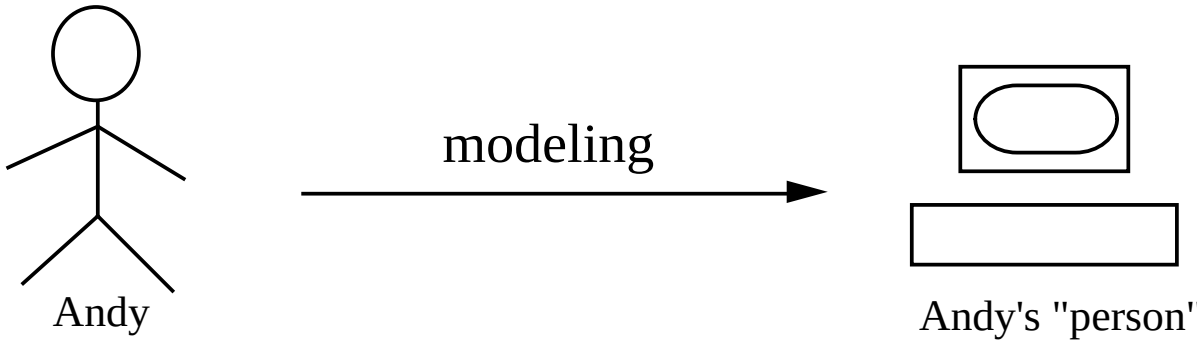
8.4 Umsetzung der Eckpfeiler in einem datenschutz-orientierten Informationssystem : DORIS

Der besseren Übersichtlichkeit wegen sind in diesem Unterkapitel Schlagworte des Datenschutzes in Avant Garde, Begriffe von DORIS in Palatino gesetzt.

0. Zuordnung zwischen Menschen der realen Welt und Akteuren des Informationssystems:
   Die Menschen werden im Informationssystem als **Person**en (*person*) modelliert.

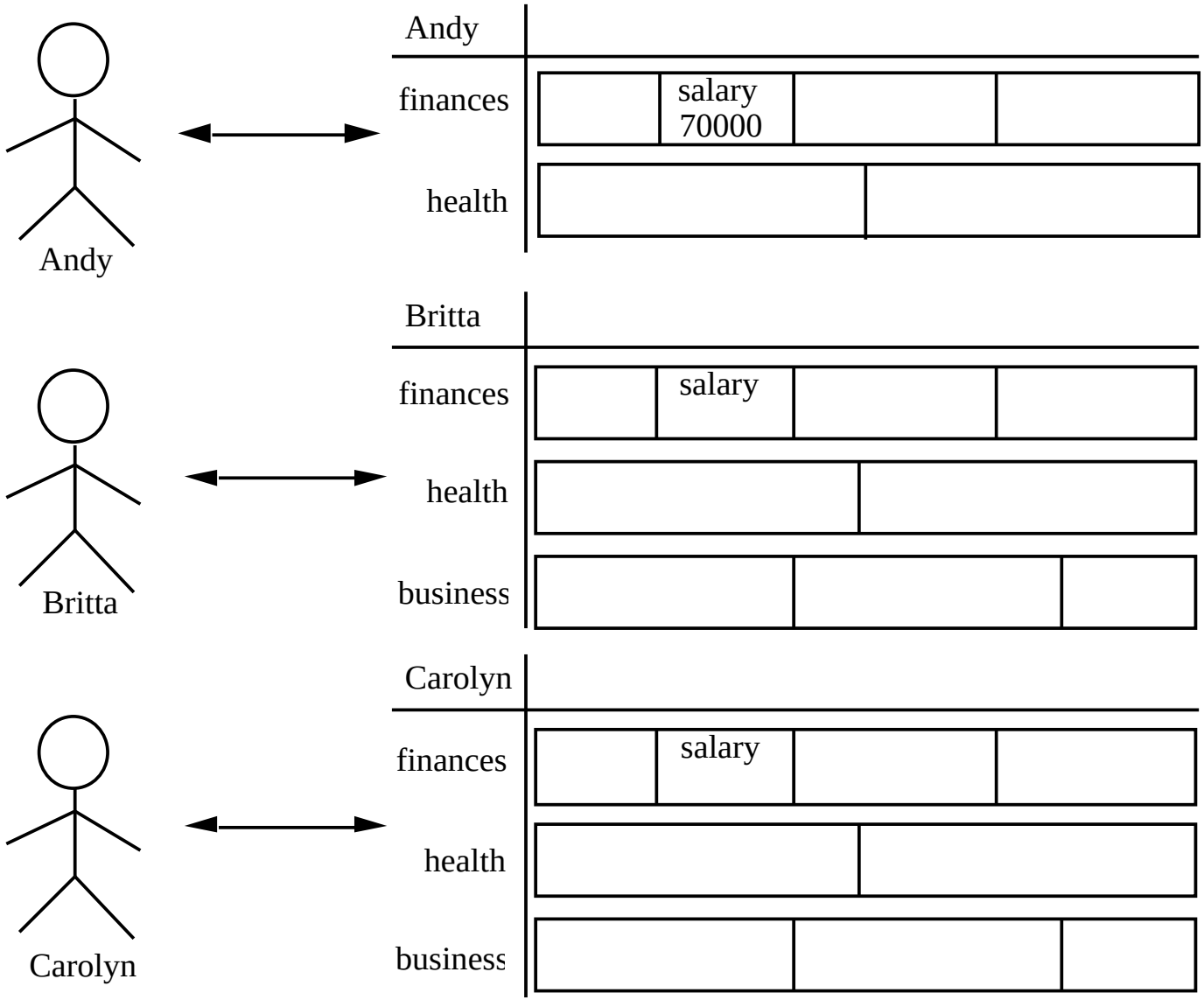
Example:



