Die Anpassung von Softwaresystemen - nach Abschluss der initialen Entwicklung - ist mit hohem Aufwand verbunden. Domänenexperten, die mit einer Software arbeiten, können Anpassungswünsche aufgrund fehlender softwaretechnischer Kenntnisse nur auf fachlicher Ebene formulieren. Softwareentwickler besitzen hingegen softwaretechnisches Wissen, kennen sich jedoch nicht unmittelbar in der Domäne aus. Dieses „User-Programmer Gap“ genannte Phänomen erfordert in der regulären Softwareentwicklung Übersetzer, bspw. Anforderungsingenieure und Softwarearchitekten, durch welche sich Verzögerungen und hohe Anpassungskosten ergeben.

Diese Arbeit untersucht die Fragestellung, ob und wie fachlich versierte Anwender ohne softwaretechnische Kenntnisse in die Lage versetzt werden können, dynamische Abläufe eines Softwaresystems selbstständig anzupassen. Als Anpassungsmethode wird dabei die visuelle Modellierung von Abläufen betrachtet, da sie das beste Verhältnis zwischen Schulungsaufwand und erreichbarem Anpassungsgrad erreicht.

Im Rahmen der Recherche werden Charakteristiken visueller Modellierungssprachen vorgestellt und dazu verwendet, existierende Modellierungsansätze aus den Bereichen der Softwaremodellierung, der Prozessmodellierung und der visuellen Programmiersprachen zu beschreiben. Standards, wie BPMN oder UML, aber auch akademische Ansätze werden im Hinblick auf ihre Einsetzbarkeit durch softwaretechnische Laien untersucht. Die so ermittelten Best Practices aus den Bereichen Notation, Ausführungssemantik, Werkzeuge und Integration erlauben die Formulierung von Anforderungen an einen idealen Anpassungsansatz vor dem Hintergrund dieser Arbeit. Als Ergebnis der Analyse zeigt sich, dass weder die bestehenden Standards, noch proprietäre Lösungen diese Anforderungen in Summe zufriedenstellend erfüllen.

In dieser Arbeit wird daher der ActivityLib-Ansatz vorgestellt, welcher sich insbesondere an softwaretechnische Laien als Endanwender richtet. Neben der softwaretechnischen Konzeption des Ansatzes wird in dieser Arbeit auch die Realisierung einer Referenzimplementierung beschrieben und eine Evaluation in Industrieprojekten vorgenommen.

Grundlage des Ansatzes stellt eine Modellierungssprache mit einer eigenen Notation dar. Die Notation der Modellierungssprache orientiert sich dabei an den Aktivitätsdiagrammen der UML und zeichnet sich vor allem durch eine einfache Erlernbarkeit aus. Neben der Modellierung von Kontrollflüssen stellt die Modellierung von typisierten Datenflüssen ein Kernkonzept der Modellierungssprache dar.

Der Ansatz definiert darüber hinaus eine eigene Ausführungssemantik, die trotz ihrer Einfachheit eine hohe Sprachmächtigkeit bietet. Als einzige visuelle Programmiersprache unterstützt ActivityLib alle 43 veröffentlichten Kontrollflussmuster und erlaubt zudem die gleichzeitige Verwendung imperativer und funktionaler Programmierparadigmen. Sowohl die Modellierung nebenläufiger Vorgänge als auch die Übergabe von Daten zwischen mehreren Ausführungssträngen werden unterstützt. Im Vergleich zu anderen Modellierungssprachen, die sich ebenfalls an softwaretechnische Laien wenden, stellt die durch ActivityLib erreichte Sprachmächtigkeit ein Alleinstellungsmerkmal dar.

Für Softwareentwickler, die den ActivityLib-Ansatz verwenden möchten, werden ebenfalls Konzepte entwickelt und realisiert, die den Integrationsaufwand in Drittanwendungen minimieren. Das ActivityLib-Typsystem ist bspw. eines dieser Konzepte und erlaubt die Definition und Erzeugung komplexer Datenstrukturen zur Laufzeit, ohne Programmcode schreiben zu müssen. Auf Basis der optional definierbaren Typmetadaten können automatisiert Bearbeitungsdialoge für diese Datenstrukturen abgeleitet werden. Durch Abstraktion von den Implementierungsdatentypen können Aktivitäten außerdem portabel modelliert werden.

Im Rahmen der Arbeit wird ein vollständiger Architekturentwurf für den ActivityLib-Kern durchgeführt, der die Laufzeitumgebung, das Metamodell der Aktivitäten sowie die Integrationsschicht umfasst, mit welcher sich der ActivityLib-Ansatz in Drittanwendungen integrieren lässt.

Als Werkzeug zur Bearbeitung der Aktivitäten dient der ActivityLib-Editor, der in Form einer Web-Anwendung realisiert wird. Die Architektur des Editors und dessen Erweiterungsmöglichkeiten sowie das zur Realisierung verwendete erweiterbare Web-Framework werden detailliert beschrieben. Über die Oberfläche des Editors werden sämtliche für die Modellierung, die Prüfung und den Test der Aktivitäten benötigten Funktionen in einer für softwaretechnische Laien angemessenen Form bereitgestellt.

Durch eine Referenzimplementierung konnte die Praxistauglichkeit des ActivityLib-Ansatzes im Rahmen mehrerer industrieller Softwareprojekte evaluiert werden. In der Arbeit werden die Integration in das jeweilige Softwaresystem, aber vor allem auch die Akzeptanz und Erfahrungen der Endanwender im Kontext verschiedener Domänen ausführlich beschrieben.

Der ActivityLib-Ansatz erfüllt alle an einen idealen Modellierungsansatz gestellten Anforderungen, die in dieser Arbeit formuliert werden. Die durchweg positiven Erfahrungen auf Seiten der Softwareentwickler, aber vor allem auch auf Seiten der Domänenexperten, sind Indizien für die praktische Relevanz der aufgestellten Kriterien. Sie untermauern zudem die Daseinsberechtigung des ActivityLib-Ansatzes und zeigen, dass sich der „User Programmer Gap“ mittels visueller Modellierung schließen lässt.