir nehmen an, dass $(\epsilon_1, \epsilon_2) \in I(E) \Leftrightarrow ((\epsilon_1, \epsilon_2) \in E \times E) \wedge (\epsilon_1$ ist mit ϵ_2 interoperabel).

Definition 4 (Lösung) *Ein Graph $g_p^E = (\mathcal{E}, \mathcal{I})$; $\mathcal{E} \subseteq E, \mathcal{I} \subseteq I(\mathcal{E})$ wird genau dann* Lösung *einer teilweise definierten Aktion p in einer pervasiven Dienstumgebung E genannt, wenn*

$$g_p^E \in \Gamma_E \tag{5}$$

$$g_p^E \textit{ ist zusammenhängend} \tag{6}$$

$$\forall e = (b, s) \in p \; \exists \epsilon = (\beta, \sigma) \in \mathcal{E} \textit{ mit } \begin{cases} \beta = b & \textit{if } s = \perp, \\ \sigma = s & \textit{if } b = \perp, \\ (\beta = b) \wedge (\sigma = s) & \textit{sonst.} \end{cases} \tag{7}$$

Definition 5 (Aktionsgraph) *Sei $\mathcal{S}_p^E$ die Menge aller Lösungen für p in einer pervasiven Dienstumgebung E. Ein* Aktionsgraph $A_p^E \in \Gamma_E$ *einer teilweise definierten Aktion p in der pervasiven Dienstumgebung E ist ein zusammenhängender Graph, der alle Lösungen der teilweise definierten Aktion p enthält:*[7]

$$A_p^E = (\bigcup\nolimits_{(\mathcal{E},\mathcal{I}) \in \mathcal{S}_p^E} \mathcal{E}, \bigcup\nolimits_{(\mathcal{E},\mathcal{I}) \in \mathcal{S}_p^E} \mathcal{I}\ ) \tag{8}$$

Definition 6 (Vollständige Aktion) *Sei $C(g, \gamma)$ die Funktion, welche die Kosten einer Lösung g bei deren Ausführung in einem Kontext γ berechnet. Damit lässt sich die* vollständige Aktion c_p^E *zu einer gegebenen teilweise definierten Aktion p in einer pervasiven Dienstumgebung E definieren als:*

$$C(c_p^E, \gamma) = \min\{C(g_p^E, \gamma) \mid g_p^E \in \mathcal{S}_p^E\} \tag{9}$$

Es sei $H(p, E, \gamma)$ die Funktion, die aus der Ausführungs-Geschichte zu einer gegebene teilweise definierten Aktion p und einer pervasiven Dienstumgebung E in einem Kontext γ die zugehörige vollständige Aktion ermittelt. Mit den angebenenen Definitionen beschreiben wir nun im Algorithmus 1 (Seite 14), wie eine Benutzer-Absicht p in eine vollständige Aktion c_p^E überführt wird.[8] Ausgehend vom Aktionsgraph (der eine gültige Lösung für p darstellt) entfernt dieser Algorithmus nach und nach die Verbindungen zwischen den Diensten (siehe Zeile 11 des Algorithmus 1). Besitzt ein abgeleiteter Dienstgraph nun keine Nachfolger mehr, die gültige Lösungen darstellen würden (Zeile 16), so ist dieser Dienstgraph ein Kandidat für die vollständige Lösung (Zeile 17). Der Kandidat mit den geringsten Kosten wird als Lösung des Algorithmus ausgewählt.

Dieser Algorithmus besitzt eine exponentielle Komplexität, ermittelt aber für eine gegebene teilweise definierte Aktion stets die optimale vollständige Aktion. Um eine größere Skalierbarkeit zu erreichen, ist es sinnvoll, strengere Rahmenbedingungen zu definieren oder heuristische Ansätze zu verwenden.

Ein solcher heuristischer Ansatz (Algorithmus 2, Seite 14), mit dem sich das Problem in polynominaler Zeit lösen lässt, basiert auf folgender Knotenfärbung:

- *Schwarze Knoten* ϵ_B: $\forall \epsilon_B = (\beta, \sigma) : \exists (\beta, \sigma) \in p$
- *Rote Knoten* ϵ_R: $\forall \epsilon_R = (\beta, \sigma) : \exists (\beta, \perp) \in p \vee \exists (\perp, \sigma) \in p$
- *Weiße Knoten* ϵ_W: sonstige Knoten

[7] In einigen sehr seltenen Fällen (wenn $\forall (b, s) \in p : \; b = \perp$) kann es sein, dass A_p^E nicht eindeutig ist. In diesem Fall muss der Algorithmus 1 (vgl. weiter unten) für jedes $(A_p^E)_i$ ausgeführt werden.