Insbesondere sind alle handelnden Menschen auch Personen des Informationssystems. Dadurch kann jede Verarbeitung von Daten Personen (und damit Menschen) zugeordnet werden.

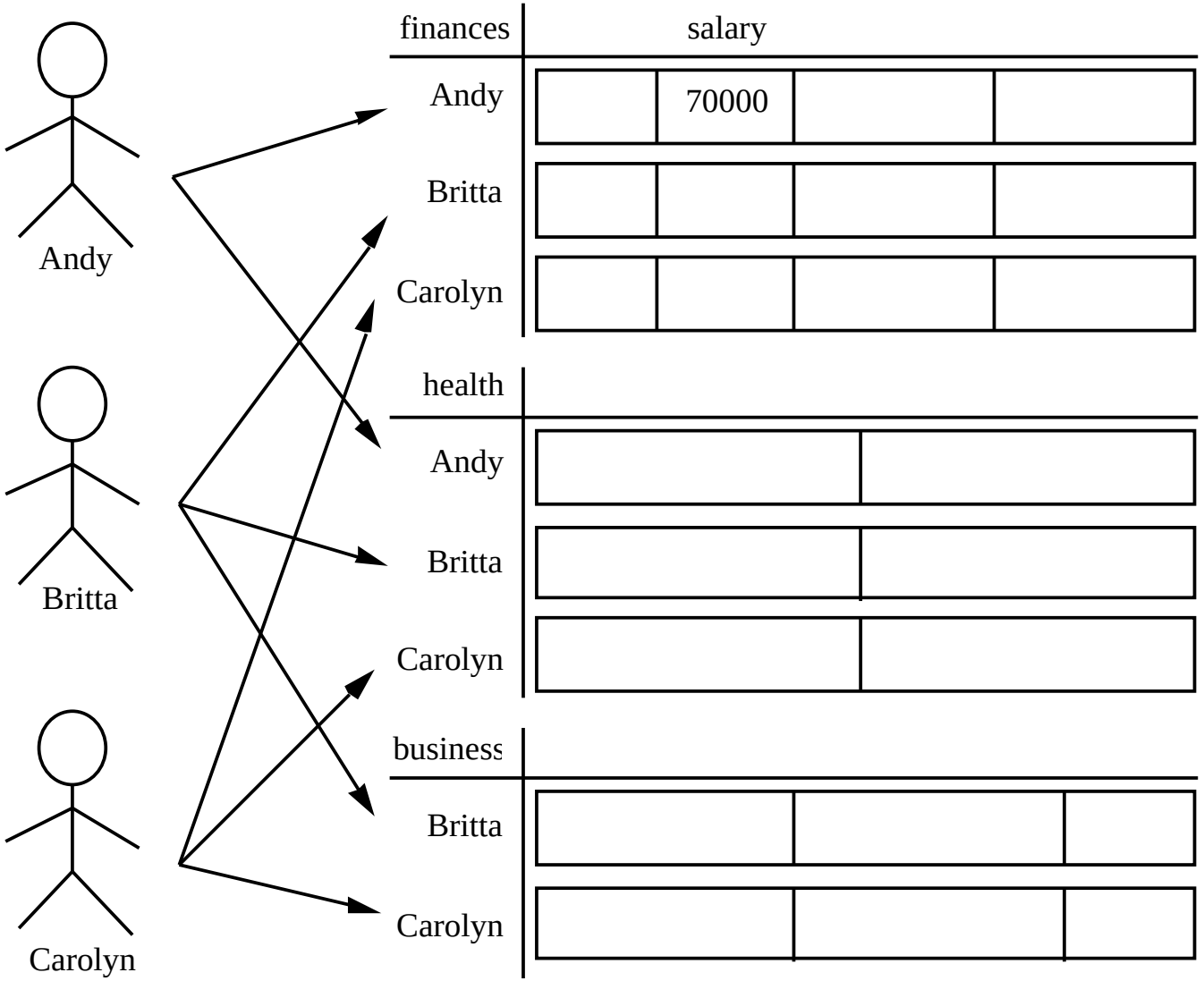
1. Personenbezogene Daten werden als "Wissen von Personen über sich selbst" modelliert:
   Die Daten eines Menschen werden zusammengefasst und abgeschirmt.