[8] A_p^E kann direkt aus $(E, I(E))$ abgeleitet werden.

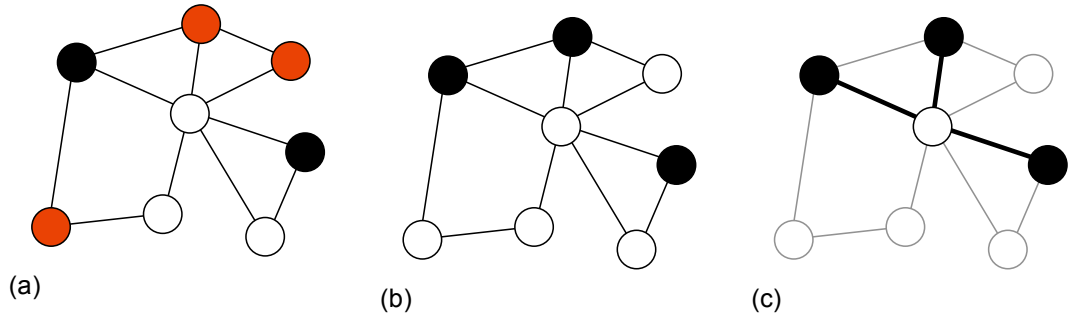

Abbildung 4: In diesem Beispiel enthält die teilweise definierte Aktion p drei Elemente: Zwei definieren eindeutig einen Koten im Aktionsgraph (schwarz), ein Element passt zu mehreren (roten) Knoten im Graphen (a). Von den roten Knoten wird nun derjenige ausgewählt, der zu den schwarzen Knoten den geringsten Abstand hat, und schwarz gefärbt, die übrigen Knoten weiß (b). Zum Abschluss wird der Minimum Cost Spanning Tree berechnet, der alle schwarzen Knoten enthält (c).

Wir nehmen an, dass jeder Knoten nur einfach gefärbt ist und eine Hierarchie durch $Schwarz > Rot > Weiß$ gegeben wurde. Somit wird E farblich partitioniert.

Die gesuchte Lösung enthält nun alle schwarzen Knoten, einige rote Knoten (so dass für jedes $e \in p$ in der Lösung ein Knoten vorhanden ist) und unter Umständen ein paar weiße Knoten. Mit der folgenden Heuristik wird versucht, die „besten“ roten Knoten auszuwählen (vgl. auch Abb. 4):

Zunächst wird mit Dijkstras Shortest Path Algorithmus [Man89, p. 204] für jeden roten Knoten ϵ_R der Abstand zu allen schwarzen Knoten ϵ_B berechnet, wobei eine modifizierte Kostenfunktion $C_I((\epsilon_R, \epsilon_B), \gamma)$ zum Einsatz kommt. Ausgehend vom roten Knoten mit dem geringsten berechneten Abstand werden nun in der Reihenfolge des berechneten Abstandes alle roten Knoten untersucht: Existiert zu einem Knoten ein Teil e der teilweise definierten Aktion p, der noch nicht von einem schwarzen Knoten abgedeckt wurde, so wird dieser Knoten ebenfalls schwarz gefärbt, ansonsten weiß. Wenn p vollständig überdeckt ist, werden alle übrigen Knoten weiß gefärbt. Mit Hilfe des Minimum Cost Spanning Tree Algorithmus [Man89, p. 208] wird aus $(E, I(E))$ der Graph, der alle schwarze Knoten enthält, als vollständige Aktion ausgewählt.

Algorithm 1 Vollständiger Übersetzungs-Algorithmus

1: $c = H(p, E, \gamma)$ *// In der Ausführungsgeschichte nach einer Lösung suchen*
2: **if** $c \neq \perp$
3: $c_p^E = c$
4: **else**
5: fifo A, fifo S
6: push(A,A_p^E)
7: **while** A $\neq \perp$
8: $(\mathcal{E}, \mathcal{I}) = g$ = shift(A) *// Ein Dienstgraph wird der Liste am Anfang entnommen*
9: is_leaf = **true**
10: **for each** $\iota \in \mathcal{I}$
11: $\mathcal{I}' = \mathcal{I} - \{\iota\}$ *// Entferne die Verbindung ι aus $\mathcal{I}$*
12: $\mathcal{E}' = \bigcup_{(\epsilon_1,\epsilon_2)\in\mathcal{I}'}\{(\epsilon_1, \epsilon_2)\}$ *// Nur zusammenhängende Elemente übernehmen*
13: **if** $g' = (\mathcal{E}', \mathcal{I}')$ ist eine Lösung für p
14: push(A,g') *// Nachfolger sichern*
15: is_leaf = **false**
16: **if** is_leaf
17: push(S,g) *// g ist Kandidat für die vollständige Aktion*
18: g = shift(S)
19: **for each** $g' \in$ S *// Die Lösung mit den geringsten Kosten ermitteln*
20: **if** $C(g', \gamma) < C(g, \gamma)$
21: $g = g'$
22: $c_p^E = g$

Algorithm 2 Heuristischer Übersetzungs-Algorithmus

1: $c = H(p, E, \gamma)$ *// In der Ausführungsgeschichte nach einer Lösung suchen*
2: **if** $c \neq \perp$
3: $c_p^E = c$
4: **else**
5: list F *// schwarze Knoten*
6: list R *// rote Knoten*
7: **for each** $\epsilon = (\beta, \sigma) \in E$
8: **if** $\exists e = (b.s) \in p : (b = \beta) \wedge (s = \sigma)$
9: push(F, ϵ)
10: **else if** $\exists e = (b.s) \in p : ((b = \beta) \wedge s = \perp) \vee ((s = \sigma) \wedge b = \perp)$
11: push(R, ϵ)
12: array D *// Minimalen Abstand berechnen*
13: **for each** $\epsilon_f \in F$
14: **for each** $\epsilon_r \in R$
15: d = length(dijkstra($(E, I(E))$,ϵ_f,ϵ_r))
16: **if** $d < D[\epsilon_r]$
17: $D[\epsilon_r] = d$
// Elemente der teilweise definierten Aktion, die nur von roten Knoten überdeckt werden:
18: $p' = \{e = (b, s) \in p \mid (b = (\perp) \vee (s = \perp) \wedge (\nexists \epsilon_f = (\beta, \sigma) \in F : (b' = \beta) \vee (s' = \sigma))\}$
19: sort(R,D) *// Sortiere die roten Knoten nach ihrem Abstand*
20: **for each** $\epsilon_R = (\beta, \sigma) \in R$
21: **if** $\exists e = (b, s) \in p' : (b = \beta) \vee (s = \sigma)$
22: push(F, ϵ_r) *// Den Knoten schwarz färben*
// Überdeckte Teile aus der teilweise definierten Aktion entfernen:
23: $p' = p' - \{e' = (b', s') \in p' \mid (b' = \beta) \vee (s' = \sigma)\}$
24: c_p^E = MCST($(E, I(E))$,F) *// Berechnung des Minimum Cost Spanning Tree über schwarze Knoten*

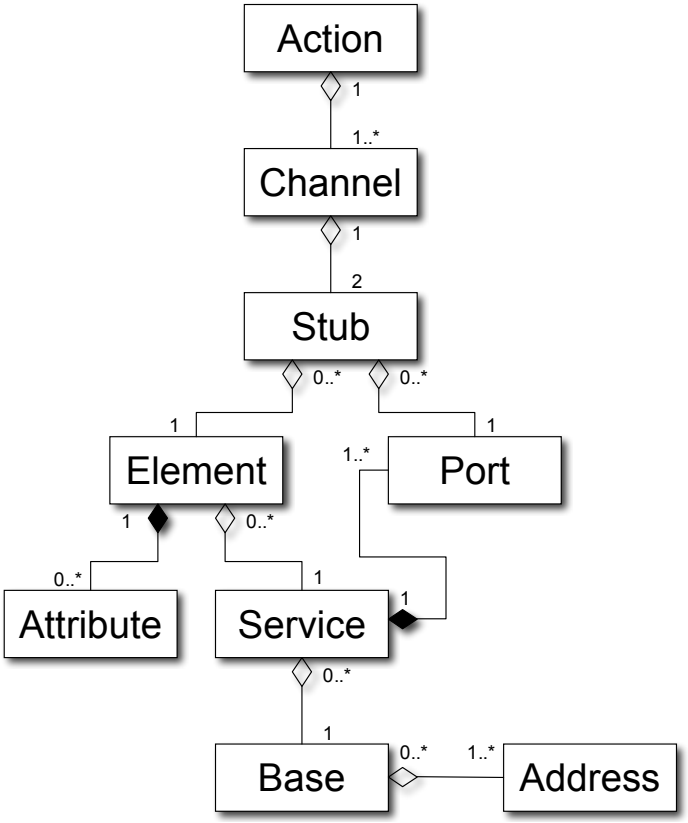

Abbildung 5: Klassendiagramm der internen Datenstruktur für die Repräsentation von Aktionsgraphen und vollständigen Aktionen.