Example:



Dies ist grundsätzlich anders als in bisherigen -z.B. relationalen- Datenbanken, wo in der Regel die Daten eines Menschen ("das, was die Datenbank über den Menschen weiß") über viele verschiedene Einheiten verstreut sind.

Example:



Da das Wissen über einen Menschen durch **Datenerhebung beim Betroffenen**, also bei eben diesem Menschen gewonnen werden soll, kann das Wissen über einen Menschen nur das Wissen dieses Menschen selbst sein (bzw. ein Teil davon).

Daher wird nur das Wissen von Menschen über sich selbst modelliert; das Wissen über (andere) Menschen durch Datenerhebung bei diesen Menschen gewonnen.

Das **Auskunftsrecht** kann dann sehr einfach realisiert werden, indem man jedem Menschen das Leserecht auf seine eigenen Daten gibt ("Jeder weiß alles über sich selbst.").

Das Wissen wird -in Anlehnung an relationale Datenbanken- in **Attribute** (*attribute*) strukturiert, die ihrerseits elementar oder strukturiert (z.B. Mengen, Records) sein können.

2. Öffentlichkeit struktureller Eigenschaften:
   Es wird unterschieden zwischen **individuellen Eigenschaften** und **strukturellen Eigenschaften**.

   Strukturelle Eigenschaften beschreiben z.B. die **Art** des Wissens, individuelle Eigenschaften sind z.B. das Wissen selbst. (siehe auch 11.)

   Geschützt werden die personenbezogenen Daten als individuelle Eigenschaften, während die strukturellen Eigenschaften die Möglichkeiten der Datenverarbeitung erst durchschaubar machen und daher öffentlich sind.

Es soll so für jeden erkennbar sein, wer Daten welcher Art (in welcher Eigenschaft) speichert und was damit passieren kann (Durchschaubarkeit der Datenverarbeitung).

Damit soll insbesondere die Funktion erfüllt werden, die dem **Dateienregister** zugedacht war.

Personen, deren strukturelle Eigenschaften gleich sind, können zu einer **Gruppe** (*group*) zusammengefasst werden.

3. Den Personen stehen -u.a. zur Bearbeitung ihres Wissens- **Operationen** (*operation*) zur Verfügung.

   Diese Operationen sind abgestützt auf systemdefinierte Operationen wie z.B. read, write, insert, modify, delete, grant, revoke.

   Für typische Aufgaben einer Personengruppe können (und sollten) zusätzliche Operationen definiert werden. Operationen gehören zu den strukturellen Eigenschaften.

4. Datenerhebung durch Kommunikation mit den Betroffenen:
   Möchte man also etwas über andere Menschen wissen, so muss man diese fragen, d.h. eine Datenerhebung bei diesen Menschen durchführen, mit ihnen **kommunizieren**.