5.2 Modellierung und Implementierung von Aktionsgraphen

Um Aktionsgraphen und vollständige Aktionen intern zu repräsentieren, schlagen wir ein objektorientiertes Datenmodell vor. Das zugehörige Klassendiagramm ist in Abb. 5 wiedergegeben: Die Modellierung der Graphen erfolgt als Menge von Kanälen, welche die Ausgangs- und die Eingangsströme der Dienste miteinander verbinden.

5.3 Darstellung von vollständig definierten Aktionen

Nachdem auf Basis der Äußerung der Absicht eines Benutzers mit Wissen um den Ausführungskontext und die Ausführungshistorie eine vollständige Aktion berechnet wurde, muss diese auch ausgeführt werden. Die Steuerung der Ausführung kann durch eine unabhängige Programmeinheit oder sogar auf einem ganz anderen Gerät erfolgen, weswegen wir ein systemunabhängiges *XML-Format* definiert haben, welches Aktionsgraphen und vollständige Aktionen repräsentiert. Diese Datei enthält sämtliche zur Ausführung notwendigen Daten, man ist also zu diesem Zeitpunkt nicht mehr auf das Vorhandensein einer Dienstbeschreibungs-Datenbank angewiesen.

5.4 Bewertungskriterien in einer pervasiven Dienstumgebung

Damit sich ein pervasives System vom Anwender unbemerkt in seine Umgebung integrieren kann, ist es notwendig, dass dieser von der Äußerung seiner Absicht bis hin zur Ausführung der dazugehörigen Funktion praktisch keine Verzögerung bemerkt. Diese Ausführungsgeschwindigkeit ist zwar ein wichtiger Faktor, der die Zufriedenheit des Anwenders beeinflusst, aber nicht der einzige. Insgesamt haben wir folgende Bereiche als Grundlagen für die Bewertung eines Algorithmus, der eine Benutzer-Absicht in eine vollständige Aktion übersetzt, herausgearbeitet:

- Die *Zufriedenheit des Anwenders* als wichtigstes Ziel beim Entwurf pervasiver Systeme.
- Die *Ausführungszeit des Algorithmus*, da sie den größten Einfluss auf die Zufriedenheit des Anwenders hat.
- Die *Geschwindigkeit der Datenübertragung* im Netzwerk, denn sie beeinflusst direkt die Ausführungszeit des Algorithmus. Neben der *Geschwindigkeit des Netzwerks*, die nur selten vom Entwickler der Middleware direkt beeinflusst werden kann, wird dieser Faktor hauptsächlich von der *Größe der übertragenen Dienstbeschreibungs-Daten* und die *Entfernung*, über die die Daten übermittelt werden, bestimmt.
- Die *Skalierbarkeit* des Algorithmus, wenn sich die *Anzahl der verfügbaren Dienste* oder die *Größe der Anfrage* erhöht.
- *Einschränkungen pervasiver Systeme*, wie zum Beispiel ein *verantwortungsbewusster Umgang mit CPU- und Speicher-Ressourcen.*

Die Art und Weise, wie ein Algorithmus die Beschreibungen der vorhandenen Dienste in seine lokale Datenbank transferiert, um auf dieser Basis eine vollständige Aktion zu berechnen, hat einen direkten Einfluss auf die Ausführungszeit des Algorithmus. Verschiedene Möglichkeiten wurden hier mit Hilfe eines Prototypen untersucht, die Ergebnisse sind im folgenden Abschnitt zusammengefasst.