   Im realen Leben kann man jeden Menschen fragen, den man "kennt" (insbesondere: irgendwie erreicht) und er antwortet richtig, falsch oder gar nicht.

Im Informationssystem fragt man die zu einem Menschen gehörende Person.

Dabei ist in DORIS diese Entscheidungsmöglichkeit wie folgt fixiert: Hat der Anfragende geeignete Rechte, so antwortet der Befragte richtig, andernfalls gar nicht (Variante: unvollständig).

Die wichtigsten Rechte in DORIS sind **Vollmachten** und **Bekanntschaften**.

Vollmachten erlauben die Ausführung von Operationen, Bekanntschaften geben an, bzgl. welchen Personen eine Vollmacht benutzt werden darf.

5. Jede **Vollmacht** (*authority*) steht -bisher leider nur implizit- für einen **Zweck**.

Der Anfragende muss bei einer Anfrage stets eine seiner Vollmachten angeben (und so dem Befragten den Zweck bekanntgeben).

Das Anfrageergebnis wird im System nach Anzeige gelöscht.

Dadurch dass der Anfragende das Ergebnis nicht permanent speichern kann, ist die Zweckbindung (innerhalb des Systems) sichergestellt:
Die Daten werden nur einmal für den durch die Vollmacht angegebenen Zweck benutzt.

Das unmittelbare Löschen von Ergebnissen sichert trivialerweise das Einhalten einer kurzen

Löschfrist .

Ebenso wird auf diese Weise das Problem der Nachberichtspflicht gelöst.

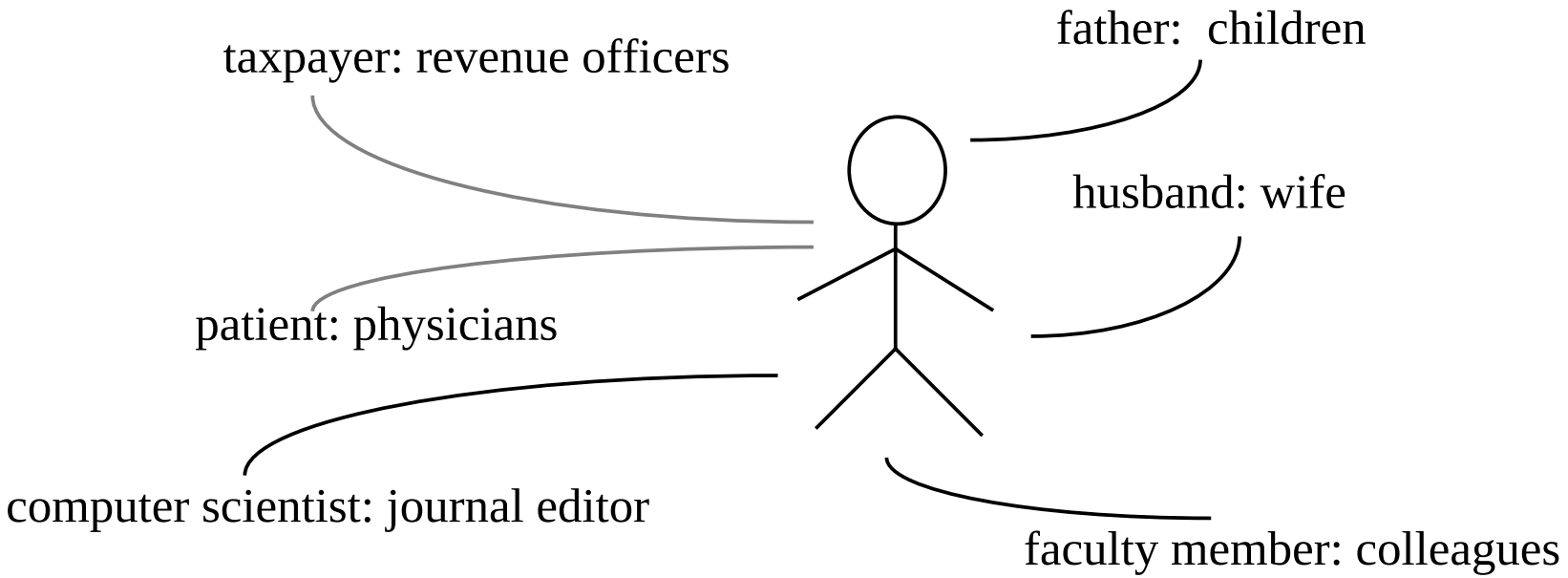
Eine Vollmacht erlaubt die Nutzung von einer Menge von Operationen bzgl. einer Personengruppe zur Bearbeitung von Anfragen bzw. Änderungen.