6 Ergebnisse der Performanz-Messungen

Um die Performanz unseres Prototypen beispielhaft zu messen, untersuchten wir die Größe und Distanz der übertragenen Daten, während die Netzwerkgröße und die Länge der Anfrage verändert wurden. Zur Bestimmung der Skalierbarkeit führten wir einen Skalierbarkeits-Faktor ein, der die mittlere Entfernung der übertragenen Daten darstellt:

$$f_{scal} = \frac{\sum_{msg} size(msg) * distance(msg)}{\sum_{msg} size(msg)}$$

Abbildung 6 zeigt die Ergebnisse der Performanz-Messung bei Veränderung der Netzwerk-Größe von 10 bis 1000 Basen.[9] „Algorithmus A“ bezeichnet hier eine Implementierung des in Abschnitt 5.1 eingeführten vollständigen Algorithmus. Bei „Algorithmus B“ wurde die Implementierung bezüglich der Skalierbarkeit verbessert: Dienstbeschreibungen wurden nur noch bei Bedarf und „on-the-fly“ abgerufen, die Verknüpfung der Dienste auf eine lineare Verkettung beschränkt.

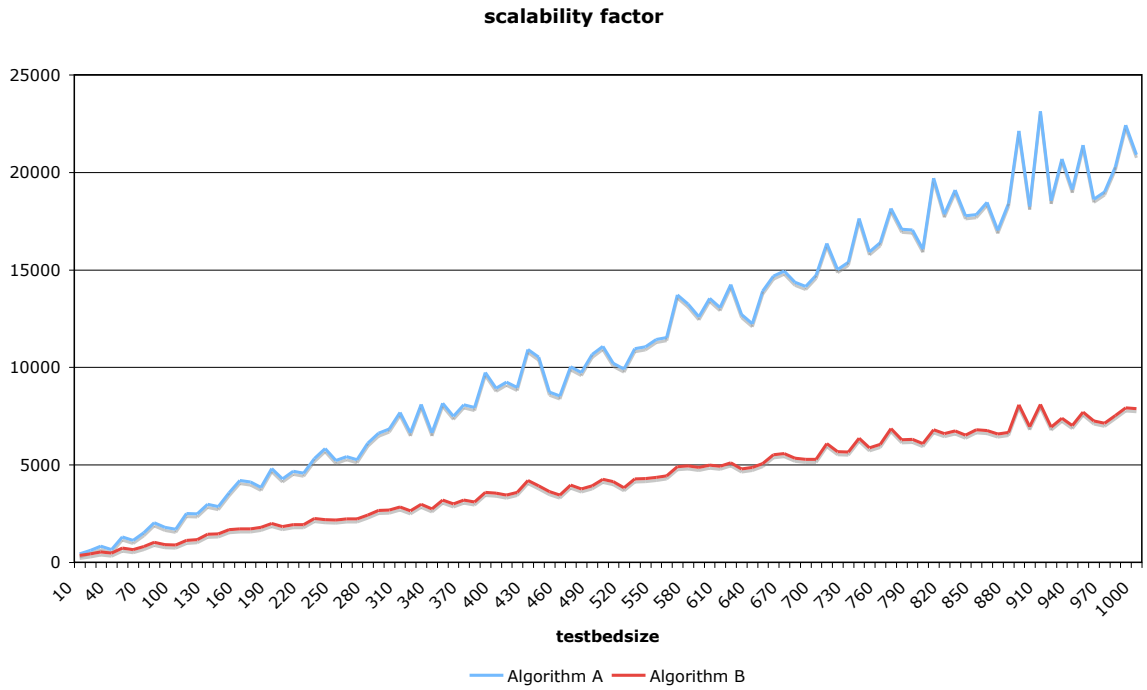

Abbildung 6: Die Entwicklung des Skalierbarkeits-Faktors (siehe Text) bei Veränderung der Testbett-Größe.

[9] Zur Generierung der simulierten Netzwerke wurde der im Rahmen dieser Arbeit entwickelte PlnGen-Algorithmus verwendet, eine Beschreibung dieses Algorithmus findet sich unter `http://liris.wh4f.de/pln_gen_algo.html`.

7 Schlussbemerkungen und offene Fragen

7.1 Zusammenfassung

In dieser Arbeit wurde eine Strategie vorgestellt, wie in einer pervasiven Dienstumgebung beim Verknüpfen von Diensten zu Aktionen die Absicht des Anwenders berücksichtigt werden kann. Mit PsaQL wurde die nach unserem Kenntnisstand erste formale Sprache zur Beschreibung von Benutzer-Absichten in pervasiven Dienstumgebungen entworfen. Zusätzlich wurden Algorithmen eingeführt, die diese teilweise definierten Aktionen mit Hilfe von Dienstbeschreibungen, Kontext-Informationen und einer Ausführungs-Geschichte in ausführbare Dienstgraphen, d. h. vollständige Aktionen übersetzen. Zur Implementierung der Algorithmen präsentierten wir für Dienstgraphen ein objekt-orientiertes Datenmodell sowie ein XML-Austauschformat.