Z.B. benötigt man zur erfolgreichen Bearbeitung von Anfragen eine Leseoperation.

Die Operationen können eingeschränkt werden, z.B. Lesen nur auf speziellen Attributen.

Vollmachten gehören zu den strukturellen Eigenschaften.

Die Berechtigungen sind an die Vollmacht (und nicht an die Person) gebunden, damit von einer Person verschiedene Vollmachten nicht gleichzeitig genutzt werden können.
Dieses Prinzip unterstützt die Trennung verschiedener sozialer Rollen eines Menschen.



Das Verhaltensmuster einer sozialen **Rolle** (*role*) wird angenähert beschrieben durch eine Menge von Operationen und eine Personengruppe, die Ziel/Adressat dieses Verhaltensmusters ist.

Die Idee ist, dass diese Menge von Operationen das Verhalten in dieser Rolle (aber möglichst nicht mehr) möglich macht.
Rollen gehören zu den strukturellen Eigenschaften.

6. Darüberhinaus werden zur Abgrenzung von Zuständigkeit en sog. **Bekanntschaften** (*acquaintance*) eingeführt, die -auf der Ebene der Personen- festlegen, bezüglich wem man seine Vollmacht anwenden kann.

Bekanntschaften (bzgl. einer Vollmacht) enthalten nur Personen, die zu einer Gruppe gehören, für die die Vollmacht gilt.
Die Bekanntschaften gehören zu den individuellen Eigenschaften.

7. Eine Person kann das **Recht**, eine Rolle auszuüben, (also eine Vollmacht) anderen Personen (z. B. zeitlich begrenzt für Vertretungen) **verfügbar machen** (*make available*), so dass diese das Recht nutzen können, ohne es zu besitzen.

   Insbesondere kann die ein Recht verfügbar machende Person dieses Recht jederzeit wieder zurücknehmen.

   Zur Nutzung dieses Rechts muss sich die nutzende Person auf die Person beziehen, die ihr das Recht verfügbar gemacht hat.
   Verfügbar gemachte Rechte gehören zu den individuellen Eigenschaften.

8. **Anfragen oder Änderungen** beginnen stets mit einer Vollmacht des Benutzers.

   So stellt der Benutzer klar, welchem Zweck die Anfrage dient und damit insbesondere, welche Rechte er gerade verwenden will.

   Er richtet damit seine Anfrage (oder Änderung) an alle Personen, mit denen er bzgl. der von ihm gewählten Vollmacht bekannt ist.

Er kann diesen Adressatenkreis einschränken (z.B. durch Selektionsbedingungen) und das gelieferte Wissen der Befragten weiterverarbeiten.

Dazu steht dem Benutzer die **Sprachmächtigkeit der relationalen Algebra** zur Verfügung (sowohl um den Adressatenkreis zu berechnen (sog. Navigation) als auch um die zurückgelieferten Werte zu bearbeiten).

9. Jede Ausführung einer Operation ist Personen (und dadurch handelnden Menschen) zugeordnet.

   Um diese Zuordnung später nachvollziehen zu können, werden die Operationen im Informationssystem **protokolliert** (*remember*) (Nachvollziehbarkeit der Datenverarbeitung).

   Entsprechende Protokollierungen finden sowohl beim Anfragenden als auch bei den die Anfrage bearbeitenden Personen statt.
   Die Protokolldaten gehören zu den individuellen Eigenschaften.

   Es gibt **Datenschutzbeauftragte** als spezielle Personen.
   Diese (und nur diese) dürfen auch Protokollinformationen fremder Personen auswerten.
   Die Datenschutzbeauftragten bilden eine eigene Gruppe.

10. Die Abschirmung des Wissens eines Menschen erfolgt mit Techniken des objekt-orientierten Programmierens.

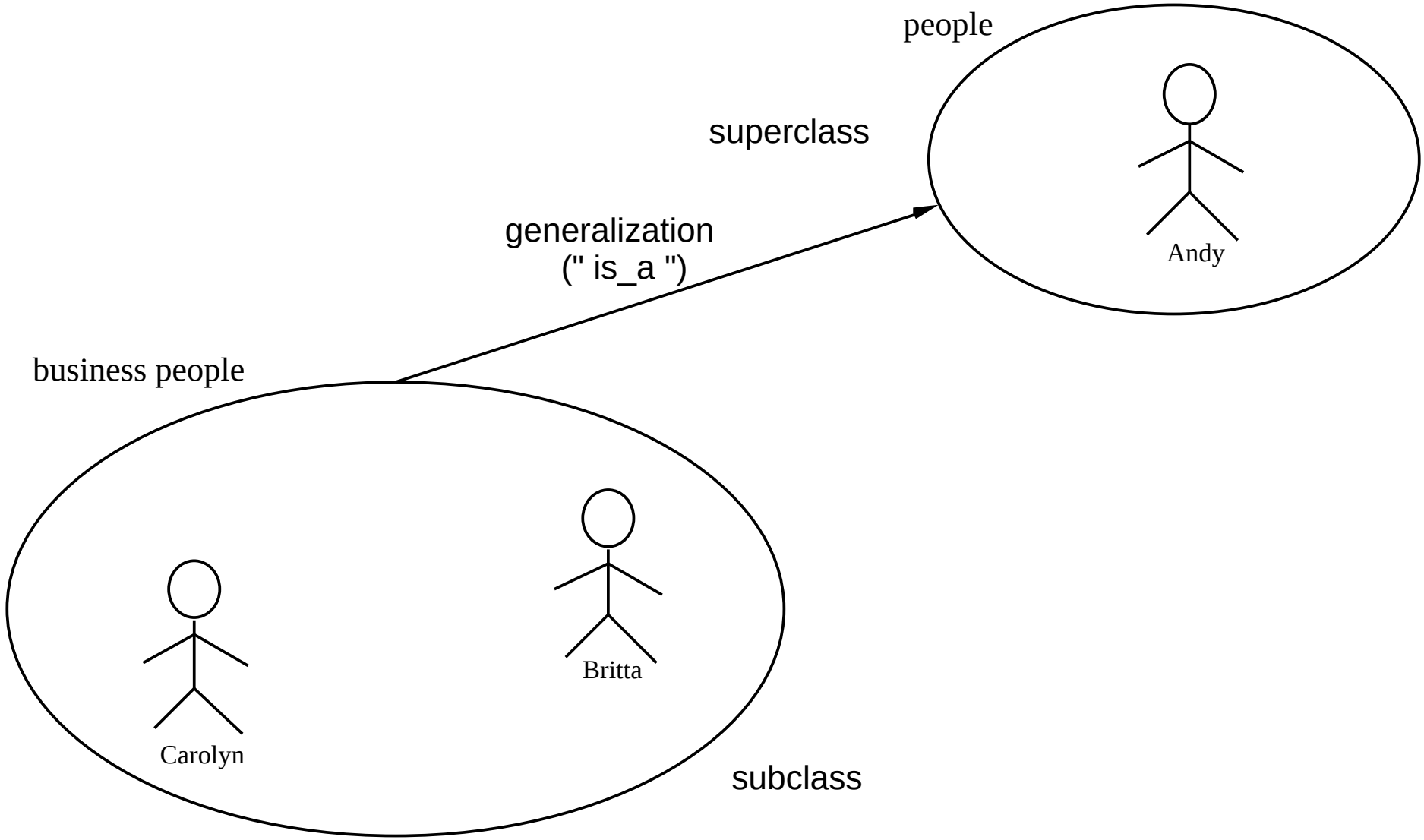
Es stellt sich als zweckmäßig heraus, nicht nur das Wissen eines Menschen als Einheit anzusehen, sondern die **Person** selbst (mit ihrer Identität, ihrem Wissen, ihren Rechten etc.) **als Objekt** (im objekt-orientierten Sinn) abzukapseln.

11. Aufteilung zwischen individuellen und strukturellen Eigenschaften:
In objekt-orientierten Programmiersprachen unterscheidet man zwischen strukturellen, gemeinsamen Eigenschaften einer Menge von Objekten und individuellen Eigenschaften von Objekten.