Zum Abschluss der Arbeit wurden einige Metriken zur Performanz-Messung für Implementierungen der vorgestellten Algorithmen herausgearbeitet. Dabei steht die Zufriedenheit des Benutzers an erster Stelle, hauptsächlich beeinflusst von der Ausführungszeit der Algorithmen, die wiederum von der Skalierbarkeit der Implementierung abhängt. Anhand eines im Rahmen dieser Arbeit entwickelten Prototypen wurde die Skalierbarkeit beispielhaft ermittelt, indem die Größe des simulierten Netzwerkes sowie die Länge der Anfrage verändert und dabei die Menge und Distanz der übertragenen Daten gemessen wurde. Die durchgeführten Benchmarks zeigten deutlich, wie stark die Skalierbarkeit von der konkreten Implementierung abhängt (vgl. Abb. 6).

7.2 Forschungsbeiträge dieser Arbeit

- Einführung und Modellierung der Konzepte der *teilweise definierten Aktion*, des *Aktionsgraphen* und der *vollständigen Aktion* im Rahmen von PerSE, einer nicht-zentralisierten pervasiven Dienstumgebung.
- Entwicklung von *PsaQL*, nach unserem Wissen die erste formale Sprache, mit der sich intuitiv und flexibel Anfragen für Dienstaktionen in einer pervasiven Dienstumgebung ausdrücken lassen.
- Entwurf von *Algorithmen*, die eine formalisierte Benutzer-Absicht in eine ausführbare vollständige Aktion überführen, indem sie auf Dienstbeschreibungen, Kontext-Informationen und eine Ausführungs-Geschichte zurückgreifen.
- Entwicklung von Vorschlägen zur *Implementierung* solcher Transformations-Algorithmen, indem ein objekt-orientiertes Datenmodell zur Beschreibung von Dienstgraphen und vollständigen Aktionen eingeführt wurde und ein dazugehöriges *XML-Datenaustauschformat*.
- Überlegungen zu *Performanz-Messungen* und ihre Durchführung an einer *Prototyp-Implementierung*.

7.3 Perspektiven

Im Anschluss an diese Arbeit bleiben weitere Punkte zu erforschen. Benutzer-Intentionen direkt in PsaQL auszudrücken wird wohl nicht der Weisheit letzter Schluss bleiben, sondern andere, unaufdringlichere Interaktionsformen sind zu entwickeln. Unabhängig davon aber, wie der Benutzer seine Absicht ausdrückt, wird PsaQL als einfache Schnittstelle zwischen den verschiedenen Programm-Modulen weiter existieren. Auch wenn die Sprache in Zukunft noch weiterentwickelt werden kann, so bleibt es wichtig, ihre Einfachheit zu erhalten.

Die Verwendung von Kontext-Informationen im Aktions-Auswahlprozess von PerSE ist momentan noch in den Anfängen, aber die Arbeiten von Ejigu et al. [ESB05] in unserer Forschungsgruppe werden dieses Thema weiter vertiefen. Ebenso wird auch der Aspekt der Ausführungs-Geschichte näher untersucht werden müssen, da unseren Schätzungen zufolge über 90 % der Aktionen wiederverwendet werden können und nur ca. 10 % neu zu berechnen sind. Dies wird sich in einer bedeutenden Geschwindigkeitssteigerung widerspiegeln.

Einen Hauptaspekt anschließender Forschungsarbeit wird das Thema Sicherheit in pervasiven Dienstumgebungen darstellen. Momentan gehen wir von der Voraussetzung aus, dass jeder Dienst von jedem Anwender verwendet werden kann, dies wird so aber nicht haltbar bleiben.

Unabhängig von den Forschungsarbeiten, die noch ausstehen, bis pervasive Systeme aus dem Forschungslabor in die „wirkliche Welt“ wandern und so tagtägliche Realität werden, ist mit dieser Arbeit doch ein weiterer Schritt in diese Richtung getan. Durch den vorgestellten Übergang von einer diffus geäußerten Absicht des Anwenders hin zu einer ausführbaren Aktion ist es möglich, das PerSE zu dem unsichtbaren System wird, welches dem Anwender das Leben erleichtert und rechnergestütze Aufgaben zu seiner Zufriedenheit ausführt.