    Man fasst dann die Objektmenge zu einer **Klasse** zusammen und verwaltet die gemeinsamen strukturellen Eigenschaften bei der Klasse. (Analog unterscheidet man im Datenbankbereich zwischen den **Datenbankinstanzen** und dem **Datenbankschema**).

    Darüberhinaus gibt es eine **Klassenhierarchie**, mit deren Hilfe Struktureigenschaften vererbt werden.
    Diese Mechanismen dienen der Komplexitätsreduktion und erhöhen so insbesondere die Durchschaubarkeit und Fehlerfreiheit.

Daher werden Personen mit gleicher Struktur (des Wissens, der Rechte etc.) zu **Gruppen** zusammengefasst und die Gruppen in **Hierarchien** geordnet (technisch objekt-orientiert über Klassen und Vererbungsmechanismus realisiert).



In der (abgeschirmten, privaten) **Beschreibung einer Person** sind fixiert:
• die Identität der Person,
• ihr Wissen,
• ihre Bekanntschaften bzgl. ihrer verschiedenen Vollmachten,
• die von ihr vergebenen vorübergehenden (verfügbar gemachten) Rechte sowie
• die Protokollierungsinformation.

Die Gruppenzugehörigkeit ist ein Zwitter: einerseits ist bei jeder Person fixiert, zu welcher Gruppe sie gehört, andererseits sollte bei jeder Gruppe die Menge der zugehörigen Personen bekannt sein.

In der (öffentlichen) **Beschreibung einer Gruppe** sind fixiert:
• die Einordnung der Gruppe in die Gruppenhierarchie,
• die Struktur des Wissens der Gruppe,
• die Operationen, die für Mitglieder dieser Gruppe definiert sind,
• die Vollmachten, die Mitglieder dieser Gruppe ausüben können, und
• die Rollen, die Mitglieder dieser Gruppe erleiden müssen.
Vererbt werden Wissensstruktur, Gruppendefinierte Operationen, Vollmachten und Rollen.

Struktur-(Schema-)informationen (Beschreibungen der Gruppen) sind grundsätzlich von jeder Person lesbar, d.h. öffentlich.

Neben der Komplexitätsreduktion dienen die Gruppen auch als Modellierungshilfe einer Organisation beim Entwurf des Systems.

Es ist damit möglich, Organisationsstrukturen zu beschreiben, ohne dass die einzelnen Personen, die die Positionen in der Organisationshierarchie wahrnehmen sollen, schon feststehen müssen.

Die syntaktische Aufteilung der Rechte in Vollmachten und Rollen ist i.a. ungewohnt und besonders die Zuordnung der Rolle nicht zur Gruppe der handelnden Personen, sondern zur Gruppe der erleidenden Personen.

Dagegen sind die Vollmachten (also die Rechte zur Ausführung von Rollen) bei der Gruppe der handelnden Personen beschrieben.

Durch diese Aufteilung der Rechte kann jedoch leicht ermittelt werden, was Mitglieder einer Gruppe maximal tun können (Überprüfe Vollmachten!) bzw. erleiden (Überprüfe Rollen!) müssen.

Darüberhinaus wird der Entwurf der Strukturinformation leichter: I.a. wird zunächst festgelegt, wie die Rollen aussehen und erst in einem späteren Entwurfsschritt bestimmt, wer diese Rollen (für welchen Zweck) ausüben darf.

12. Außerdem können Personen Zugang zu **Dokumenten** haben.

    Dokumente sind Objekte, die nicht (mehr) verändert werden können.

    Mit Hilfe von Dokumenten kann Wissen, das mehrere Personen gemeinsam haben, dargestellt werden, z.B. Verträge.

Weitere Konzepte in DORIS betreffen Weitergabe von Bekanntschaften, Mechanismen zur Delegation sowie Servicevollmachten.

8.5 Ein kleines DORIS-Beispiel

Beispiele für DORIS-Gruppenbeschreibungen:

```
GROUP          All          IS
SUPERGROUPS
```

(Falls eine Gruppe Wurzel im Gruppenbaum ist, also keine Supergruppe hat, gibt es folgende syntaktische Möglichkeiten: SUPERGROUPS All; oder SUPERGROUPS oder keine Supergroups-Klausel.)

```
ATTRIBUTES     name :       RECORDOF   last_name:   STRING;
                                       first_name:  STRING    END;
               address:     RECORDOF   street:      STRING;
                                       city:        STRING;
                                       zip:         STRING    END;
               birthday:    STRING;
OPERATIONS
ROLES          inspect_my_data = read[*];
```

(read wird in Anlehnung an die Grundoperation statt tell benutzt. Der * ist ein Pseudo-Attribut und bedeutet alle Attribute. Bisher hieß kein Eintrag (= keine Restriktion), dass alle Attribute gelesen werden konnten. Der * verhindert, dass bei vergessener Attributangabe dann versehentlich alle Attribute gelesen werden dürfen.)

```
AUTHORITIES    self_read = I : inspect_my_data;
```

(Eigentlich steht nach dem Gleichheitszeichen ein Gruppenname, der besagt, bzgl. welcher Gruppe die Vollmacht gilt. Durch die Pseudo-Gruppe I wird festgelegt, dass die Vollmacht nur bzgl. der eigenen Person gilt: Jeder darf mittels self_read nur seine eigenen Daten lesen.)

```
GROUP          Employee       IS
SUPERGROUPS    All;
ATTRIBUTES     employee_no:   INTEGER;
               salary:        INTEGER;
OPERATIONS
ROLES          hire = create[Employee], insert[name, address, birthday, employee_no, salary];
               earn = read[salary];
               change_in_salary = read[salary], modify[salary];
               fire = dispose[Employee];
AUTHORITIES    self_earn = I : earn;

GROUP          Personnel_Manager          IS
SUPERGROUPS    Employee;
ATTRIBUTES
OPERATIONS
ROLES
AUTHORITIES    hire_auth = Employee : hire;
               change_in_salary_auth = Employee : change_in_salary;
               fire_auth = Employee : fire;
```

Beispiele für DORIS-Personenbeschreibungen:

SURROGATE	<MILLER>		
GROUPS	Employee		
KNOWS	name:	last_name:	MILLER
		first_name:	
	address:	street:	LITTLE RD
		city:	
		zip:	
	birthday:	1952 03 21	
	employee_no:	37	
	salary:	40,000	
ALIVE	TRUE		
ACQUAINTED	self_read:	{<MILLER>}	
	self_earn:	{<MILLER>}	
AVAILABLE			
REMEMBERS			

SURROGATE	<WILLIAMS>		
GROUPS	Employee		
KNOWS	name:	last_name:	WILLIAMS
		first_name:	
	address:	street:	MAIN ST
		city:	
		zip:	
	birthday:	1954 06 14	
	employee_no:	12	
	salary:	42,000	
ALIVE	TRUE		
ACQUAINTED	self_read:	{<WILLIAMS>}	
	self_earn:	{<WILLIAMS>}	
AVAILABLE			
REMEMBERS			

SURROGATE	<PARKER>		
GROUPS	Personnel_Manager		
KNOWS	name:	last_name:	PARKER
		first_name:	
	address:	street:	LUCKY RD
		city:	
		zip:	
	birthday:	1953 02 02	
	employee_no:	7	
	salary:	55,000	
ALIVE	TRUE		
ACQUAINTED	self_read:	{<PARKER>}	
	self_earn:	{<PARKER>}	
	hire_auth:	EMPLOYEE	
	change_in_salary_auth:	{<MILLER>, <WILLIAMS>}	
	// Parker darf nicht sein eigenes Gehalt ändern!		
	fire_auth:	{<MILLER>, <WILLIAMS>}	
AVAILABLE	{ ({<WILLIAMS>}, {<MILLER>, <WILLIAMS>}, fire_auth, 1992 08 30) }		
REMEMBERS			

Beispiele für DORIS-Anfragen:

von MILLER: (* Miller fragt zunächst nach seinem Gehalt, später nach seiner Angestelltennr und Geburtsdatum *)
self_earn: ACQ TELL salary;
self_read: ACQ TELL employee_no, birthday;

von PARKER: (* Parker gewährt Williams eine 4,5%ige Lohnerhöhung *)
change_in_salary_auth: ACQ KNOW last_name='WILLIAMS' MODIFY salary BY salary * 1.045;

(* Parker fragt nach Angestellten, mit einem Gehalt > 40000 *)
change_in_salary_auth: ACQ TELL last_name, salary WHERE salary > 40,000;