Glossar[10]

Bluetooth Ein Industriestandard gemäß IEEE 802.15.1 für die drahtlose Vernetzung von Geräten über kurze Distanz (10m–100m)

BNF Backus-Naur Form, eine kompakte formale Metasyntax, die verwendet wird, um kontextfreie Grammatiken darzustellen

Dienst Eine Anwendung ohne Bedienoberfläche, die ihre spezifische Funktionalität mittels einer Programmier-Schnittstelle bereitstellt

Ethernet Eine rahmenbasierte Computer-Vernetzungstechnologie für lokale Netzwerke

IEEE 802.11b/g Ein Synonym für *WLAN*

LIRIS Laboratoire d'InfoRmatique en Images et Systèmes d'information, Forschungszentrum für Bildverarbeitung und intelligente Informationssysteme, Lyon

OWL-S Web Ontology Language, ist eine XML-basierte Beschreibungssprache, um Ontologien im Internet erstellen, publizieren und verteilen zu können

PDA Personal Digital Assistant, ein kleiner tragbarer Computer

PerSE Pervasive Service Environment, eine am *LIRIS* entwickelte Platform zur Verwaltung komplexer Computersysteme, die aus unabhänigigen, vernetzten, eingebetten Geräten bestehen, die *Dienste* anbieten, vgl. Kapitel 3

Pervasives Rechnen (*engl.* pervasive computing) bezeichnet die alles durchdringende Vernetzung von „intelligenten“ Gegenständen des Alltags

PsaQL Pervasive Service Action Query Language, eine in dieser Arbeit entwickelte, formale Sprache zur Beschreibung von Anfragen an eine pervasive Dienstumgebung, vgl. Kapitel 4

SQL Structured Query Language, eine deklarative Abfragesprache für Relationale Datenbanken

Ubiquitäres Rechnen Ein Synonym für *Pervasives Rechnen*

Web Service eine Software-Anwendung, die mit einem Uniform Resource Identifier (URI) eindeutig identifizierbar ist und deren Schnittstellen als XML-Artefakte definiert, beschrieben und gefunden werden können

WiFi Eine Werbebezeichnung für *WLAN*

WLAN Wireless LAN, bezeichnet ein „drahtloses“ lokales Funknetz, wobei meist nur die letzte Verbindung zum Anwender kabellos ausgeführt ist

HTTP(S) HyperText Transfer Protocol (Secure), das hauptsächlich verwendete Kommunikationsschema, um Informationen im World Wide Web zu übertragen

XML Extensible Markup Language, ein Standard zur Erstellung textbasierter, maschinen- und menschenlesbarer Dokumente in Form einer Baumstruktur

[10] Erstellt unter Zuhilfenahme von Wikipedia, der freien Enzyklopädie (http://wikipedia.org)

Literatur

[BB03] Sven Buchholz and Thomas Buchholz, *Adaptive content networking*, ISICT '03: Proceedings of the 1st international symposium on Information and communication technologies, Trinity College Dublin, 2003, pp. 213–219.

[Bra05] Ingo Braun, *Semantic web services: A logic-based framework to dynamic and approximate composition by constraints*, Master-Thesis INSA de Lyon, unpublished, June 2005.

[EKvBS04] Khalil El-Khatib, Gregor v. Bochmann, and Abdulmotaleb El Saddik, *A distributed content adaptation framework for content distribution networks (online: http://beethoven.site.uottawa.ca/dsrg/PublicDocuments/Publications/ElKh04c.pdf)*, 2004, last checked: 2005-06-15.

[ESB05] Dejene Ejigu, Vasile-Marian Scuturici, and Lionel Brunie, *Neighbourhood based architecture for context-aware platform in pervasive computing*, unpublished, 2005.

[GSSS02] David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste, *Project aura: Toward distraction-free pervasive computing*, IEEE Pervasive Computing **1** (2002), no. 2, 22–31.

[GWvB+01] Steven D. Gribble, Matt Welsh, Rob von Behren, Eric A. Brewer, David Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R. H. Katz, Z. M. Mao, S. Ross, B. Zhao, and Robert C. Holte, *The ninja architecture for robust internet-scale systems and services*, IEEE Intelligent Systems, vol. 35, Elsevier North-Holland, Inc., New York, NY, USA, March 2001, pp. 473–497.

[Man89] Udi Manber, *Introduction to algorithms: A creative approach*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[Mos03] Soraya Kouadri Mostéfaoui, *A Context-Based Services and Discovery and Composition Framework for Wireless Environments*, Proc. of the 3rd IASTED International Conference on Wireless and Optical Communications (WOC'2003) (Banff, Canada), 14-16 July 2003, pp. 637–642.

[MSZ01] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng, *Semantic web services*, IEEE Intelligent Systems, vol. 16, IEEE Educational Activities Department, March 2001, pp. 56–53.

[MvH04] Deborah L. McGuinness and Frank van Harmelen, *OWL Web Ontology Language Overview (online: http://www.w3.org/TR/2004/REC-owl-features-20040210/)*, 2004-02-10, last checked: 2005-06-15.

[MYA+05] Jianhua Ma, Laurence T. Yang, Bernady O. Apduhan, Runhe Hunag, Leonard Barolli, and Mokoto Takizawa, *Towards a smart world and ubiquitous intelligence: A walktrough from smart things to smart hyperspaces and ubickids*, International Journal of Pervasive Computing and Communications, vol. 1, Troubador Publishing Ltd., March 2005, pp. 53–68.

[NSDM03] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello, *A system for principled matchmaking in an electronic marketplace.*, WWW, 2003, pp. 321–330.

[PM94] Claudia Popien and Bernd Meyer, *A service request description language*, FORTE'94, Chapman & Hall, Bern, January 1994, pp. 14–32.

[RC03] Anand Ranganathan and Roy H. Campbell, *An infrastructure for context-awareness based on first order logic*, Personal and Ubiquitous Computing, vol. 7, Springer-Verlag London Ltd, December 2003, pp. 353–364.

[RHC+02] Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganat, Roy H. Campbell, and Klara Nahrstedt, *Gaia: A middleware infrastructure to enable active spaces*, IEEE Pervasive Computing (2002), 74–83.

[SdF03] Mithun Sheshagiri, Marie desJardins, and Tim Finin, *A planner for composing services described in daml-s*, Proceedings of ICAPS'03 Workshop on Planning for Web Services, June 2003.

[SG03] João Pedro Sousa and David Garlan, *Improving user-awareness by factoring it out of applications*, UbiSys'03 - System Support for Ubiquitous Computing Workshop, October 12 2003.

[SPAS03] Katia P. Sycara, Massimo Paolucci, Anupriya Ankolekar, and Naveen Srinivasan, *Automated discovery, interaction and composition of semantic web services.*, J. Web Sem. **1** (2003), no. 1, 27–46.

[SPP+03] Umar Saif, Hubert Pham, Justin Mazzola Paluska, Jason Waterman, Chris Terman, and Steve Ward, *A case for goal-oriented programming semantics*, UbiSys'03 - System Support for Ubiquitous Computing Workshop, October 12 2003.

[SvdAB+03] Steffen Staab, Wil M. P. van der Aalst, V. Richard Benjamins, Amit P. Sheth, John A. Miller, Christoph Bussler, Alexander Maedche, Dieter Fensel, and Dennis Gannon, *Web services: Been there, done that?*, IEEE Intelligent Systems **18** (2003), no. 1, 72–85.

[VR04] Maja Vukovic and Peter Robinson, *Adaptive, planning-based, web service composition for context awareness*, Second International Conference on Pervasive Computing, April 2004.

[VRV05] Mathieu Vallée, Fano Ramparany, and Laurent Vercouter, *Composition flexible de services d'objets communicants*, UBIMOB 05, Mai 31 - June 3 2005.

A Konferenzbeiträge für IFIP EUC'2005 und IEEE SITIS-05

Die wichtigsten Forschungsergebnisse dieser Arbeit wurden als Konferenzbeiträge der Arbeitsgruppe des LIRIS Lyon bei der *2005 IFIP International Conference on Embedded And Ubiquitous Computing (EUC'2005, Nagasaki, Japan)* und der *2005 IEEE International Conference on SIGNAL-IMAGE TECHNOLOGY and INTERNET- BASED SYSTEMS (SITIS-05, Yaoundé Cameroon)* eingereicht und für beide Konferenzen angenommen. Auf den folgenden Seiten ist das Paper, das im Konferenzband der EUC'2005 veröffentlicht wird, beispielhaft wiedergegeben.